dSCope Series III

Scripting Manual

by Liam Elliott

This manual is also available as 'on-line help' from the dScope software. You can access the on-line help from the 'Help' menu. The on-line version is context-sensitive: by pressing F1, you can get immediate help for whichever menu or dialogue box you are currently using.

Table of Contents

Part 1

General information

1 General information

Manual revision history

Rev	Date	Author	Notes
1.00	7th Jan 2003	L. R. Elliott	To accompany software V1.00
1.01	10th Dec 2003	L. R. Elliott	To accompany software V1.01
1.01a	2nd Jun 2004	L. R. Elliott	To accompany software V1.01a
1.10	5th May 2005	L. R. Elliott	To accompany software V1.10
1.11	13th Jul 2005	L. R. Elliott	To accompany software V1.11
1.20	8th Jan 2007	L. R. Elliott	To accompany software V1.20
1.20b	29th Mar 2007	L. R. Elliott	To accompany software V1.20b
1.21	14th Apr 2007	L. R. Elliott	To accompany software V1.21
1.30	22nd Jun 2009	L. R. Elliott	To accompany software V1.30
1.30a	11th Feb 2010	L. R. Elliott	To accompany software V1.30a
1.30b	4th Mar 2010	L. R. Elliott	To accompany software V1.30b
1.41	2nd May 2012	L. R. Elliott	To accompany software V1.41
1.42	17th August 2012	L. R. Elliott	To accompany software V1.42
1.43a	18th January 2013	L. R. Elliott	To accompany software V1.43a
1.44	9th April 2013	M.C.Harvey	To accompany software V1.44
1.45	6th November 2013	K.S.C.Stubbs	To accompany software V1.45

Support contacts

Prism Media Products Limited
The Old School
Stretham
Ely
Cambridgeshire CB6 3LD
UK

Telephone: +44 1353 648888
Fax: +44 1353 648867

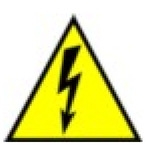
Prism Media Products Inc
21 Pine Street
Rockaway
NJ 07866
USA

Telephone: +1 973 983 9577
Fax: +1 973 983 9588

Email: tech.support@prismsound.com
Web: http://www.prismsound.com

Or contact your local Prism Sound distributor as detailed on the website.

WARNING!



TO PREVENT FIRE OR SHOCK HAZARD DO NOT EXPOSE THIS EQUIPMENT TO RAIN OR MOISTURE. DO NOT REMOVE THE COVER. NO USER-SERVICEABLE PARTS INSIDE. REFER SERVICING TO QUALIFIED SERVICE PERSONNEL.

Statements of conformity

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against interference in a residential area. This device generates and uses radio frequency energy and, if not installed and used in accordance with the instructions, may cause interference to radio or TV reception. If this unit does cause interference to radio or TV reception, please try to correct the interference by one or more of the following measures:
a) Reorient or relocate the receiving antenna.
b) Increase the separation between the equipment and the receiving antenna.
c) Plug the equipment into an outlet on a different circuit from the receiver.
d) If necessary, consult your dealer or an experienced radio or TV technician.
CAUTION: Changes or modifications to this equipment not expressly approved by the manufacturer could void the user's authority to operate this equipment.

THIS DIGITAL APPARATUS MEETS ALL CLASS B LIMITS FOR RADIO NOISE EMISSIONS AS LAID DOWN IN THE RADIO INTERFERENCE REGULATIONS OF THE CANADIAN DEPARTMENT OF COMMUNICATIONS.
CET APPAREIL NUMÉRIQUE RESPECTE TOUTES LES EXIGIENCES APPLICABLES AUX APPAREILS NUMÉRIQUES DE CLASSE B SUR LE BROUILLAGE RADIOELECTRIQUE EDICTE PAR LE MINISTERE DES COMMUNICATIONS DU CANADA.

Prism Media Products Ltd hereby declares that this equipment conforms to the following standards:
EN55103-1, environment category E4
EN55103-2, environment category E4

NOTE: The use of this equipment with non-shielded interface cabling is not recommended by the manufacturer and may result in non-compliance with one or more of the above directives. All coaxial connections should be made using a properly screened 75R cable with the screen connected to the outer of the connector at both ends. All XLR connections should use a screened twisted pair cable with the screen connected to pin 1 of the XLR connector at both ends. In the case of the digital XLR connections this cable should be of 110R impedance.

Trademark acknowledgements

Access, ActiveX, Excel, Microsoft, MS-DOS, Visual Basic, VB, VBA, VBScript, Visual C++ and Windows are trademarks of Microsoft Corporation.
Borland and Delphi are trademarks of Borland Software Corporation.
LabVIEW and LabWindows/CVI are trademarks of National Instruments Corporation.
Dolby and the double-D symbol are trademarks of Dolby Laboratories.
DTS is a trademark of DTS, Inc.
Audio Precision is a registered trademark of Audio Precision, Inc.

All trademarks acknowledged.

In accordance with our policy of continual development, features and specifications are subject to change without notice.

Part 2

Scripting and OLE Automation

2 Scripting and OLE Automation

Scripting and OLE Automation are powerful techniques by which the basic functionality of the dScope may be extended to fulfil a wide range of user-specific operations.

The dScope's scripting language is Microsoft "VBScript", which is enhanced by a wide range of dScope-specific functions and variables made available by a simple "drag-and-drop" feature in dScope's built-in script editor. Alternatively, the dScope can be automated externally, by accessing its properties and methods from another application or by using any "Active-X Scripting" language from an external scripting host.

Documentation on the VBScript programming language can be downloaded from the Microsoft web site.

To find out the answers to common automation questions, see How do I...

If you're having problems with an issue to do with scripting or automation, see Common scripting problems.

The dScope scripting reference contains full details of every property and method of the dScope available to anyone writing a script or externally controlling the dScope.

For information about the Script Edit window in dScope, see the section on the Script Edit window.

2.1 Ways of automating dScope

This section summarises the ways that the dScope Series III software can be controlled using any of the available automation methods - from a script within dScope, scripting from another Windows program, or by writing a program in another language. It gives brief examples for each method to enable you to get started.

See Principles of automation for a description of the concept of OLE Automation and how the dScope uses this functionality of Windows to allow other programs to control it.

Automation from within dScope
Automation from the Windows Scripting Host
Automation from VB or VBA (Visual Basic or Visual Basic for Applications)
Automation from LabVIEW
Automation from LabWindows/CVI
Automation from C++
Automation from Delphi
Automation from C#

Automation from within dScope

Within the dScope program, the software will respond to scripts written in VBScript. This is a standard Microsoft language that is basically a subset of Visual Basic. Scripts can be run from within dScope by selecting the Run Script option from the Automation menu. This allows you to browse for a script, which is then run when the [Open] button is selected.

The dScope software contains a script editing environment, available by selecting the Edit Script option from the Automation menu. This brings up an editing window, with a tree on the right hand side detailing all the properties that can be set and the methods that can be called. Double-clicking on the relevant method or property will insert it into the script on the left at the current cursor position. When you select a property that has a possible range of values, these values will be shown in a list at the bottom right of the window. Double-clicking on a value in this list will also insert this item at the current position in the script.

Note that double-clicking a dScope script (.dss file) from the Windows shell will start the dScope software, and then run that script.

Automation scripting from within dScope is detailed in the Automation scripts section.

Automation from the Windows Scripting Host (WSH)

The Windows Scripting Host is built into the Windows 98, Windows 2000 and Windows XP operating systems. It simply allows a script to be run directly from the operating system. This script can be written in VBScript (.vbs file) or JavaScript (.js file). Double-clicking on a file with either of these extensions will automatically cause the Windows Scripting Host to try to run that file.

Running a script from the Windows Scripting Host is slightly different to running a script from within the dScope software. The main reason for this is that when the script starts running, Windows does not know about the dScope at all. The first thing that the script needs to do is to create a dScope object; further lines of the script then use this object to access the various properties and methods of that object, for example:

```
' Declare Variables
Dim dScope
Dim DI

' Initialise the dScope object
Set dScope = CreateObject("dScope.Application")

' Wait until initialized
While Not dScope.IsInitialised()
   dScope.Sleep(1)
Wend

' Initialize an object for the Digital Inputs
Set DI = dScope.DigitalInputs

' Perform a couple of operations - in this case,
' loading a configuration and setting the Digital
' Input source
dScope.LoadConfiguration("Example.dsc")
DI.DI_Source = DI.DI_SOURCE_XLR
```

Automation from VB or VBA

Automation from VB (Visual Basic) or VBA (Visual Basic for Applications) is done in a very similar way to automation from the Windows Scripting Host. VBA involves using the Visual Basic Editor contained within Microsoft Word, Microsoft Excel etc.

VB and VBA have the useful feature of the Object Browser, which allows the user to see all the properties and methods of the dScope application. To make use of this, dScope must be set up as a Reference. To do this, select the References option from the Tools menu of the Visual Basic Editor. This will bring up a list of the available references. If dScope is in the list, then simply check the box next to it and select OK. Otherwise, click on the Browse button to bring up the Add Reference file selection box. Ensure that the "Files of Type" box at the bottom contains "Executable files", navigate to the dScope folder, and select the dScope.exe file. Pressing Open will return you to the list of available references; ensure that dScope is ticked in the list and click on OK to add the dScope as a reference.

The dScope will now appear in the Object browser. Also, whenever you type in the name of a dScope object into the editor, a drop-list will appear of the object's methods and variables that you can call, for example:

```
' Declare variables of certain types
Dim DS As DScope.Application
Dim DI As DScope.DigitalInputs

' Initialise the variables
Set DS = CreateObject("DScope.Application")
Set DI = DS.DigitalInputs

' Wait until initialised
While Not DS.IsInitialised()
   DS.Sleep(1)
Wend

' Note that from this point onwards, if you type
' 'DS.', you will be shown a list of all the
' dScope methods and properties; similarly, typing
' 'DI.' will bring up a list of all the Digital
' Inputs methods and properties.

' Perform a couple of operations.
DS.LoadConfiguration ("Example.dsc")
DI.DI_Source = DI.DI_SOURCE_BNC
```

Events from within VB

Events in the dScope are part of the Automation object. Within VB, you need to set up the Automation object as **WithEvents**, which will then allow you to capture events fired from the dScope.

```
' Declare variables of certain types
Dim DS As DScope.Application
Dim Sweeps As DScope.Sweeps
Dim WithEvents DSEvents As DScope.Automation

' Initialise the variables
Set DS = CreateObject("DScope.Application")
Set Sweeps = DS.Sweep
Set DSEvents = DS.Automation

' Wait until initialised
While Not DS.IsInitialised()
   DS.Sleep(0)
Wend

' Note that from this point onwards, if you type
' 'DS.', you will be shown a list of all the
' dScope methods and properties; similarly, typing
' 'DI.' will bring up a list of all the Digital
' Inputs methods and properties.

' Perform a couple of operations.
DS.LoadConfiguration ("Sweep setup.dsc")
Sweeps.SW_Go()

' Events are then all prefixed with DSEvents_
Sub DSEvents_SweepFinished()
   MsgBox "Sweep finished!"
End Sub
```

Accessing constants from within VB

Two files are provided for use within a Visual Basic project automating the dScope. These are installed to the dScope program folder ("C:\Program Files\Prism Sound\dScope Series III" by default). These files are:

- dS3Const.bas for use in VB6 projects
- dS3Const.vb for use in VB.NET projects.

The relevant file can be included by adding the file to your project using "Add Module".

Automation from LabVIEW

1) On the LabVIEW front panel, right-click to bring up the "Controls" menu.

2) Select the ActiveX panel, click on "Automation RefNum", and add this to your panel.

3) Once you have added this object, right-click on it and select "Select ActiveX class" from the resulting menu, then the "Browse" sub-menu.

4) In the "Select object from Type Library" dialogue box, click on the [Browse] button.

5) Select "All files" in the "files of type" box at the bottom of this window. Browse for the dScope.exe file, which will be in the program folder you installed the software to ("C:\Program Files\Prism Sound \dScope Series III" by default).

Automation from LabWindows/CVI

LabWindows allows you to create a dScope instrument, which basically acts as a layer between your LabWindows code and the dScope software. To create the dScope instrument (in LabWindows Version 5) :

1) Select Create ActiveX Automation Controller from the Tools menu. LabWindows will search for a list of available automation servers.

2) Because the dScope's Type library is hidden within the dScope.exe program file, you may have to select "Browse" to search for the file. Once this file is opened, you will be shown a list of all the available objects in dScope.

3) Select as many (or as few) of the available objects in the upper list as you wish to include, and click on the [Generate] button. (Note that you may have to shorten some of the default names that are given).

4) Select target files for the dScope front panel, and click OK. Your dScope instrument will be created.

NB: **This process will have to be repeated every time the dScope automation interface changes, which may occur on new software releases.**

Some example code (assuming the default names are kept when creating the dScope instrument):

```
// Define objects for the dScope, and the
// Digital Inputs part of the dScope object
static CAObjHandle dscope_1;
static CAObjHandle dig_inputs;
static CAObjHandle sig_ana;

// Create these objects
DScope_NewIDScope (NULL, &dscope_1);
DScope_IDScopeDigitalInputs (dscope_1,
         NULL, &dig_inputs);
DScope_IDScopeSignalAnalyzer(dscope_1,
```

```
         NULL, &sig_ana);

// Set the Digital Inputs source to be back-to-back
DScope_SetProperty (dig_inputs, NULL,
DScope_IDigitalInputsDI_Source,
         CAVT_SHORT, DI_SOURCE_GENXLR);

// Set the Signal Analyzer unit to V
DScope_SetProperty (sig_ana, NULL,
         DScope_ISignalAnalyzerSA_RMSAmplUnit,
         CAVT_SHORT, UNIT_V);

// Close the dScope
DScope_IDScopeCloseApplication (dscope_1, NULL);
```

NB: **In LabWindows/CVI 7.0, you may need to use the GetObjHandleFromActiveXCtrl function to retrieve an object handle for the dScope, after using DScope_NewIDScope.**

Automation from C++

Automation from a C++ program uses a "wrapper" class around the dScope Type library, which allows your program to call any of the dScope methods or properties with ease. This wrapper class can easily be created using the Microsoft Visual Studio ClassWizard - if you are not using this development environment, please contact Prism Sound for further details.

You'll need to create your project as an OLE Automation container, which will correctly initialise all its OLE capabilities. This is easy using the AppWizard in the Visual Studio. You will also have to call CoInitialize(NULL) when your application starts up, and CoUninitialize() when it shuts down.

Creating the wrapper class from the Microsoft Developer Studio:

1) From your project workspace, click on the ClassWizard option from the View menu.

2) Click on the Add Class button and select From a Type library from the resulting menu.

3) Browse to the dScope.exe file in the dScope program folder and select Open.

4) Select all of the objects in the list, and click OK to create the wrapper class.
In any of your source files that will use dScope constants, #include the **ds3const.h** file provided with the application.

Example code using this wrapper class:

```
USES_CONVERSION;

// Open the dScope object
HRESULT  hr;
CString  str;
CLSID    clsid;
IDScope  dScope;

// Initialise COM
CoInitialize(NULL);

// String ID unique to this application
LPTSTR pstrProgID = _T("dScope.Application");

// Get the unique class ID for this string
hr = ::CLSIDFromProgID(T2OLE(pstrProgID), &clsid);
if (FAILED(hr)) {
```

```
   // Exit cleanly
} // End (if)

// Create the dispatch interface that we'll use to
// access the dScope object
dScope.CreateDispatch(clsid);

// Wait until the dScope software has initialised,
// so that we don't try and access bits of the
// dScope before it's ready.
while (!dScope.IsInitialised()) {
} // End (while)

// Access the Signal Analyzer settings
ISignalAnalyzer SigAna(dScope.SignalAnalyzer());

// Set the source to analogue and unit to dBu
// Note we must have "#include"d ds3const.h to
// get at these constants.
SigAna.SetSA_Source     (SA_DIGITAL);
SigAna.SetSA_RMSAmplUnit(UNIT_DBU);

// Read the amplitude
double dAmplitude = SigAna.GetSA_ChARMSAmpl();
str.Format("%.2f", dAmplitude);
AfxMessageBox(str);

// Clean up the dispatch interface
dScope.ReleaseDispatch();
dScope.DetachDispatch();

// Uninitialise COM
CoUninitialize();
```

Automation from Delphi

Automation from Delphi is very similar to scripting. It involves firstly creating an OLE Object to represent the dScope, and then accessing its properties and methods, for example:

```
var
   dscope : Variant;

begin
   dscope := CreateOleObject('dScope.Application');

// Set the Digital Inputs back-to-back
dscope.DigitalInputs.DI_Source =
      dscope.DigitalInputs.DI_SOURCE_GENXLR;

// Set the Signal Analyzer source and unit
dscope.SignalAnalyzer.SA_Source =
      dscope.SignalAnalyzer.SA_ANALOGUE;
dscope.SignalAnalyzer.SA_RMSAmplUnit =
      dscope.UNIT_DBU;
```

Automation from C#

There are a number of issues to consider when automating the dScope from within C#:

How to add dScope as a reference to the Project

- In the "Solution Explorer" window, right-click on the solution and select "Add Reference"
- Click on the "COM" tab (Visual Studio 2010 or before), or the "COM" item on the left hand side (Visual Studio 2012)
- Select the "dScope Series III" object.

NOTE that this creates a .NET library called "Interop.dScope" which acts as a wrapper around the dScope COM object.
You will probably need to remove and re-add the dScope as a reference, every time the dScope's automation interface changes, i.e. when new software versions are released.

dScope constants

The dScope software ships with a file called "dS3Const.cs" which contains all the constants available to the dScope defined in C#. This resides in the dScope program folder and you should add a link to this file into your project too, as follows:

- Right-click on the solution in "Solution Explorer"
- Select "Add" and "Existing item"
- Navigate to the dScope program folder and select the dS3Const.cs file
- Click the little arrow on the "Add" button and select "Add as Link".

To use this file, you will need to include the following at the top of the code file:

```
  using PrismSound.dScope3;
```

How to create the dScope object

```
  try
  {
      DScope.IDScope ds3 = new DScope.Application();
      while (!ds3.IsInitialised())
      {
          ds3.Sleep(1);
      }
  }
  catch (COMException ex)
  {
      Trace.WriteLine("Failed to create dScope object!\nError was: {0}",
ex.Message);
  }
```

NOTE that this 'dScope' object is the one-and-only dScope object, so you'll need to make sure this is available to all the rest of the program that wants to use it.

How to close the dScope at shutdown

Using the 'dS3' object that you created above:

```
  ds3.CloseApplication();
```

How to access the Signal Generator to change signal values

Create a variable of the right type (note that all parts of the dScope have their own interface that you should use, prefixed with 'I'):

```
DScope.ISignalGenerator signalGenerator = ds3.SignalGenerator();
```

Set various parts of the generator:

```
signalGenerator.SG_ChAFunction = dS3Const.SG_FUNCTION_SINE;
// Note: You must set the unit BEFORE the value
signalGenerator.SG_ChAAmplUnit = dS3Const.UNIT_DBU;
signalGenerator.SG_ChAAmpl = -10.0;
signalGenerator.SG_ChAFreq = 1000.0;
```

Other parts of the dScope can be accessed in a similar way, for example:

```
DScope.IAnalogueInputs analogueInputs = ds3.AnalogueInputs();
analogueInputs.AI_Impedance = dS3Const.AI_IMPEDANCE_600R;
```

or, just

```
dS3.AnalogueInputs().AI_Impedance = dS3Const.AI_IMPEDANCE_600R;
```

How to use the Signal Analyzer to read values

Again, note that the Signal Analyzer has its own interface, prefixed with 'I':

```
DScope.ISignalAnalyzer signalAnalyzer = dS3.SignalAnalyzer();

signalAnalyzer = dScope.SignalAnalyzer();
signalAnalyzer.SA_RMSAmplUnit = dS3Const.UNIT_DBFS;
double rmsA = signalAnalyzer.SA_ChARMSAmpl;
double rmsB = signalAnalyzer.SA_ChBRMSAmpl;
```

How to use dScope Events

The dScope's Automation object exposes all events that the dScope can fire, as an interface called `IDScopeEvents_Event`. To respond to events, just hook into the relevant event(s) on this interface: For example:

```
DScope.IDScopeEvents_Event events = dS3.Automation();
events.BufferProcessed += OnBufferProcessed;
```

Unhook from the event in the following way:

```
events.BufferProcessed -= OnBufferProcessed;
```

When the event occurs, the event handler method `OnBufferProcessed` will be called:

```
// Handles FFT Buffer processed event
private void OnBufferProcessed()
{
    // Take appropriate action here…
}
```

(Note: Don't forget to turn on the relevant event in the Event Manager, to enable the event that you want to respond to!)

How to use functions that return values as parameters

Some events in the dScope return values in their parameters, as “reference parameters”. The way that C# accesses these is slightly complicated. You need to create a VariantWrapper object around the value that you want returned, as in the following example (to retrieve the current minimum and maximum Y values of a Trace scale).

```
  // This assumes that you already have an object ‘trace’, of type ‘
DScope.ITrace’
  double min = 0.0;
  double max = 0.0;
  object objMin = new VariantWrapper(min);
  object objMax = new VariantWrapper(max);
  trace.TRACE_GetYRange(ref objMin, ref objMax);
  min = (double)objMin;
  max = (double)objMax;
```

2.2 Principles of automation

The Windows operating system allows different pieces of software to talk to and control each other using a process known as OLE Automation. If a program wants to allow something else to control it, it must expose properties and methods to the outside world.

The dScope Series III software utilises this functionality to allow considerable flexibility for automation of testing.

Automation in the dScope

The dScope III software exposes an interface, defined in a standard language called "Object Definition Language", to the Windows operating system. This interface is defined in a Type library, and any Windows program that can control automation-enabled objects can control the dScope.

NB: **The Type library in dScope is built into the executable file dScope.exe which resides in the dScope program folder. This is to ensure that the Type library is always up-to-date with the software version. However, if you are trying to control the dScope from another program, this may mean that Windows cannot find the file automatically and will ask you to browse for this program when it needs to know about the Type library.**

2.3 Hints and tips

This section contains some hints and tips for writing scripts in the dScope.

Re-using VBScript functions

Sometimes it is desirable to create useful functions in VBScript that can be re-used over and over again in new scripts that you write. Rather than re-write these routines for every script, the **#Include** feature can be used.

The **#Include** feature can be used at the top of a script to specify that you wish to be able to use any of the functions defined in that script. It has the following format:

' **#Include <filename>**

where **<filename>** is the name of any valid script file, in quotation marks. A full path name can be specified (for example "C:\Program Files\Prism Sound\dScope Series III\Scripts\Automation\Useful.

dss", or just a file name. If a full path is not specified, then the dScope will look for the script in the same folder as the script that is running.

NB: **Note that since the '#Include' feature is dScope-specific, the VBScript compiler does not understand it, so it must be contained within a comment. Scripts running from outside the dScope will simply ignore this command.**

When a file is included in this way, VBScript treats this file as if it has been inserted into the main script. Any functions, declarations or variables in the included script are then also available to the main script, and vice versa.

NB: **Any files included in this way cannot contain any code that runs directly; all code must be contained within functions or subroutines.**

For example, the top of a script may look like the following:

```
' TYPE              Automation
' DESCRIPTION       This script demonstrates use of
'                   the Include feature

' #Include "ConversionRoutines.dss"
' #Include "Useful.dss"
```

Multiple script lines

VBScript, by default, requires you to type a command on a single line, without any carriage returns at the end of the line. However, you may wish to split a line of code neatly onto more than one line.

To do this, put the underscore ("_") character at the end of each line that wraps onto the next line.

For example:

```
strMesg = "This is a string that " & _
          "goes over more than " & _
          "one line"
MsgBox strMesg
```

2.4 How do I...?

This section is intended to provide quick answers to "How do I..." questions concerning scripting of the dScope.

For a more detailed description of all the dScope's commands available to scripts, see dScope scripting reference.

Controlling other applications from VBScript

How do I use Microsoft Word to output a test report using scripting?

See Automation of Microsoft Word.

How do I output test results to a Microsoft Access database?

See Automation of Microsoft Excel.

Different ways of automating the dScope

How do I control the dScope from LabVIEW?

See Ways of automating dScope - Automation from LabVIEW.

How do I control the dScope from LabWindows/CVI?

See Ways of automating dScope - Automation from LabWindows/CVI.

How do I write a C++ program to control the dScope?

See Ways of automating dScope - Automation from C++.

How do I write a Delphi program to control the dScope?

See Ways of automating dScope - Automation from Delphi.

Detectors

How do I create and use FFT Detectors from within a script?

See Creating and accessing FFT Detectors.

Sweeps and Traces

How do I access Traces from within a script?

See Creating and accessing Traces.

Miscellaneous

How do I re-use functions from one VBScript in another?

See Re-using VBScript functions.

2.5 Common scripting problems

This section is intended to provide quick answers to a number of problems commonly experienced when trying to control the dScope via its automation interface.

If you're having problems working out how to do something, you could try How Do I....

Problems in the Script Edit window

When I type a comment or string that goes over two lines, it gets formatted incorrectly.

The Script Edit window currently has some problems recognizing comments or strings that go over more than one line. It works okay until a keyword is entered (for example, "And" or "For") but then decides to reformat the text around this word. Because the comment or string did not start on the same line, it doesn't know that this is a comment or string and so changes the colours of the words on the line.

This will be fixed in a future software version, but in the meantime you can work around the problem by starting each line of the comment or string on a new line. For example:

```
' This is a long comment,
' that goes over more than one line
```

or:

```
str = "This is a long string "
str = str & "that goes over more than one line "
str = str & "in the Script Edit window "
```

Miscellaneous problems

I don't know what values to pass to a function

Many of the functions available to the dScope take certain values as parameters. These values are often constants that are equivalent to items in the software's drop-lists (for example, the Digital Inputs Source can have values of DI_SOURCE_XLR, DI_SOURCE_BNC, DI_SOURCE_TOSLINK, DI_SOURCE_GENXLR or DI_SOURCE_GENBNC).

When using the dScope's Script Edit window, the bottom right-hand corner will list the available constants that you can pass a function.

When controlling the dScope from an external programming language, all the available constants are in the following files:

- "**ds3const.h**" - for C and C++ languages.
- "**ds3const.bas**" - for Visual Basic.

These files will be copied into the dScope program folder on installation.

Events in my script don't fire

There may be a number of reasons why events do not fire in a script:

1) Check the Event Manager. (Select "Event Manager" from the Automation menu). This contains a list of all the events that can go off.
The Event Manager itself must be turned on (the check box at the top of the window), and the event that you require must also be turned on (the check box in the left-hand column, against the event in question).

2) If you are using the ScriptDlg ActiveX Control in your script, the ScriptDlg form must be created as a modeless ScriptDlg.

3) The script must be event-driven.
The dScope can run scripts as simply a sequence of commands. However, VBScript will then not call any event subroutines until the script has finished running.
To counter this, you must ensure that the main body of your script is enclosed in a function called dScope_Main. This routine will get called when the script is first run, and because it's actually an event subroutine itself, will allow other event subroutines to be called even while this subroutine is executing.

For example, the following code will not successfully fire the events, because the script is stuck in the

While loop and cannot run the event code:

```
Sweeps.SW_Go()
While Not Sweeps.SW_IsSweepFinished()
    ' Wait. Note that the script will not fire events
    ' while in this loop...
Wend
MsgBox "Sweep finished!"

Sub Event_MinTraceLimit(lParam)
  MsgBox "Trace limit breached!"
End Sub
```

This code would need to be re-written to be event-driven, i.e. with the main body of code within the dScope_Main event routine, as follows:

```
Sub dScope_Main

  Sweeps.SW_Go()
  While Not Sweeps.SW_IsSweepFinished()
     ' Wait
  Wend
  MsgBox "Sweep finished!"

End Sub ' dScope_Main

Sub Event_MinTraceLimit(lParam)
  MsgBox "Trace limit breached!"
End Sub
```

When I try to specify an amplitude, it tells me that it's an invalid value.

When you specify a value in a specific unit, you must ensure that the unit is set correctly before trying to alter the value.

For example, if the generated signal is currently in dBFS, then the following VBScript code will not work :

```
SignalGenerator.SG_ChAAmpl     = 10.0
SignalGenerator.SG_ChAAmplUnit = UNIT_DBU
```

This is because at the point the amplitude is set, the unit is still dBFS, so it checks the value specified in the current unit and determines that 10.0 is too high.

Changing the order of the commands, as follows, will solve the problem.

```
SignalGenerator.SG_ChAAmplUnit = UNIT_DBU
SignalGenerator.SG_ChAAmpl     = 10.0
```

2.6 Issues with software upgrades

The following sections give details of changes that have been made to the dScope for software upgrades, that may affect the operation of existing scripts or other automation of the dScope software (for example, from a language such as Visual Basic).

After upgrading to a new version of the dScope software, please review the following sections to see if they will affect the way your automated tests work.

Version 1.30

The software upgrade to V1.30 contained the following changes that may affect existing scripts:

- Changes to the way Digital Events are fired to external applications (e.g. VB6)

Changes to the way Digital Events are fired to external applications (e.g. VB6)

Software versions before V1.30 exposed some events to external applications with event names that contained the underscore character ("_"). This became a problem when trying to control the dScope from a VB6 application.

From V1.30 onwards, the following events have been renamed:

ChannelCheckFailed_ChA	to ChAChannelCheckFailed
ChannelCheckFailed_ChB	to ChBChannelCheckFailed
CS_ProfBit	to CSProfBit
CS_CopyrightBit	to CSCopyrightBit
CS_Emphasis	to CSEmphasis
CS_ChannelMode	to CSChannelMode
CS_CRCError	to CSCRCError
CS_ANotEqualToB	to CSANotEqualToB

Version 1.11

The software upgrade to V1.11 contained the following changes that may affect existing scripts:

- Changes to the way I/O Switcher Channel Arrays are stored
- Changes to the way Reading min/max values behave

Changes to the way I/O Switcher Channel Arrays are stored

Version 1.11 allows Channel Arrays to be set up and controlled using the dScope software, rather than just from a script (as was the case with previous versions). This means that the details of these Channel Arrays are saved and re-loaded with Configurations.

In older versions of the software, a script using Channel Arrays used to take the following actions:

Script:	Sets up Channel Arrays.
Configuration:	Loaded by script. The Configuration has no knowledge of the existence of Channel Arrays, so it leaves the existing arrays as they are.
Script:	Uses the Channel Arrays set up earlier.

In V1.11, the software DOES know about Channel Arrays. When a Configuration is saved using V1.11, if there are no Channel Arrays on the system, this information is saved with the Configuration.

So, using the same script as before, the following happens:

Script:	Sets up Channel Arrays.
Configuration:	Loaded by script. The Configuration knows about Channel Arrays, but also knows that none were in the Configuration at the time of loading. So the software deletes any existing Channel Arrays in the process of "clearing up" before the load.
Script:	Tries to use the Channel Arrays set up earlier, but this fails because they no longer exist.

To avoid this happening, ensure that you save your Configuration WITHOUT details of Channel Arrays. That way, when it re-loads, the software will not try to clean up existing arrays before loading new ones.

To do this, save the Configuration using "Save As" rather than "Save". When the "Save As" dialogue box opens, it contains a tree of items to save on the right-hand side. Open up "dS-NET peripherals" and ensure that "Channel Arrays" is UN-selected before saving the Configuration.

Changes to the way Reading min/max values behave

In Version 1.11, Readings have been changed so that Min and Max values ("peak hold" values) are no longer updated if they are not turned on. In addition, resetting of these values is no longer done automatically by the software when the Peak Hold values are turned on. Resetting must be done explicitly either using the user interface, or (via automation) using the Reading property RDG_ResetMinAndMaxValues.

The reason for this is twofold:

1) It does not make much sense for a Reading's min and max values to be updated if they are not actually being used;
2) The new method of operation allows multiple channels or devices to be measured using the same Reading (and therefore the same min and max values). Automation of the dScope simply needs to turn the peak hold values off during the channel or device swap, thus avoiding any erroneous readings (drop-out of signal etc) associated with the change.

For example, the changes to V1.11 now allow the following operation to measure the peak values across a range of channels:

```
Reading.RDG_ResetMinAndMaxValues()
For iChannel = 1 To NumChannels
    ' Turn OFF Min/Max around channel change
    Reading.RDG_ShowMinAndMaxValues = False
    ChannelArray.CA_ExclusiveChannel(iChannel)
    ' TODO: Wait for channel change to settle
    Reading.RDG_ShowMinAndMaxValues = True
    ' TODO: Make measurements
Next
```

This ensures that any change of level, drop-out etc associated with changing a channel does not affect the peak hold values on the Reading.

However, this means that existing tests must ensure that the min and max values are turned ON before starting measurements where the peak hold values are relevant. This can be done in one of two ways:

1) Using automation, set the Reading's RDG_ShowMinAndMaxValues property to **True**.
2) Load the Configuration that is being used for the test. Turn the Reading's min and max values ON using the Reading Properties dialogue box, before re-saving the Configuration.

Part 3

Types of dScope script

3 Types of dScope script

The dScope uses its built-in scripting capabilities in a number of ways which allow flexibility of various different areas of the software. The types of script are listed and described below.

Types of dScope Script

The following types of script can be created and used in the dScope:

General automation
Detector Functions
Generator wavetable scripts
FFT Detector Weighting filter scripts
FFT Detector Window function scripts
FFT Detector Calculation scripts
Sweep data tables
Limit Table scripts

Specifying the script type in a script

The top of every script should contain a comment line detailing the type of script it is. It is also useful here to add a brief description of the script and any other relevant comments.
The top line of the script should contain a comment with the word "TYPE" followed by the type of script it is.
For example:

```
' TYPE CT Detector Function
```

The following text is valid for the different script types:

- `Automation`
- `Generator wavetable`
- `FFT Window`
- `FFT Detector Weighting filter`
- `CT Detector function`
- `FFT Detector Function`
- `FFT Detector Calculation`
- `Limit Table`
- `Sweep data table`

Note that if you are editing your script in the Script Edit window, then changing the script type in the Toolbar's drop-list will automatically edit the comment at the top of the script to contain the correct text.
Altering your script in this way will also ensure that the system chooses the correct default folder when saving the script.

3.1 Automation scripts

The main use of the dScope's scripting capabilities is to automate test and measurement processes, so that complex tests can be performed at the click of a button.

Every property that can be set manually in the dScope application can also be controlled via automation. In this way, rather than having to manually change settings one at a time, a script can be run that will do all of this for you.

Any Configuration file can also be loaded as part of a script, to make testing even easier; the Configuration can be loaded to give an initial test setup, and then individual settings can be changed for particular tests.

For a full list of all functions available to automation scripts, see the dScope scripting reference section.

3.2 Detector functions

Detector scripts are used by dScope to define the settings of the Continuous-Time Detector and FFT Detector dialogue boxes to implement each measurement function. Each Detector script is effectively an automation script that sets up the details of the Detector.

For example, the "THD+N - relative" script performs the following steps:

```
' NAME          THD+N - relative.DSS
' TYPE          CT Detector Function
' DESCRIPTION   Function script used to set details of
'               Continuous-Time Detector
'
CTDetector.CTD_BPBRMode            = CTD_BPBRMODE_BR
CTDetector.CTD_BPBRBandwidth       = CTD_BPBRBANDWIDTH_3
CTDetector.CTD_BPBRFreqMode        = CTD_BPBRFREQMODE_INPUT
CTDetector.CTD_HPFilter            = CTD_HP_DEFAULT
CTDetector.CTD_LPFilter            = CTD_LP_DEFAULT
CTDetector.CTD_WeightingFilter     = CTD_WEIGHTING_DEFAULT
CTDetector.CTD_Response            = CTD_RESPONSE_RMS
CTDetector.CTD_Relativity          = CTD_RELATIVITY_SELF
```

NB:

These scripts simply set up the initial state of the Detector. After this, any of the fields of the Detector can still be altered, so it's possible to end up with a Detector whose settings are completely different from the function that it purports to be!

If settings are altered, the dScope's user interface will show an asterisk (*) in the title bar of the Detector, to indicate that it has changed from the default.

During a dScope session, any changes to the settings of a particular function will be remembered (if set using the Options setting OPT_RememberDetectorDetails). For example, changing the "THD+N - relative" function's BP/BR bandwidth from 1/3 octave to 1/6 octave will mean that every time this particular function is re-selected, the BP/BR bandwidth change will be remembered as 1/6 octave. The title bar of the Detector will show an asterisk (*) to indicate that a change has been made to the settings specified by the original script.

If you want the setting changes to be remembered in-between sessions, you should create a new script (or edit an existing script) to make these changes. If you create a script with a script type of "CT Detector Function" and save it in the same folder as all the other CT Detector function scripts, it will be added to the list of selectable functions in the Continuous-Time Detector's "function" drop-list.

3.3 Generator wavetables

Generator wavetable scripts allow the user to define custom waveforms for the Signal Generator.

As a Generator wavetable script runs, it will fill a buffer with sample values. This buffer can then be directly downloaded into the hardware, or used to create a waveform file; this waveform file can then be used at a later date instead of the script file. The waveform file has the advantage that it's quicker to download (because the calculation of each sample value doesn't have to be done) but the disadvantage that it's just a series of numbers and can't access any of the other dScope settings and use them to create the buffer.

For a full list of the functions available to create user-defined wavetables, see the Generator wavetable reference section.

3.4 FFT Detector Weighting filters

Weighting filter scripts allow the user to define custom weighting filters for use with the FFT Detectors. These can be user-defined weightings, high-pass or low-pass filters, or a combination of the three.

These scripts simply fill a table with a series of gain factors to apply to the FFT buffer

For a full list of all functions available to weighting filter scripts, see the Weighting filter reference section.

3.5 FFT Detector Window functions

FFT window scripts allow the user to define custom Window functions for use by the FFT Analyzer.

These scripts simply allow the user to specify a table of gain factors that will be applied to the sample buffer. This ensures that the effect of any discontinuities at the start and end of the buffer used for FFT calculations is minimized. By selecting a particular window funcion, it is possible to trade off dynamic range versus broadening of tonal components in the resulting FFT.

For a full list of all functions available to weighting filter scripts, see the FFT Window reference section.

3.6 FFT Detector Calculation scripts

Detector Calculation scripts allow the user to define complex functions for FFT Detectors, which transcend the mere setting of FFT Detector parameters. A Detector Calculation script actually processes the data from the bins of the FFT buffer, applying any desired algorithm to produce the custom result.

An FFT Detector Calculation script has access to the incoming sample buffer used for the FFT, as well as the FFT buffer before and after filters have been applied.

For a full list of all functions available to FFT Detector Calculation scripts, see the FFT Detector Calculation scripts reference section.

3.7 Sweep data tables

Sweep data table scripts allow exact specification of source values for Sweeps, rather than using the default linear or logarithmic steps specified on the Sweep Setup window.

For example, a simple frequency response may only require values at low and high frequencies, without needing many of the intermediate frequencies. In this case, a Sweep data table could be specified to define exactly the frequencies at which you wish to take Sweep results.

For a full list of all functions available to Sweep data table scripts, see the Sweep data table reference section.

3.8 Limit Table scripts

Limit Table scripts allow the user to specify Limit Lines for Traces, as a series of exact limit values rather than drawing them onto the Trace window.

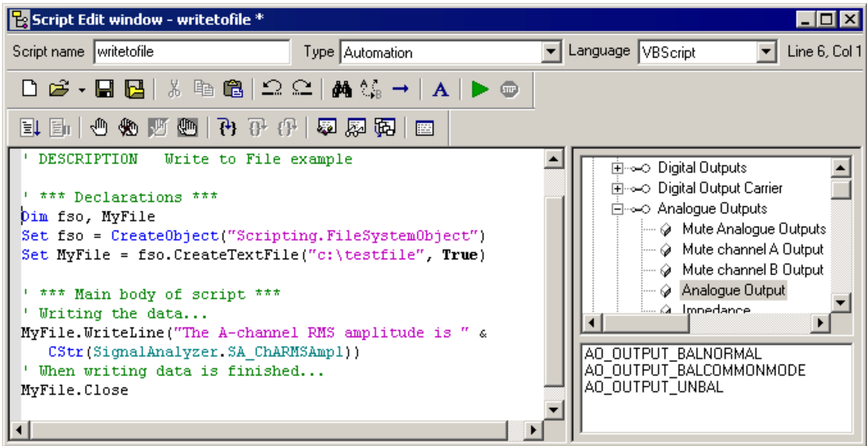
For a full list of all functions available to Limit Table scripts, see the Limit Table reference section.

Part 4

Script Edit window

4 Script Edit window

The Script Edit window can be used to edit and test any of the various scripts which can be used within the dScope. Since all dScope scripts are simple text files, they can also be edited with any other preferred text editor.



4.1 Script editor

The script editor allows for easy entry of scripts due to its colouring of code. Different parts of the script are displayed in different colours for quick reference:

"Pink"	denotes strings and characters.
' Green	denotes comments in the code (these have no effect but to explain what the code does).
Blue	indicates a VBScript keyword.
Grey	indicates a dScope method or property.
PALE_BLUE	indicates a dScope constant.

4.2 Buttons and Commands

The following buttons and commands appear on the Script Edit window:

Script toolbar

Script name	Allows entry of the name of the script. This will be used to create the file name for this script when it is saved.
Script type	Allows the user to select the type of script. This will alter the ' TYPE line of the script being edited to reflect the type chosen. (See Specifying the script type in a script). It will also determine which folder to save in by default, when "Save" or "Save As" is selected.
Language	Allows selection of the language that the script can be written in. This is selectable between **VBScript** and **JScript**.
Line details	These details show the current line and column that the cursor is on in the script.
	**New script** - Opens a brand new script in the editor. If a script is currently being edited, and has not been saved, you will firstly be asked if you wish to save the old script.

This option will create a new script, with a few lines of default code (for example, the script type, and a few lines of default code).

Open - Opens an already saved script in the script editor. The default folder will be the one defined by the current script type, as selected in the drop-list at the top of the window.
At the right of the button is a drop-list which can be used to select a script that has been recently opened in the Script Edit window.

Save - Saves the current script. If the script has already been saved, then the script will be saved to the same file name without further prompting.
If the script has not yet been saved, or the script name has been changed, this option will behave as the "Save As" option (see below).

Save As - Prompts the user for a filename before saving the script.

Cut - Cuts the current selection out of the script and copies it to the clipboard.

Copy - Copies the current selection from the script to the clipboard.

Paste - Pastes text from the clipboard into the script.

Undo - Reverts the script to the state it was in before the last action.

Redo - Reverts the script to the state it was in before the last "Undo" command was performed.

Change font - Allows you to select a font for the script.
This font is applied to all the text in the script editor.

Find - Brings up the "Find" dialogue box to allow you to search the script for a particular piece of text.

Replace - Brings up the "Replace" dialogue box to allow you to search the script for some text, and replace it with different text.

Run script - Runs the current script in the script editor. Note that this will not ask you whether you wish to save the script unless running the script tries to load a Configuration.

Stop script - Stops the currently running script.

Debug Toolbar

Go - Starts the script, in debugging mode.

Break - Breaks into the script at the point where it is currently running.

Toggle breakpoint - Adds a breakpoint at the line in the script containing the cursor. If the line already has a breakpoint, it will be removed.

Remove all breakpoints - Removes all breakpoints from the script.

Enable/disable breakpoint - Enables or disables the breakpoint at the line of the script containing the cursor. If this line does not have a breakpoint, this will be ignored.
An enabled breakpoint is show with a small circle in the left-hand margin (●). A disabled breakpoint is shown as a hollow circle (○).

Disable all breakpoints - Disables all breakpoints in the script.

Step into - Steps into the function or subroutine at the current cursor position.

Step out - Steps out of the current function or subroutine.

Step over - Steps over the function or subroutine at the current cursor position.

Show/hide Variables dialogue box - Shows/hides the Variables dialogue box.

Show/hide Watch dialogue box - Shows/hides the Watch dialogue box.

Show/hide Call stack dialogue box - Shows/hides the Call stack dialogue box.

Show/hide Breakpoints dialogue box - Shows/hides the Breakpoints dialogue box.

4.3 Tree of methods and properties

On the right hand side of the Script Edit window, you will find a list of all the properties and methods that are available for use in the script. This list is arranged under the same headings as the dScope menus, to make it easy to find a particular setting.

The list is presented as a "tree" structure, with branches that can be expanded or closed. Each item in the list is given a brief description rather than the more concise (and less understandable) method or property name.

To insert a method or property into the script, firstly ensure that the cursor in the script is positioned where you want the insertion to be made. Then, simply double-click the item in the tree, or drag it across to the script. Note that you do not have to release the mouse button in exactly the right place, since the item will be inserted at *the current cursor position*, and not necessarily the point at which the mouse is released.

Methods and parameter values

When a method is inserted in this way, the full name of the method, including parameter names and return value (if relevant), will be inserted into the script. In this way, you can see what parameters need to be specified to each method.

In a lot of cases, the parameters needed will be specific dScope constants, and only a certain range of constants are allowed. In these cases, the small window at the bottom right of the Script Edit window will contain a list of the appropriate variables. To insert these values into the script, simply double-click on the relevant name and it will be inserted into the script at the current cursor position.

4.4 Debugging a script

The debugging capabilities of the dScope Script Edit window allow you to step through a script, set breakpoints, and see the current state of variables while a script is running.

Before you can use the Script debugger built into dScope, you must have an external script debugger installed on your PC. The Microsoft Script Debugger can be downloaded from the Microsoft web site at http://www.microsoft.com/downloads/details.aspx?familyid=2f465be0-94fd-4569-b3c4-dffdf19ccd99&displaylang=en, or by searching for "Microsoft Script Debugger download" on the Microsoft web site www.microsoft.com.

Setting breakpoints

Before running a script, it can be useful to set breakpoints in the script. This will cause the script to stop, and break into the debugger, when the line of code containing the breakpoint is run.

To set a breakpoint, move the cursor to the line on which you want to set the breakpoint. Select the Toolbar icon to add a breakpoint. The breakpoint will be shown in the left margin of the Script Edit window as a filled circle (●). To remove this breakpoint, simply select the Toolbar icon again while the cursor is on the line containing the breakpoint, or use the Breakpoints dialogue box. To remove all breakpoints from the script, select the Toolbar icon.

Once a breakpoint is set, it can be enabled or disabled using the Toolbar icon. A disabled breakpoint is shown in the margin as a hollow circle (○). To disable all breakpoints in the script, select the Toolbar icon.

To run the script up to the first breakpoint, select the Toolbar icon. When the breakpoint is encountered, the script will stop and the current line of the script will be highlighted in yellow:

```
dCTVal = CTDetector.CTD_ChA
strMesg = "CT Value = " & dCTVal & vbCrLf & vbCrLf
If LastResultSettled() Then
 strMesg = strMesg & "Settled."
Else
 strMesg = strMesg & "!!! DID NOT SETTLE !!!"
End If
```

NB: **Running a script may cause the Script Edit window to close, for example if a Configuration is loaded by the script. If this happens, then the Script Edit window will re-open when the breakpoint is reached.**

Stepping through a script

Once the script is stopped (or when starting to run the script), the debugger can be used to step through each line of the script individually. If the script is a call to a function or subroutine, this can be stepped into or over as necessary.
To step through the script, use the Toolbar icon. This will step into any calls to functions or subroutines that are encountered. Note that this will even work when stepping into #Included scripts.
If you don't want to step into a function or subroutine, use the Toolbar icon to step over it. If you are in a function or subroutine, and simply want to step out of it back to the next line of script after the call to the function, then use the Toolbar icon to step out of the function.

Examining the values of variables

At any point while the script debugger is halted at a specific line of code, you can examine the contents of any variables in memory. The list of variables currently in scope can be viewed using the Variables dialogue box. Any variables in the script can be viewed, or expressions can be evaluated, using the Watch dialogue box. See the respective topics describing these dialogue boxes for further information.

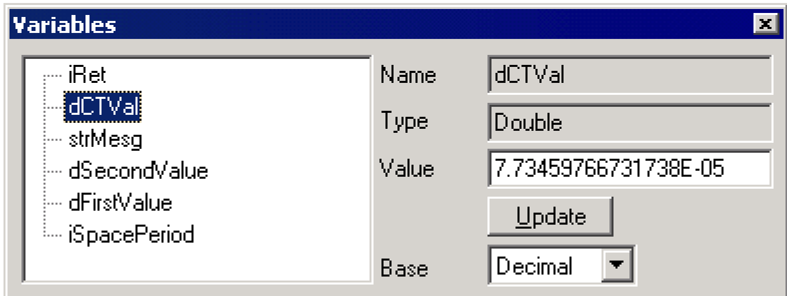
Note that if a script has been run in debugging mode (i.e. using the Toolbar icon, rather than the icon), you can break into the script at any point using the Toolbar icon. This will cause the current line to be highlighted in yellow, as shown above, and will enable the current state of variables to be viewed as described above.

4.4.1 The Variables dialogue box

The Variables dialogue box allows you to view and change the contents of variables that are within the current scope. To show or hide the Variables dialogue box, select the Toolbar icon from the

Script Edit window.

NB: **At present, if this dialogue box is shown while the script is within a function or subroutine, any variables with *global* scope are not shown. To see variables within global scope from within a function or subroutine, use the Watch dialogue box.**



The left-hand side of the dialogue box contains a list of the current variables that are within scope. When one of these variables is selected, its full details are shown on the right-hand side. If the variable is an array of values, it can be expanded by clicking on the "+" icon to the left of the variable name, to access individual elements of the array.

Name Shows the name of the selected variable.

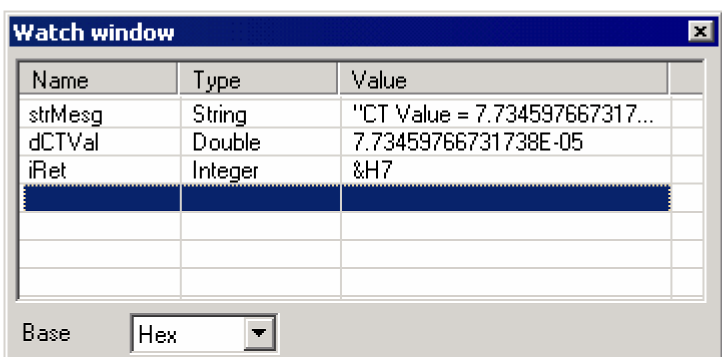
Type Shows the type of the selected variable. If the variable has been declared but not yet initialised, the debugger will not know the type of the variable and it will be listed as a "User-defined type".

Value Shows the current value of the variable. The value of a variable can be changed by editing this field, and selecting the [Update] button.

Base Allows the variable's value to be displayed in decimal or hexadecimal format, if applicable (byte, long and short integer values only).

4.4.2 The Watch dialogue box

The Watch dialogue box allows you to view and change the contents of variables that are within the current scope. It can also be used to evaluate expressions using variables in the script. To show or hide the Watch dialogue box, select the Toolbar icon from the Script Edit window.

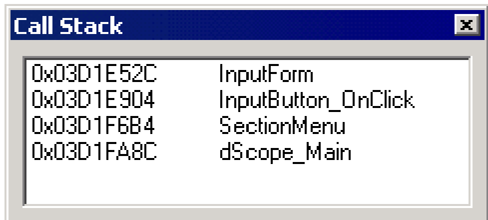


To find out the contents of a variable, or the result of an expression, simply click in the **Name** column and type in the expression or the name of the variable. This does not have to be case-sensitive. Pressing the <Enter> key will cause the expression entered to be evaluated (if the entry made is simply a variable name, then the current value of the variable will be shown).

The type of the variable or expression is shown in the middle column, and its value in the last column. The **Base** drop-list allows changing between decimal and hexadecimal display for relevant variable types (byte, long and short integer values only). The value of a variable can be changed by clicking in the last column and typing in a new value; when the <Enter> key is pressed, the variable's value will be changed to the value entered (providing the value is valid for that type of variable).

4.4.3 The Call stack dialogue box

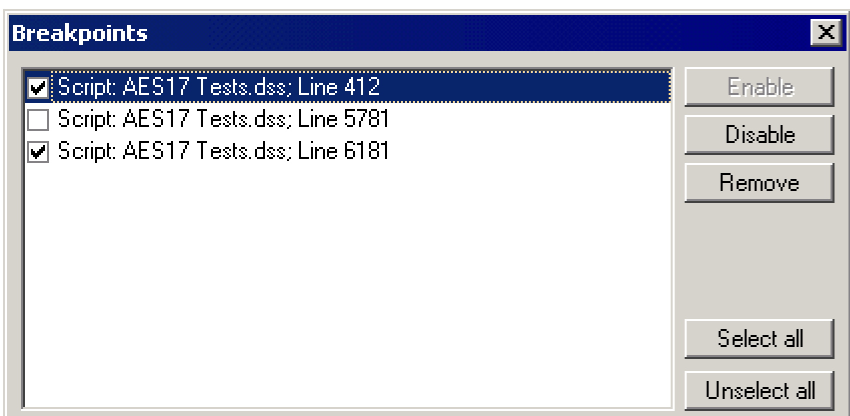
The Call stack dialogue box shows the stack of function calls that are currently active. To show or hide the Call stack dialogue box, select the Toolbar icon from the Script Edit window.



When a function is called, it is pushed onto the stack. When the function returns, it is popped off the stack. The Call stack dialogue box displays the currently executing function at the top of the stack, and older function calls below that.

4.4.4 The Breakpoints dialogue box

The Breakpoints dialogue box lists all the breakpoints currently set in a script. To show or hide the Call stack dialogue box, select the Toolbar icon from the Script Edit window.



The list of breakpoints shows a check box indicating whether the breakpoint is currently enabled or disabled, along with the script file and line that the breakpoint is on. A breakpoint can be enabled or disabled by checking or un-checking its associated check box, or by selecting the breakpoint in question and clicking the [Enable] or [Disable] button. A breakpoint can be removed by selecting it and clicking the [Remove] button.

Note that multiple breakpoints can be selected at the same time, and these operations performed on all selected breakpoints. To select or un-select all breakpoints in the list, use the [Select all] or [Unselect all] buttons.

Part 5

dScope scripting reference

5 dScope scripting reference

This section is a complete list of all the functions and settings available to control the dScope. The list is mainly arranged in the order that the functions appear in the tree on the right hand side of the Script Edit window (This also corresponds to the menu structure of the dScope).

Note that where reference is made to something being controllable from a script, the same applies to any application which is controlling the dScope externally. Where there is a difference in operation between the dScope scripting and external control, this will be stated.

Some dScope properties are read-only; this is specified in the text. Otherwise, the value can be written or read.

5.1 Data types and naming conventions

In the dScope software's automation interface, properties and parameters to functions can be any of a number of different data types. This section describes the different types of data available for use in the dScope.

In this scripting reference, variables and parameters have been specified using a convention loosely based on Hungarian notation. This means that the type of the variable (boolean, integer etc) is specified as a letter or group of letters at the start of the variable name, and it is easy to spot the variable type just by looking at its name.

Data types and their prefixes

Data type	Prefix	Meaning
boolean	b	Can contain the values **True** or **False**
byte / unsigned char	uc	A single-byte unsigned variable. This can have the values 0 to 255.
short	s	A signed short integer, 2 bytes (16 bits). Can have values from -32,768 to 32,767
long	l	A long integer, 4 bytes (32 bits). Can have values from -2,147,483,648 to 2,147,483,647.
double	d	A double-precision, floating point number occupying 8 bytes (64 bits). Can have values from -1.79769313486232e+308 to 1.7976931348623158e +308
string	str	A string, enclosed in double quotation marks. ("...") In the Type library, this data type is referred to as **BSTR** which stands for **B** inary **STR**ing.

For example:
`sReturnVal` would represent a return value of type "short".
`bOn` would represent a boolean; **True** if On, **False** if Off.

5.2 Inputs and Outputs

The Inputs and Outputs section of this reference contains details of all the properties and methods of the following areas of the dScope:

Digital Outputs
Digital Output Carrier

5.2.1 Digital Outputs

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The Digital Outputs section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"DigitalOutputs."**

Properties

Methods

There are no methods available to control the dScope Digital Outputs.

5.2.1.1 Properties

5.2.1.1.1 DO_Mute

Description

This property is used to mute or un-mute either or both channels of the Digital Outputs.

To mute either channel individually, use DO_MuteChA or DO_MuteChB.

Values

True	Mute both channels of the Digital Outputs.
False	Un-mute both channels of the Digital Outputs.

5.2.1.1.2 DO_MuteChA

Description

This property is used to mute or un-mute channel A of the Digital Outputs.

To mute both channels at the same time, use DO_Mute.

Values

True	Mute channel A of the Digital Outputs.
False	Un-mute channel A of the Digital Outputs.

5.2.1.1.3 DO_MuteChB

Description

This property is used to mute or un-mute channel B of the Digital Outputs.

To mute both channels at the same time, use DO_Mute.

Values

True	Mute channel B of the Digital Outputs.
False	Un-mute channel B of the Digital Outputs.

5.2.1.1.4 DO_Source

Description

This property allows configuration of the Digital Outputs source.

Values

DO_SOURCE_GEN	Specifies that the Digital Outputs should be sourced from the Signal Generator.
DO_SOURCE_LOOPTHROUGH	Specifies that the Digital Outputs should be looped through from the Digital Inputs. This is useful for monitoring a digital signal 'in-line', in which case the terminating impedance would normally be switched out in the Digital Inputs (see DI_InputsTerminated).
DO_SOURCE_CHANNELCHECK	Specifies that the Digital Outputs should be set to channel-check mode. This generates a pseudo-random bit sequence, of which each

bit can be predicted from the sequence of bits before it.
To specify how many bits to generate in the channel-check signal, use DO_ChannelCheck.
This mode is equivalent to the Channel Check mode on the DSA-1.

5.2.1.1.5 DO_ChannelCheck

Description

This property allows selection of the number of bits to generate the Channel Check signal for.

NB: **This property is ignored unless the Digital Outputs source (DO_Source) is set to DO_SOURCE_CHANNELCHECK.**

Values

DO_CHANNELCHECKBITS_16	Specifies that 16 bits of channel-check information should be generated.
DO_CHANNELCHECKBITS_20	Specifies that 20 bits of channel-check information should be generated.
DO_CHANNELCHECKBITS_24	Specifies that 24 bits of channel-check information should be generated.

5.2.1.1.6 DO_RefSyncSource

Description

This property allows selection of the Reference Sync source for the dScope's Digital Outputs.

Values

DO_REFSYNCSOURCE_INTERNAL	Specifies that the dScope's Reference Sync should its internal 96k clock.
DO_REFSYNCSOURCE_DIGINPUT	Specifies that the dScope's Reference Sync should be the Digital Input (on the connector specified using DI_Source).
DO_REFSYNCSOURCE_XLRAES11DARS	Specifies that the dScope's Reference Sync should be an AES11 Digital Audio Reference Signal on the XLR Ref Sync input.
DO_REFSYNCSOURCE_BNCAES11DARS	Specifies that the dScope's Reference Sync should be an AES11 Digital Audio Reference Signal on the BNC Ref Sync input.
DO_REFSYNCSOURCE_BNCWORDCLOCK	Specifies that the dScope's Reference Sync should be a word clock on the BNC Ref Sync input.
DO_REFSYNCSOURCE_BNCVIDEONTSC30	Specifies that the dScope's Reference Sync should be an NTSC(30) video signal on the BNC Ref Sync input.
DO_REFSYNCSOURCE_BNCVIDEO	Specifies that the dScope's Reference Sync should be a PAL/SECAM/NTSC(29.97) video signal on the BNC Ref Sync input.

5.2.1.1.7 DO_RefSyncInputsTerminated

Description

This property is used to specify whether the Reference Sync input is terminated.

Values

True	Reference Sync input is terminated.
False	Reference Sync input is not terminated.

5.2.1.1.8 DO_RefSyncFrameRate

Description

This **read-only** property represents the current frame rate of the incoming Reference Sync signal, rounded to the nearest standard rate, in Hz.

NB: **This property represents the nearest standard frame rate, for example 44100, 48000 or 96000. To obtain the exact frame rate, use DO_RefSyncActualFrameRate.**

Values

The Reference Sync frame rate is represented as a double-precision floating point value.

5.2.1.1.9 DO_RefSyncActualFrameRate

Description

This **read-only** property represents the current frame rate of the incoming Reference Sync signal, in Hz.

NB: **To obtain the nearest standard frame rate, use DO_RefSyncFrameRate.**

Values

The frame rate is represented as a double-precision floating point value.

5.2.1.1.10 DO_RefSyncFrameRateDeviation

Description

This **read-only** property shows the deviation in parts per million of the current Reference Sync input frame rate, from the nearest standard rate (32000, 44100, 48000, 88200 or 96000). If the Reference Sync is set to Video, it will be the deviation from the nearest relevant video frame rate (25.0, 29.97 or 30.0)

Values

The Reference Sync frame rate deviation is represented as a double-precision floating point value.

5.2.1.1.11 DO_FrameRate

Description

This property allows selection of the frame rate of the Digital Outputs.

NB: **If the Digital Outputs have been set to generate Split96 data (using DO_Split96), then the actual sample rate will be twice the specified frame rate.**
This feature does not affect sample rates over 96kHz.

Values

DO_FRAMERATE_32000	Specifies that the Digital Output frame rate should be 32kHz.
DO_FRAMERATE_44100	Specifies that the Digital Output frame rate should be 44.1kHz.
DO_FRAMERATE_48000	Specifies that the Digital Output frame rate should be 48kHz.
DO_FRAMERATE_88200	Specifies that the Digital Output frame rate should be 88.2kHz.
DO_FRAMERATE_96000	Specifies that the Digital Output frame rate should be 96kHz.
DO_FRAMERATE_176400	Specifies that the Digital Output frame rate should be 176.4kHz.
DO_FRAMERATE_192000	Specifies that the Digital Output frame rate should be 192kHz.
DO_FRAMERATE_FOLLOWSYNC	Specifies that the Digital Output frame rate should follow the Reference Sync frame rate. NB: If the Reference Sync is set to video or internal, this setting is ignored and the internal Reference Sync will be followed.

5.2.1.1.12 DO_AddFrameRateDeviation

Description

This property is used to specify whether to add the frame rate deviation (specified by DO_FrameRateDeviation) to the Digital Output frame rate.

Values

True	Add the specified deviation to the Digital Output frame rate.
False	Do not add the specified deviation to the Digital Output frame rate.

5.2.1.1.13 DO_FrameRateDeviation

Description

This property allows specification of the deviation to add to the Digital Output frame rate, in ppm.

This property is ignored unless the DO_AddFrameRateDeviation property is set to **True**.

Values

The frame rate deviation is represented as a short integer value. It can be any number between -1500 and 1500.

5.2.1.1.14 DO_ChAValidBit

Description

This property allows specification of the Valid bit for channel A of the Digital Output.

Values

DO_VBIT_VALID	Sets the Valid bit to valid.
DO_VBIT_INVALID	Sets the Valid bit to invalid.

5.2.1.1.15 DO_ChBValidBit

Description

This property allows specification of the Valid bit for channel B of the Digital Output.

Values

DO_VBIT_VALID	Sets the Valid bit to valid.
DO_VBIT_INVALID	Sets the Valid bit to invalid.

5.2.1.1.16 DO_UserBitCheck

Description

This property is used to specify whether to output a sequence of User bits that can be used to check a device for User bit transparency. This sequence can then be checked using the Digital Inputs properties DI_ChAUserBitError and DI_ChBUserBitError.

Values

True	Output the User bit check sequence.
False	Output User bits as zero (no check sequence).

5.2.1.1.17 DO_Wordlength

Description

This property allows specification of the wordlength of the Digital Output.

The number specified is the number of bits of signal left after truncation.

Values

The wordlength is represented as a short integer value. It can be any number between 8 and 24 bits.

5.2.1.1.18 DO_Dithering

Description

This property allows specification of the dither to apply to the Digital Outputs. This is applied after any DC offset and before wordlength truncation.

Values

DO_DITHERING_UNDITHERED	The Digital Outputs are not dithered.
DO_DITHERING_WHITE	TPDF white noise is applied to the Digital Outputs.

5.2.1.1.19 DO_DCOffset

Description

This property allows specification of the DC offset to apply to the Digital Outputs.

The value must be specified in the unit selected by DO_DCOffsetUnit.

Values

The DC offset is represented as a double-precision floating point value.

5.2.1.1.20 DO_DCOffsetUnit

Description

This property allows selection of the unit for the DC offset to be applied to the Digital Outputs (see DO_DCOffset).

Note that this may be applied as a positive or negative offset using DO_DCOffsetPolarity.

Values

UNIT_DBFS	Sets DC offset unit to dBFS.
UNIT_PERCENTFS	Sets DC offset unit to %FS (percentage of full scale).
UNIT_FFS	Sets DC offset unit to FFS (fraction of full scale).
UNIT_HEX	Sets DC offset unit to Hex.

5.2.1.1.21 DO_DCOffsetPolarity

Description

This property allows specification of the polarity of the DC offset to apply to the Digital Outputs.

Values

DO_DCOFFSETPOLARITY_POS	Specifies that the DC offset to be applied to the Digital Outputs should have positive polarity.
DO_DCOFFSETPOLARITY_NEG	Specifies that the DC offset to be applied to the Digital Outputs should have negative polarity.

5.2.1.1.22 DO_Split96

Description

This property is used to specify whether the reference output connector should be used as a second channel in Split96 mode.

This property is ignored if the Digital Outputs frame rate is greater than 96kHz.

Values

True	Use the reference output connector for a second channel in Split96 mode.
False	Use the reference output connector for the reference signal.

5.2.2 Digital Output Carrier

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The Digital Output Carrier section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"DigitalOutputCarrier."**

Properties

DOC_XLRAmpl
DOC_BNCAmpl
DOC_XLRRiseTime
DOC_BNCRiseTime
DOC_PhaseOffset
DOC_PhaseOffsetUnit
DOC_AddCMSignal
DOC_CMFreq
DOC_CMAmpl
DOC_AddJitter
DOC_JitterFunction
DOC_JitterFreq

DOC_JitterAmpl
DOC_JitterAmplUnit
DOC_AddDifferentialNoise
DOC_XLRNoiseAmpl
DOC_BNCNoiseAmpl

Methods

There are no methods available to control the dScope Digital Output Carrier.

5.2.2.1 Properties

5.2.2.1.1 DOC_XLRAmpl

Description

This property allows selection of the carrier amplitude on the XLR output, in Volts.

Values

The XLR carrier amplitude is represented as a double-precision floating point value. Any value from 120mV to 10.24V can be entered.

NB: **The XLR carrier amplitude can only be set in steps of 40mV; the dScope will round this property to the nearest 40mV.**

5.2.2.1.2 DOC_BNCAmpl

Description

This property allows selection of the carrier amplitude on the BNC output, in Volts.

Values

The BNC carrier amplitude is represented as a double-precision floating point value. Any value from 30mV to 2.56V can be entered.

NB: **The BNC carrier amplitude can only be set in steps of 10mV; the dScope will round this property to the nearest 10mV.**

5.2.2.1.3 DOC_XLRRiseTime

Description

This property allows selection of the rise time to apply to the XLR output, in ns.

Values

DOC_XLRRISETIME_5	Selects a rise time of 5 ns on the XLR carrier output.
DOC_XLRRISETIME_10	Selects a rise time of 10 ns on the XLR carrier output.
DOC_XLRRISETIME_20	Selects a rise time of 20 ns on the XLR carrier output.

DOC_XLRRISETIME_30	Selects a rise time of 30 ns on the XLR carrier output.
DOC_XLRRISETIME_40	Selects a rise time of 40 ns on the XLR carrier output.
DOC_XLRRISETIME_50	Selects a rise time of 50 ns on the XLR carrier output.
DOC_XLRRISETIME_60	Selects a rise time of 60 ns on the XLR carrier output.
DOC_XLRRISETIME_70	Selects a rise time of 70 ns on the XLR carrier output.
DOC_XLRRISETIME_80	Selects a rise time of 80 ns on the XLR carrier output.
DOC_XLRRISETIME_90	Selects a rise time of 90 ns on the XLR carrier output.
DOC_XLRRISETIME_100	Selects a rise time of 100 ns on the XLR carrier output.

5.2.2.1.4 DOC_BNCRiseTime

Description

This property allows selection of the rise time to apply to the BNC output, in ns.

Values

DOC_BNCRISETIME_5	Selects a rise time of 5 ns on the BNC carrier output.
DOC_BNCRISETIME_10	Selects a rise time of 10 ns on the BNC carrier output.
DOC_BNCRISETIME_20	Selects a rise time of 20 ns on the BNC carrier output.
DOC_BNCRISETIME_30	Selects a rise time of 30 ns on the BNC carrier output.
DOC_BNCRISETIME_40	Selects a rise time of 40 ns on the BNC carrier output.
DOC_BNCRISETIME_50	Selects a rise time of 50 ns on the BNC carrier output.
DOC_BNCRISETIME_60	Selects a rise time of 60 ns on the BNC carrier output.
DOC_BNCRISETIME_70	Selects a rise time of 70 ns on the BNC carrier output.
DOC_BNCRISETIME_80	Selects a rise time of 80 ns on the BNC carrier output.
DOC_BNCRISETIME_90	Selects a rise time of 90 ns on the BNC carrier output.
DOC_BNCRISETIME_100	Selects a rise time of 100 ns on the BNC carrier output.

5.2.2.1.5 DOC_PhaseOffset

Description

This property allows specification of the phase offset from the Reference Sync to apply to the carrier output, in the unit selected by DOC_PhaseOffsetUnit.

Values

The phase offset is represented as a double-precision floating point value.

5.2.2.1.6 DOC_PhaseOffsetUnit

Description

This property allows selection of the unit for specifying the phase offset of the carrier output (see DOC_PhaseOffset).

Values

UNIT_PHASE_PERCENT	Sets unit for carrier phase offset to be % of a frame.
UNIT_PHASE_DEGREES	Sets unit for carrier phase offset to be degrees (360 degrees is 1 frame).
UNIT_PHASE_UI	Sets unit for carrier phase offset to be UI.

5.2.2.1.7 DOC_AddCMSignal

Description

This property is used to select whether to add a common-mode signal to the carrier output.

The common-mode signal itself is specified using DOC_CMFreq and DOC_CMAmpl.

Values

True	Add the common-mode signal to the carrier output.
False	Do not add the common-mode signal to the carrier output.

5.2.2.1.8 DOC_CMFreq

Description

This property allows specification of the frequency of the common-mode signal to add to the carrier output, in Hz.

The common-mode signal added will be sinusoidal.

Values

The common-mode signal frequency is represented as a double-precision floating point value. It can be any value between 10Hz and 40kHz.

5.2.2.1.9 DOC_CMAmpl

Description

This property allows specification of the amplitude of the common-mode signal to add to the carrier output, in Volts.

Values

The common-mode signal amplitude is represented as a double-precision floating point value. It can be any value up to 20V (peak-to-peak).

5.2.2.1.10 DOC_AddJitter

Description

This property is used to select whether to add jitter to the carrier output.

The jitter signal itself is specified using DOC_JitterFunction, DOC_JitterFreq and DOC_JitterAmpl.

Values

True	Add the jitter to the carrier output.
False	Do not add the jitter to the carrier output.

5.2.2.1.11 DOC_JitterFunction

Description

This property allows selection of the jitter signal to apply to the carrier output.

Values

DOC_JITTERFUNCTION_SINE	Applies a sinusoidal jitter function to the carrier output, up to 1/2 UI.
DOC_JITTERFUNCTION_LFSINE	Applies a low-frequency sinusoidal jitter function to the carrier output. This signal is necessary to produce jitter of up to 20UI, to cover the jitter-tolerance requirements of AES3.
DOC_JITTERFUNCTION_AUDIONOISE	Applies audio-band white noise jitter to the carrier output.
DOC_JITTERFUNCTION_WIDEBANDNOISE	Applies wide-band (0 - 64 x frame rate) noise jitter to the carrier output.

5.2.2.1.12 DOC_JitterFreq

Description

This property allows specification of the frequency of the jitter signal to add to the carrier output, in Hz.

Values

The jitter signal frequency is represented as a double-precision floating point value.

If the jitter function (see DOC_JitterFunction) is set up to be sine (**DOC_JITTERFUNCTION_SINE)**, then the allowed range of values is 10Hz to 40kHz.

If the jitter function is set up to be low-frequency sine (**DOC_JITTERFUNCTION_LFSINE)**, then the allowed range of values is 10Hz to 10kHz.

5.2.2.1.13 DOC_JitterAmpl

Description

This property allows specification of the amplitude of the jitter signal to add to the carrier output, in the unit specified by DOC_JitterAmplUnit.

Values

The jitter signal amplitude is represented as a double-precision floating point value. If the jitter function (DOC_JitterFunction) is set to **DOC_JITTERFUNCTION_LFSINE**, any value from 0 to 20 UI can be entered. For all other functions, any value from 0 to 0.5UI can be entered.

5.2.2.1.14 DOC_JitterAmplUnit

Description

This property allows selection of the unit for the jitter signal to apply to the carrier output, as specified using DOC_JitterAmpl.

Values

UNIT_JITTER_NS	Sets unit for the jitter amplitude to ns.
UNIT_JITTER_UI	Sets unit for the jitter amplitude to UI.

5.2.2.1.15 DOC_AddDifferentialNoise

Description

This property is used to select whether to add differential noise to the carrier output.

The noise amplitude itself is specified using DOC_XLRNoiseAmpl or DOC_BNCNoiseAmpl.

Values

True	Add differential noise to the carrier output.
False	Do not add differential noise to the carrier output.

5.2.2.1.16 DOC_XLRNoiseAmpl

Description

This property allows specification of the amplitude of the differential noise to be added to the XLR carrier output, in Volts.

Values

The differential noise amplitude is represented as a double-precision floating point value. Any value from 0V to 2.56V (peak-to-peak) can be entered.

NB:

The XLR differential noise amplitude is tied to the BNC differential noise amplitude, and can only be set in steps of 10mV; the dScope will round this property to the nearest 10mV.

The XLR noise amplitude is always four times the BNC noise amplitude - when the XLR noise amplitude is changed, the BNC amplitude will be set to a quarter of the value.

5.2.2.1.17 DOC_BNCNoiseAmpl

Description

This property allows specification of the amplitude of the differential noise to be added to the BNC carrier output, in Volts.

Values

The differential noise amplitude is represented as a double-precision floating point value. Any value from 0V to 0.64V (peak-to-peak) can be entered.

NB:

The BNC differential noise amplitude is tied to the XLR differential noise amplitude, and can only be set in steps of 2.5mV; the dScope will round this property to the nearest 2.5mV.

The BNC noise amplitude is always a quarter of the XLR noise amplitude - when the BNC noise amplitude is changed, the XLR amplitude will be set to four times the value.

5.2.3 Analogue Outputs

The Analogue Outputs section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"AnalogueOutputs."**

Properties

AO_Mute
AO_MuteChA
AO_MuteChB
AO_Output
AO_Impedance
AO_Grounding

Methods

There are no methods available to control the dScope Analogue Outputs.

5.2.3.1 Properties

5.2.3.1.1 AO_Mute

Description

This property is used to mute or un-mute both channels of the Analogue Outputs.

To mute either channel individually, use AO_MuteChA or AO_MuteChB.

Values

True	Mute both channels of the Analogue Outputs.
False	Un-mute both channels of the Analogue Outputs.

5.2.3.1.2 AO_MuteChA

Description

This property is used to mute or un-mute channel A's Analogue Output.

To mute both channels at the same time, use AO_Mute.

Values

True	Mute channel A's Analogue Output.
False	Un-mute channel A's Analogue Output.

5.2.3.1.3 AO_MuteChB

Description

This property is used to mute or un-mute channel B's Analogue Output.

To mute both channels at the same time, use AO_Mute.

Values

True	Mute channel B's Analogue Output.
False	Un-mute channel B's Analogue Output.

5.2.3.1.4 AO_Output

Description

This property allows configuration of the Analogue Outputs for balanced, unbalanced or common-mode testing.

Values

AO_OUTPUT_BALNORMAL	Selects a normal balanced signal on the XLR and BNC Analogue Outputs. The inner of the BNC and pin 2 of the XLR are connected to 'hot', and the outer of the BNC and pin 3 of the XLR to 'cold'. Pin 1 of the XLR is connected to signal ground.
AO_OUTPUT_BALCOMMONMODE	Selects a common-mode signal on the Analogue Outputs, where the two balanced legs of the output carry the same signal with respect to ground rather than a differential signal.
AO_OUTPUT_UNBAL	Selects a normal balanced signal on the XLR and BNC Analogue Outputs. The inner of the BNC and pin 2 of the XLR are connected to 'hot', and the outer of the BNC and pin 3 of the XLR (as well as pin 1 of the XLR) are connected to signal ground.

NB: **The maximum amplitude capability of the Analogue Outputs is +28dBu (balanced) or +22dBu (unbalanced), into a minimum load of 150R.**

5.2.3.1.5 AO_Impedance

Description

This property allows selection of the output impedance for the Analogue Outputs.

Values

The allowed values for this property depend on the current output selection (see AO_Output).

If the outputs are unbalanced (**AO_OUTPUT_UNBAL** is selected), then the following impedances are valid:

AO_IMPEDANCE_UNBAL_MIN	Selects the minimum impedance for the unbalanced Analogue Outputs. This impedance is 25R, so using this value has exactly the same effect as using **AO_IMPEDANCE_UNBAL_25R**.
AO_IMPEDANCE_UNBAL_25R	Selects a 25R impedance for the unbalanced Analogue Outputs.
AO_IMPEDANCE_UNBAL_600R	Selects a 600R impedance for the unbalanced Analogue Outputs.

If the outputs are balanced (**AO_OUTPUT_BALNORMAL** or **AO_OUTPUT_BALCOMMONMODE** are selected), then the following impedances are valid:

AO_IMPEDANCE_BAL_MIN	Selects the minimum impedance for the balanced Analogue Outputs. This impedance is 50R, so using this value has exactly the same effect as using **AO_IMPEDANCE_UNBAL_50R**.
AO_IMPEDANCE_BAL_50R	Selects a 50R impedance for the balanced Analogue Outputs.
AO_IMPEDANCE_BAL_150R	Selects a 150R impedance for the balanced Analogue Outputs. Note that this value can only be set if the jumper is correctly set on the analogue board (see section on PCB jumper options for further details).
AO_IMPEDANCE_BAL_200R	Selects a 200R impedance for the balanced Analogue Outputs.

	Note that this value can only be set if the jumper is correctly set on the analogue board (see section on PCB jumper options for further details).
AO_IMPEDANCE_BAL_600R	Selects a 600R impedance for the balanced Analogue Outputs.
AO_IMPEDANCE_BAL_ASYMMETRIC	Selects an asymmetric impedance for the balanced Analogue Outputs (600R in one leg and 25R in the other). This mode is useful for testing the 'real world' common-mode rejection of an input circuit, since many input circuit designs rely on having a balanced source impedance to maintain good CMRR performance.

5.2.3.1.6 AO_Grounding

Description

This property allows selection of the grounding arrangement of the Analogue Outputs.

Values

AO_GROUNDING_FLOATING	Selects the Analogue Outputs to be floating.
AO_GROUNDING_CHASSIS	Selects the Analogue Outputs to be grounded (XLR pin 1 connected to the chassis).

5.2.4 Soundcard Outputs

The Soundcard Outputs section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"SoundcardOutputs."**

Properties

SO_UseWDM
SO_WDMSoundcard
SO_ASIOSoundcard
SO_Soundcard
SO_SampleRate
SO_Wordlength
SO_Mute
SO_MuteChA
SO_MuteChB
SO_Dithering
SO_BypassACM

Methods

SO_SetChannel
SO_GetChannel

5.2.4.1 Properties

5.2.4.1.1 SO_UseWDM

Description

This property is used to specify whether to use the specified WDM Soundcard (SO_WDMSoundcard) or the specified ASIO Soundcard (SO_ASIOSoundcard).

Values

True	Use the specified WDM Soundcard for output (default).
False	Use the specified ASIO Soundcard for output.

5.2.4.1.2 SO_WDMSoundcard

Description

This property allows selection of the WDM soundcard to use for output. This soundcard will be used for output if the SO_UseWDM property is set to **True**.

Values

Valid entries for this property will depend on the soundcards set up on the PC on which dScope is installed. Any string value with the name of an existing soundcard can be used, or "**- None -**" to prevent output on a soundcard. The list of available soundcards can be found in the Soundcard Outputs Dialogue Box.

5.2.4.1.3 SO_ASIOSoundcard

Description

This property allows selection of the ASIO soundcard to use for output. This soundcard will be used for output if the SO_UseWDM property is set to **False**.

Values

Valid entries for this property will depend on the soundcards set up on the PC on which dScope is installed. Any string value with the name of an existing soundcard can be used, or "**- None -**" to prevent output on a soundcard. The list of available soundcards can be found in the Soundcard Outputs Dialogue Box.

5.2.4.1.4 SO_Soundcard

Description

This property allows selection of the soundcard to use for output. This will change the ASIO or WDM Soundcard, depending on the value of the SO_UseWDM property.

Values

Valid entries for this property will depend on the soundcards set up on the PC on which dScope is installed. Any string value with the name of an existing soundcard can be used, or "**- None -**" to prevent output on a soundcard. The list of available soundcards can be found in the Soundcard Outputs Dialogue Box.

NB:

This property exists for legacy reasons; it was originally used to specify the WDM Soundcard, since ASIO soundcards were not originally available. For new scripts, use the more specific SO_WDMSoundcard or SO_ASIOSoundcard property, and then use SO_UseWDM to select which soundcard is used.

5.2.4.1.5 SO_SampleRate

Description

This property allows selection of the sample rate of the Soundcard Outputs.

Values

The valid range of values will depend on the currently selected soundcard (See SO_Soundcard). Any valid sample rate, in Hz, can be set (for example, **11025** or **48000**).

5.2.4.1.6 SO_Wordlength

Description

This property allows specification of the wordlength of the Soundcard Outputs.

Values

The wordlength is represented as a short integer value, and can be **8**, **16** or **24**. Whether this wordlength is valid will depend on the currently selected soundcard (see SO_Soundcard).

5.2.4.1.7 SO_BypassACM

Description

This property is used to specify whether the selected soundcard should bypass the Windows ACM drivers.

Values

True	Bypass the Windows ACM drivers (default).
False	Allow the Windows ACM drivers to perform conversions on the audio data.

5.2.4.1.8 SO_Mute

Description

This property is used to mute or un-mute both channels of the Soundcard Outputs.

To mute either channel individually, use SO_MuteChA or SO_MuteChB.

Values

True	Mute both channels of the Soundcard Outputs.
False	Un-mute both channels of the Soundcard Outputs.

5.2.4.1.9 SO_MuteChA

Description

This property is used to mute or un-mute channel A of the Soundcard Outputs.

To mute both channels at the same time, use SO_Mute.

Values

True	Mute channel A of the Soundcard Outputs.
False	Un-mute channel A of the Soundcard Outputs.

5.2.4.1.10 SO_MuteChB

Description

This property is used to mute or un-mute channel B of the Soundcard Outputs.

To mute both channels at the same time, use SO_Mute.

Values

True	Mute channel B of the Soundcard Outputs.
False	Un-mute channel B of the Soundcard Outputs.

5.2.4.1.11 SO_Dithering

Description

This property allows specification of the dither to apply to the Soundcard Outputs. This is applied before wordlength truncation.

Values

SO_DITHERING_UNDITHERED	The Soundcard Outputs are not dithered.
SO_DITHERING_WHITE	TPDF white noise is applied to the Soundcard Outputs.

5.2.4.2 Methods

5.2.4.2.1 SO_SetChannel

bRet = SO_SetChannel (sDeviceChannel, sGenChannel)

This method allows mapping of output channels on the selected soundcard (SO_Soundcard) to one of the channels of the dScope's Generator.

Parameters

sDeviceChannel	Which channel of the soundcard to map. This can be any number between 1 and the number of output channels on the soundcard. This can also be set to the value **SO_DEVCHANNEL_ALL**, which will cause all of the channels of the soundcard to be mapped to the specified Generator channel.
sGenChannel	The channel of the dScope's Generator to which the selected soundcard channel will be mapped. It can be one of the values listed under Channels, below.

Return value

This method returns **True** if it successfully found a Reading, or **False** otherwise.

Channels

SO_GENCHANNEL_MUTED	Mutes the specified channel(s) of the soundcard.
SO_GENCHANNEL_A	Maps the specified channel(s) of the soundcard to channel A of the dScope's Generator.
SO_GENCHANNEL_B	Maps the specified channel(s) of the soundcard to channel B of the dScope's Generator.
SO_GENCHANNEL_STEREO	If called with a device Channel of **SO_DEVCHANNEL_ALL**, maps all channels of the soundcard alternately to channels A and B of the dScope's Generator, alternately. If called with a specific device channel number, then an odd-numbered channel will be mapped to channel A of the Generator, while an even-numbered channel will be mapped to channel B.

5.2.4.2.2 SO_GetChannel

sChannel = SO_GetChannel (sDeviceChannel)

This method returns the channel of the dScope's Generator that is mapped to an output channel of the selected soundcard (SO_Soundcard).dScope's Generator.

Parameters

sDeviceChannel	Which channel of the soundcard to map. This can be any number between 1 and the number of output channels on the soundcard.

Return value

This method returns the channel of the dScope's Generator that is mapped to the specified channel of the device. It can be one of the following values:

SO_GENCHANNEL_MUTED	The specified soundcard channel is muted.
SO_GENCHANNEL_A	The specified soundcard channel is mapped to channel A of the dScope's Generator.
SO_GENCHANNEL_B	The specified soundcard channel is mapped to channel B of the dScope's Generator.

If the method fails (for example, because the device channel passed is invalid), then **-1** will be returned.

5.2.5 Digital Inputs

NB: This part of the dScope's scripting interface may not be available, depending on the dScope model number.

The Digital Inputs section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"DigitalInputs."**

Properties

Methods

There are no methods available to control the dScope Digital Inputs.

5.2.5.1 Properties

5.2.5.1.1 DI_Source

Description

This property allows selection of the Digital Input to be analyzed.

Values

DI_SOURCE_XLR	Selects the Digital Input to be analyzed to be the XLR input.
DI_SOURCE_BNC	Selects the Digital Input to be analyzed to be the BNC input.
DI_SOURCE_TOSLINK	Selects the Digital Input to be analyzed to be the TOSLINK (optical) input.
DI_SOURCE_GENXLR	Selects the Digital Input to be a direct-relay connection from the XLR Digital Output.
DI_SOURCE_GENBNC	Selects the Digital Input to be a direct-relay connection from the BNC Digital Output.

5.2.5.1.2 DI_InputsTerminated

Description

This property is used to set whether the Digital Inputs are terminated.

Values

True	Digital Inputs are terminated.
False	Digital Inputs are not terminated.

5.2.5.1.3 DI_Loopback

Description

This property allows selection of whether the Digital Inputs are looped back, i.e. the output of a channel is selected on the input of the other channel.

Values

DI_LOOPBACK_NONE	Selects no loopback on the Digital Inputs
DI_LOOPBACK_CHA	Selects the Digital Input on channel A to be channel A of the currently selected input source (see DI_Source), and the Digital Input on channel B to be routed from the channel A Digital Output. This enables analysis of the output and input on a single channel.
DI_LOOPBACK_CHB	Selects the Digital Input on channel B to be channel B of the currently selected input source (see DI_Source), and the Digital Input on channel A to be routed from the channel B Digital Output. This enables analysis of the output and input on a single channel.

5.2.5.1.4 DI_InputUnlocked

Description

This **read-only** property indicates whether a compliant digital input has been detected on the currently selected Digital Input.

Values

True	No valid input has been detected.
False	A valid input has been detected.

5.2.5.1.5 DI_BiphaseViolation

Description

This **read-only** property indicates whether a biphase violation has been detected on the currently selected Digital Input (i.e. required carrier transitions are missing).

Values

True	A biphase violation has been detected.
False	No biphase violation has been detected.

5.2.5.1.6 DI_BlockLengthError

Description

This **read-only** property indicates whether a block-length error has been detected on the currently selected Digital Input (i.e. the repeat rate of the Z-preamble is not 192 frames).

Values

True	A block-length error has been detected.
False	No block-length error has been detected.

5.2.5.1.7 DI_EyeNarrowing

Description

This **read-only** property indicates whether the eye-narrowing of the Digital Input Carrier is close to failure. This occurs when the cell-duration falls below 50% of the ideal value.

Values

True	The eye-narrowing has been detected as near failure.
False	No eye-narrowing near failure has been detected.

5.2.5.1.8 DI_Asynchronous

Description

This **read-only** property indicates whether the Digital Input Carrier is detected to be asynchronous with respect to the Digital Output Carrier. This will occur when the input is either outside +/- 90 degrees of the generated carrier phase, or the input is slipping with respect to the generated carrier.

Values

True	The carrier is asynchronous with respect to the Digital Output Carrier.
False	The carrier is not asynchronous with respect to the Digital Output Carrier.

5.2.5.1.9 DI_FrameRate

Description

This **read-only** property represents the current frame rate of the incoming Digital Input signal, rounded to the nearest standard rate, in Hz.

NB: **This property represents the nearest standard frame rate, for example 44100, 48000 or 96000. To obtain the exact frame rate, use DI_ActualFrameRate.**

Values

The frame rate is represented as a double-precision floating point value.

5.2.5.1.10 DI_ActualFrameRate

Description

This **read-only** property represents the current frame rate of the incoming Digital Input signal, in Hz.

NB: **To obtain the nearest standard frame rate, use DI_FrameRate.**

Values

The frame rate is represented as a double-precision floating point value.

5.2.5.1.11 DI_FrameRateDeviation

Description

This **read-only** property shows the deviation in parts per million of the current Digital Input frame rate, from the nearest standard rate (32000, 44100, 48000, 88200 or 96000).

Values

The frame rate deviation is represented as a double-precision floating point value.

5.2.5.1.12 DI_ChABitActivity

Description

This **read-only** property shows the bit-activity on channel A of the current Digital Input.

Values

The bit activity is represented as a long integer, with the bottom 24 bits of this value representing the current activity. A bit set to 1 indicates that the bit has changed in the last measurement period, a 0 indicates that it has not.

For example:
If this property is set to **2348**, this is equivalent to a Hexadecimal value of **0x092C**.
This in turn translates to a binary value of **0000 1001 0010 1100**
Which indicates that bits 2, 3, 5, 8, and 11 have changed in the last measurement period, but the other bits have not.

5.2.5.1.13 DI_ChABitState

Description

This **read-only** property shows the state of the bits on channel A of the current Digital Input.

Values

The bit state is represented as a long integer, with the bottom 24 bits of this value representing the current bit state.

NB: **This bit is only valid if the corresponding bit in the DI Bit Activity is set to zero (otherwise, the bit is changing and so its value is irrelevant).**

For example:
If this property is set to **1170**, this is equivalent to a Hexadecimal value of **0x0492**.
This in turn translates to a binary value of **0000 0100 1001 0010**
Which indicates that bits 1, 4, 7 and 10 are all set high (1's), but the other bits are all low (0's).

5.2.5.1.14 DI_ChBBitActivity

Description

This **read-only** property shows the bit-activity on channel B of the current Digital Input.

Values

The bit activity is represented as a long integer, with the bottom 24 bits of this value representing the current activity. A bit set to 1 indicates that the bit has changed in the last measurement period, a 0 indicates that it has not.

For example:
If this property is set to **2348**, this is equivalent to a Hexadecimal value of **0x092C**.
This in turn translates to a binary value of **0000 1001 0010 1100**
Which indicates that bits 2, 3, 5, 8, and 11 have changed in the last measurement period, but the other bits have not.

5.2.5.1.15 DI_ChBBitState

Description

This **read-only** property shows the state of the bits on channel B of the current Digital Input.

Values

The bit state is represented as a long integer, with the bottom 24 bits of this value representing the current bit state.

NB: **This bit is only valid if the corresponding bit in the DI Bit Activity is set to zero (otherwise, the bit is changing and so its value is irrelevant).**

For example:
If this property is set to **1170**, this is equivalent to a Hexadecimal value of **0x0492**.
This in turn translates to a binary value of **0000 0100 1001 0010**
Which indicates that bits 1, 4, 7 and 10 are all set high (1's), but the other bits are all low (0's).

5.2.5.1.16 DI_ChAValid

Description

This **read-only** property shows the state of the Valid bit on channel A of the current Digital Input.

Values

True	The Valid bit is valid (i.e. set to 0)
False	The Valid bit is invalid (i.e. set to 1).

5.2.5.1.17 DI_ChBValid

Description

This **read-only** property shows the state of the Valid bit on channel B of the current Digital Input.

Values

True	The Valid bit is valid (i.e. set to 0)
False	The Valid bit is invalid (i.e. set to 1).

5.2.5.1.18 DI_ChAUserBitActive

Description

This **read-only** property shows whether there is activity on the User bits of channel A of the current Digital Input.

Values

True	The User bits are active.
False	The User bits are inactive.

5.2.5.1.19 DI_ChBUserBitActive

Description

This **read-only** property shows whether there is activity on the User bits of channel B of the current Digital Input.

Values

True	The User bits are active.
False	The User bits are inactive.

5.2.5.1.20 DI_ChAUserBitError

Description

This **read-only** property shows whether a User bit transparency check (as set using DO_UserBitCheck) has succeeded on channel A.

Values

True	The User bit transparency check has succeeded.
False	The User bit transparency check has failed.

5.2.5.1.21 DI_ChBUserBitError

Description

This **read-only** property shows whether a User bit transparency check (as set using DO_UserBitCheck) has succeeded on channel B.

Values

True	The User bit transparency check has succeeded.
False	The User bit transparency check has failed.

5.2.5.1.22 DI_MaskBits

Description

This property allows masking of the incoming audio word to a fixed length prior to analysis.

This can be useful in simulating an input with limited wordlength, or to check the dither of a non-truncated output.

Values

The mask bits are specified as a short integer, indicating the number of bits to truncate the input signal to. Any number between 8 and 24 bits can be specified.

5.2.5.1.23 DI_Split96

Description

This property can be used to specify that the Digital Input should be treated as Split96, i.e. a single frame contains details of two successive samples on the same channel, rather than a sample from each channel. This means that the A and B channels are assumed to be shared by a single channel whose sample rate is twice the indicated frame rate.

Split96 mode at the supported frame rates of 32kHz to 96kHz therefore corresponds to sampling rates of 64kHz to 192kHz.

Values

True	Treat the incoming digital carrier as a Split96 signal.
False	Treat the incoming digital carrier as a standard two-channel AES3 signal.

5.2.5.1.24 DI_UseRefInputForSplit96

Description

This property is used to specify whether the reference input connector should be treated as a second channel in Split96 mode.

This property is ignored if the Digital Inputs frame rate is greater than 96kHz.

Values

True	Use the reference input connector for a second channel in Split96 mode.
False	Use the reference input connector for the reference signal.

5.2.5.1.25 DI_ChannelCheck

Description

This property allows you to specify that you wish to enable Channel Check on the Digital Inputs.

This mode assumes that the signal on the current Digital Input is a certain pseudo-random sequence (as generated by the dScope Digital Outputs in the equivalent output mode, or a DSA-1). See DO_Source and DO_ChannelCheckBits to find out how to set up the dScope to generate this signal.

When in this mode, the DI_ChAChannelCheckFailed and DI_ChBChannelCheckFailed values can be read to find out whether there have been any bit errors on the specified channel. Note that this property is "sticky", i.e. once a failure has been detected, the Channel Check will indicate failure even if the input is currently correct.

Values

DI_CHANNELCHECK_OFF	Sets the Digital Input to normal analysis, i.e. no Channel Check is done.
DI_CHANNELCHECK_16BITS	Sets Channel Check for only the top 16 bits of the Digital Input signal.
DI_CHANNELCHECK_20BITS	Sets Channel Check for only the top 20 bits of the Digital Input signal.
DI_CHANNELCHECK_24BITS	Sets Channel Check for the entire 24 bits of the Digital Input signal.

NB: **Flagging of failure is dependent on the current Channel Status, as well as the current input. See DI_ChAChannelCheckFailed or DI_ChAChannelCheckFailed for details.**

5.2.5.1.26 DI_ChAChannelCheckFailed

Description

This property shows whether the current signal on channel A of the Digital Input is a valid Channel Check pseudo-random bit sequence (as generated by the dScope Digital Outputs in the equivalent output mode, or a DSA-1 in Channel Check mode). Note that this property is "sticky", i.e. once a failure has been detected, the Channel Check will indicate failure even if the input is currently correct. This property can be written as **False** to reset the flag before a subsequent check.

This will *always* be **False** if the Digital Inputs are not set up to be in Channel Check mode (See DI_ChannelCheck).

Values

True	Indicates that the Channel Check has found errors on this channel.
False	Indicates that the Channel Check has found no errors on this channel, i.e. the pseudo-random bit sequence is as expected.

5.2.5.1.27 DI_ChBChannelCheckFailed

Description

This property shows whether the current signal on channel B of the Digital Input is a valid Channel Check pseudo-random bit sequence (as generated by the dScope Digital Outputs in the equivalent output mode, or a DSA-1 in Channel Check mode). Note that this property is "sticky", i.e. once a failure has been detected, the Channel Check will indicate failure even if the input is currently correct. This property can be written as **False** to reset the flag before a subsequent check.

This will *always* be **False** if the Digital Inputs are not set up to be in Channel Check mode (See DI_ChannelCheck).

Values

True	Indicates that the Channel Check has found errors on this channel.
False	Indicates that the Channel Check has found no errors on this channel, i.e. the pseudo-random bit sequence is as expected.

5.2.6 Digital Input Carrier

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The Digital Input Carrier section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"DigitalInputCarrier."**

Properties

DIC_CarrierAmplMode
DIC_CarrierAmpl

DIC_JitterMode
DIC_Jitter
DIC_JitterUnit
DIC_CarrierPhase
DIC_CarrierPhaseUnit

Methods

There are no methods available to control the dScope Digital Input Carrier.

5.2.6.1 Properties

5.2.6.1.1 DIC_CarrierAmplMode

Description

This property allows selection of the carrier amplitude measurement mode.

Values

DIC_CARRIERAMPLMODE_DIFFERENTIAL	Selects the carrier amplitude measurement to be differential amplitude of the carrier.
DIC_CARRIERAMPLMODE_COMMONMODE	Selects the carrier amplitude measurement to be the common-mode amplitude (XLR only).
DIC_CARRIERAMPLMODE_AUDIOBAND	Selects the carrier amplitude measurement to be audio-band (band-limited to 20kHz to detect accidental routing/mixing of an analogue source or mains interference).

5.2.6.1.2 DIC_CarrierAmpl

Description

This **read-only** property represents the current carrier amplitude, in Volts.

This value depends on the current measurement mode, as set using DIC_CarrierAmplMode.

Values

The carrier amplitude is represented as a double-precision floating point value.

5.2.6.1.3 DIC_JitterMode

Description

This property allows selection of the jitter amplitude measurement mode. This specifies which part of the degradation of the incoming carrier is accessible using the DIC_Jitter property.

Values

DIC_JITTERMODE_FSJITTER	Sets the jitter amplitude measured to be the

	amplitude of the fs jitter (attributable to sample rate jitter at the source)
DIC_JITTERMODE_DATAJITTER	Sets the jitter amplitude measured to be the amplitude of data jitter (attributable to source and cabling jitter).
DIC_JITTERMODE_EYENARROWING_0X	Sets the jitter amplitude measured to be the reduction in duration of the eye-pattern from the "ideal" (1UI), at the zero-crossing.
DIC_JITTERMODE_EYENARROWING_200MV	Sets the jitter amplitude measured to be the reduction in duration of the eye-pattern from the "ideal" (1UI), with a 200mV threshold (as specified in the AES3 standard).
DIC_JITTERMODE_CARRIERDISPLAY	This cannot be set by the user; it is set automatically by the dScope software. Jitter measurement cannot be carried out if the carrier display is currently open, so the jitter measurement mode in the dScope will be set to this value.

5.2.6.1.4 DIC_Jitter

Description

This **read-only** property represents the current jitter amplitude, in the unit specified by DIC_JitterUnit.

This value depends on the current jitter measurement mode, as set using DIC_JitterMode.

Values

The jitter amplitude is represented as a double-precision floating point value.

NB: **This amplitude is always a peak-to-peak measurement.**

5.2.6.1.5 DIC_JitterUnit

Description

This property allows selection of the jitter amplitude unit. This specifies which unit the jitter measurement property (DIC_Jitter) will be returned in.

The measurement being performed is specified by setting the jitter measurement mode using DIC_JitterMode.

Values

UNIT_JITTER_NS	Specifies that the jitter amplitude should be measured in ns.
UNIT_JITTER_UI	Specifies that the jitter amplitude should be measured in UI.

NB: **Both these units are peak-to-peak units.**

5.2.6.1.6 DIC_CarrierPhase

Description

This **read-only** property represents the current Digital Input Carrier phase, with respect to the Reference Sync, in the unit specified by DIC_CarrierPhaseUnit

Values

The carrier phase is represented as a double-precision floating point value; its sign shows the polarity of the phase.

5.2.6.1.7 DIC_CarrierPhaseUnit

Description

This property allows selection of the carrier phase unit. This specifies which unit the carrier phase property (DIC_CarrierPhase) will be returned in.

Values

UNIT_PHASE_DEGREES	Specifies that the carrier phase should be measured in degrees.
UNIT_PHASE_PERCENT	Specifies that the carrier phase should be measured in % of a frame.
UNIT_PHASE_UI	Specifies that the carrier phase should be measured in UI.

5.2.7 Carrier Display

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The Carrier Display section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"CarrierDisplay."**

Properties

CD_XUnit
CD_ShowAESDetails
CD_StepSize
CD_StepTime
CD_Interpolate
CD_IncreaseRes

Methods

CD_Restart
CD_GetXRange
CD_SetXRange
CD_GetYRange
CD_SetYRange

5.2.7.1 Drawing of the Carrier Display

The drawing parameters that can be specified on the Carrier Display allow a high degree of flexibility in the way the Carrier Display is built up.

Specifying that data should be interpolated (CD_Interpolate) determines whether the graph drawing will interpolate between successive points.

The Gate Time (CD_GateTime) determines how long the dScope will scan on each point - this is important in determining the certainty with which infrequent carrier transitions will be detected: for example the X-preamble is replaced by a single Z-preamble every 192 frames. This equates to every 4ms at a frame rate of 48kHz. If the Gate Time is set below 4ms, then detection of Z-preamble activity will be unreliable. Setting a short Gate Time speeds up the scan, whereas a long Gate Time improves detection of infrequent events.

The Resolution (CD_Resolution) can be set in arbitrary units between 1 and 256. The smaller the number, the finer the resolution. A setting of 1 corresponds to a time resolution of about 300ps. Low settings produce a very finely detailed graph, but very slowly. High settings are faster but reduce the level of detail.

Increasing resolution with each pass (CD_IncreaseRes) gives a useful compromise, where each successive pass is made with an increased time resolution. Thus it is possible to quickly see the shape of the carrier in the area of interest, and adjust if necessary, before waiting for the required degree of resolution to be attained.

5.2.7.2 Properties

5.2.7.2.1 CD_XUnit

Description

This property allows selection of the unit for the X axis of the Carrier Display.

This specifies which unit the methods CD_SetXRange and CD_GetXRange will use.

Values

UNIT_JITTER_NS	Specifies that the Carrier Display X axis should be shown in ns.
UNIT_JITTER_UI	Specifies that the Carrier Display X axis should be shown in UI.

5.2.7.2.2 CD_ShowAESDetails

Description

This property determines whether to show the AES specification for minimum allowed eye-closure on the Carrier Display.

The minimum specification indicates that the eye-size must be at least 0.5UI horizontally, and 200mV vertically.

Values

True	Show the AES specification rectangle on the Carrier Display.
False	Remove the AES specification rectangle from the Carrier Display.

5.2.7.2.3 CD_Interpolate

Description

This property determines whether to interpolate the drawing of the Carrier Display.

See Drawing of the Carrier Display for further details.

Values

True	Specifies that drawing should be interpolated.
False	Specifies that drawing should not be interpolated. This may cause the Carrier Display to appear quite blocky.

5.2.7.2.4 CD_GateTime

Description

This property allows specification of the Gate Time for the Carrier Display, i.e. how long it spends obtaining each data point.

See Drawing of the Carrier Display for further details.

Values

Any number between 1 and 256 can be entered - the higher the number, the longer it will take to obtain the data but the more accurate the graph detail will be.

5.2.7.2.5 CD_IncreaseRes

Description

This property determines whether to increase resolution for each pass of drawing the Carrier Display.

See Drawing of the Carrier Display for further details.

Values

True	Specifies that resolution should be increased for every pass through the data.
False	Specifies that resolution should not be increased for every pass through the data.

5.2.7.2.6 CD_Resolution

Description

This property allows specification of the resolution for the Carrier Display, i.e. how many points it will take on one Sweep.

See Drawing of the Carrier Display for further details.

Values

Any number between 1 and 256 can be entered - the higher the number, the higher the resolution, i.e. the more data points it will obtain on a pass through the display.

5.2.7.3 Methods

5.2.7.3.1 CD_Restart

CD_Restart ()

This method restarts the Carrier Display data acquisition from the current point at the left of the X Axis.

NB: **The Carrier Display resolution (See CD_Resolution) will be reset to its initial value.**

Parameters

This method has no parameters.

Return value

This method has no return value.

5.2.7.3.2 CD_GetXRange

CD_GetXRange (dMinValue, dMaxValue)

This method gets the current X range, in the current unit (as specified using the CD_XUnit property).

Parameters

dMinValue	After this method is called, this parameter will hold the current minimum X value (in the unit specified by CD_XUnit).
dMaxValue	After this method is called, this parameter will hold the current maximum X value (in the unit specified by CD_XUnit).

Return value

This method has no return value.

5.2.7.3.3 CD_SetXRange

CD_SetXRange (dMinValue, dMaxValue)

This method sets the current X range, in the current unit (as specified using the CD_XUnit property).

Parameters

dMinValue	The value to use as the minimum value for the X axis (in the unit specified by CD_XUnit).
dMaxValue	The value to use as the maximum value for the X axis (in the unit specified by CD_XUnit).

Return value

This method has no return value.

5.2.7.3.4 CD_GetYRange

CD_GetYRange (dMinValue, dMaxValue)

This method gets the current Y range, in Volts.

Parameters

dMinValue	After this method is called, this parameter will hold the current minimum Y value (in Volts).
dMaxValue	After this method is called, this parameter will hold the current maximum Y value (in Volts).

Return value

This method has no return value.

5.2.7.3.5 CD_SetYRange

CD_SetYRange (dMinValue, dMaxValue)

This method sets the current Y range, in Volts.

Parameters

dMinValue	The value to use as the minimum Y axis value (in Volts).
dMaxValue	The value to use as the maximum Y axis value (in Volts).

Return value

This method has no return value.

5.2.8 Analogue Inputs

The Analogue Inputs section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"AnalogueInputs."**

Properties

AI_Source
AI_SamplingRate
AI_Impedance
AI_AutoRange
AI_Range
AI_RangeChA
AI_RangeChB
AI_RangeOverriddenChA
AI_RangeOverriddenChB
AI_RangesTied
AI_RangeStepSize

Methods

There are no methods available to control the dScope Analogue Inputs.

5.2.8.1 Properties

5.2.8.1.1 AI_Source

Description

This property allows selection of the Analogue Input to be analyzed.

Values

AI_SOURCE_BAL_UNBAL	Selects the Analogue Input to be analyzed to be the normal balanced or unbalanced input (BNC or XLR connectors). The BNC and XLR connectors are wired in parallel, so either can be used without separate selection. (See note below).
AI_SOURCE_BALANCED	Selects the Analogue Input to be analyzed to be the balanced input (XLR connectors).
AI_SOURCE_UNBALANCED	Selects the Analogue Input to be analyzed to be the unbalanced input (RCA connectors).
AI_SOURCE_JITTERDEMOD	Selects the demodulated fs jitter signal (from the current Digital

_FS	Inputs) to be routed to the analogue analyzer.
AI_SOURCE_JITTERDEMOD_DATA	Selects the demodulated data jitter signal (from the current Digital Inputs) to be routed to the analogue analyzer.
AI_SOURCE_GENOUTPUT	Selects the Analogue Inputs to be internally fed from the Analogue Outputs.
AI_SOURCE_CHA	Selects the Analogue Input on channel A to be the current balanced/ unbalanced input, and the Analogue Input on channel B to be routed from the channel A Analogue Output. This enables analysis of the output and input on a single channel.
AI_SOURCE_CHB	Selects the Analogue Input on channel B to be the current balanced/ unbalanced input, and the Analogue Input on channel A to be routed from the channel B Analogue Output. This enables analysis of the output and input on a single channel.

NB: **If AI_SOURCE_BAL_UNBAL is selected as the Analogue Input, take care to ensure that the unused connector is not connected, since this will adversely affect Results.**

5.2.8.1.2 AI_SampleRate

Description

This property allows selection of the Analogue Inputs sample rate.

NB: **Selection of the sample rate on the Analogue Inputs will also change the sample rate of the Analogue Outputs.**

Values

AI_SAMPLERATE_48	Sets the sample rate of the Analogue Inputs and Outputs to 48kHz.
AI_SAMPLERATE_96	Sets the sample rate of the Analogue Inputs and Outputs to 96kHz.
AI_SAMPLERATE_192	Sets the sample rate of the Analogue Inputs and Outputs to 192kHz.

NB: **Your dScope III hardware must have the 192kHz analogue capability to set the sample rate to 192kHz. For information about upgrading your hardware to 192kHz, please contact Prism Sound.**

5.2.8.1.3 AI_Impedance

Description

This property allows selection of the differential input impedance of the Analogue Inputs.

Values

AI_IMPEDANCE_100K	Selects the Analogue Inputs impedance to 100kR.
AI_IMPEDANCE_150R	Selects the Analogue Inputs impedance to 150R. Note that this value can only be set if the jumper is correctly set on the analogue board (see section on PCB jumper options for further details).

AI_IMPEDANCE_200R	Selects the Analogue Input impedance to 200R. Note that this value can only be set if the jumper is correctly set on the analogue board (see section on PCB jumper options for further details).
AI_IMPEDANCE_600R	Selects the Analogue Input impedance to 600R.

NB: **Note that the dScope software may defeat an input impedance selection if the detected level is sufficient to damage the impedance-setting resistor.**

5.2.8.1.4 AI_AutoRange

Description

This property is used to enable auto-ranging of the Analogue Inputs.

Under normal operation, auto-ranging should be used since it will improve the performance of the analogue analyzer. However, fixed ranging is useful if awkward waveforms or very low frequencies are being analyzed which might cause continuous hunting by the auto-range algorithm.

Values

True	Allow Analogue Inputs to auto-range.
False	Fix the range of the Analogue Inputs (at the range specified using AI_Range).

NB: **If a fixed ranged is entered and the input amplitude exceeds it, auto-ranging takes over until the overload is removed.**

5.2.8.1.5 AI_Range

Description

This property is used to set the fixed range of both channels of the Analogue Input converters, in dBu.

This value will only be used if AI_AutoRange has been set to **False**.

NB: **Setting the input range using this method will cause the AI_RangesTied property to be set to True.**

Values

Any number between **-18dBu** and **46dBu** can be specified. The software internally uses the nearest 2dBu step.

5.2.8.1.6 AI_RangeChA

Description

This property is used to set the fixed range of channel A of the Analogue Input converters, in dBu.

This value will only be used if AI_AutoRange has been set to **False**.

NB: **If the AI_RangesTied is set to True, the range will also be set for channel B.**

Values

Any number between **-18dBu** and **46dBu** can be specified. The software internally uses the nearest 2dBu step.

5.2.8.1.7 AI_RangeChB

Description

This property is used to set the fixed range of channel B of the Analogue Input converters, in dBu.

This value will only be used if AI_AutoRange has been set to **False**.

NB: **If the AI_RangesTied is set to True, the range will also be set for channel A.**

Values

Any number between **-18dBu** and **46dBu** can be specified. The software internally uses the nearest 2dBu step.

5.2.8.1.8 AI_RangeOverriddenChA

Description

This **read-only** property is set whenever a manually set input range for channel A is overridden by the auto-ranging, to protect the analogue converters.

Values

This property will return **True** if the manual range is being overridden by auto-ranging, or **False** if the input signal is within the current range (or if auto-ranging is turned on). Note that this value is not sticky, so will only return True while the range is overridden, and will revert to False when it is no longer overridden.

5.2.8.1.9 AI_RangeOverriddenChB

Description

This **read-only** property is set whenever a manually set input range for channel B is overridden by the auto-ranging, to protect the analogue converters.

Values

This property will return **True** if the manual range is being overridden by auto-ranging, or **False** if the input signal is within the current range (or if auto-ranging is turned on). Note that this value is not sticky, so will only return True while the range is overridden, and will revert to False when it is no longer overridden.

5.2.8.1.10 AI_RangesTied

Description

This property is used to tie together the manual range of the Analogue Input converters for both channels.

Values

True	Tie together the Analogue Input range of channel A and B (default).
False	Do not tie together the Analogue Input range of channel A and B.

NB: **If this property is set to True, while the ranges on each channel are different, then the range for both channels will be set to the value currently specified for channel A.**

5.2.8.1.11 AI_RangeStepSize

Description

This property allows selection of the step size to use for settings the range of the Analogue Inputs.

Values

AI_RANGESTEPSIZE_FINE	Selects a fine step size (2dB) for the Analogue Inputs auto-ranging.
AI_RANGESTEPSIZE_MEDIUM	Selects a medium step size (6dB) for the Analogue Inputs auto-ranging.
AI_RANGESTEPSIZE_COARSE	Selects a coarse step size (20dB) for the Analogue Inputs auto-ranging.

5.2.9 Soundcard Inputs

The Soundcard Inputs section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"SoundcardInputs."**

Properties

SI_UseWDM
SI_WDMSoundcard
SI_ASIOSoundcard
SI_Soundcard
SI_SampleRate
SI_Wordlength
SI_BypassACM
SI_ChannelA
SI_ChannelB
SI_NoInput

Methods

There are no methods available to control the dScope Soundcard Inputs.

5.2.9.1 SI_UseWDM

Description

This property is used to specify whether to use the specified WDM Soundcard (SI_WDMSoundcard) or the specified ASIO Soundcard (SI_ASIOSoundcard).

NB: **The selected soundcard will only be used for analysis if the Signal Analyzer's input source (SA_Source) is set to 'Soundcard' (SA_SOUNDCARD).**

Values

True	Use the specified WDM Soundcard for input (default).
False	Use the specified ASIO Soundcard for input.

5.2.9.2 SI_WDMSoundcard

Description

This property allows selection of the WDM soundcard to use for input. This soundcard will be used for input if the SI_UseWDM property is set to **True**.

NB: **The selected soundcard will only be used for analysis if the Signal Analyzer's input source (SA_Source) is set to 'Soundcard' (SA_SOUNDCARD).**

Values

Valid entries for this property will depend on the soundcards set up on the PC on which dScope is installed. Any string value with the name of an existing soundcard can be used, or "**- None -**" to disable analysis of input from a soundcard. The list of available soundcards can be found in the Soundcard Inputs Dialogue Box.

5.2.9.3 SI_ASIOSoundcard

Description

This property allows selection of the ASIO soundcard to use for input. This soundcard will be used for input if the SI_UseWDM property is set to **False**.

NB: **The selected soundcard will only be used for analysis if the Signal Analyzer's input source (SA_Source) is set to 'Soundcard' (SA_SOUNDCARD).**

Values

Valid entries for this property will depend on the soundcards set up on the PC on which dScope is installed. Any string value with the name of an existing soundcard can be used, or "**- None -**" to disable analysis of input from a soundcard. The list of available soundcards can be found in the Soundcard Inputs Dialogue Box.

5.2.9.4 SI_Soundcard

Description

This property allows selection of the soundcard to use for input. This will change the ASIO or WDM Soundcard, depending on the value of the SI_UseWDM property.

NB: **The selected soundcard will only be used for analysis if the Signal Analyzer's input source (SA_Source) is set to 'Soundcard' (SA_SOUNDCARD).**

Values

Valid entries for this property will depend on the soundcards set up on the PC on which dScope is installed. Any string value with the name of an existing soundcard can be used, or "**- None -**" to disable analysis of input from a soundcard. The list of available soundcards can be found in the Soundcard Inputs Dialogue Box.

NB: **This property exists for legacy reasons; it was originally used to specify the WDM Soundcard, since ASIO soundcards were not originally available. For new scripts, use the more specific SO_WDMSoundcard or SO_ASIOSoundcard property, and then use SO_UseWDM to select which soundcard is used.**

5.2.9.5 SI_SampleRate

Description

This property allows selection of the sample rate of the Soundcard Inputs.

Values

The valid range of values will depend on the currently selected soundcard (See SI_Soundcard). Any valid sample rate, in Hz, can be set (for example, **11025** or **48000**).

5.2.9.6 SI_Wordlength

Description

This property allows specification of the wordlength of the Soundcard Inputs.

Values

The wordlength is represented as a short integer value, and can be **8**, **16** or **24**. Whether this wordlength is valid will depend on the currently selected soundcard (see SI_Soundcard).

5.2.9.7 SI_BypassACM

Description

This property is used to specify whether the selected soundcard should bypass the Windows ACM drivers.

Values

True	Bypass the Windows ACM drivers (default).
False	Allow the Windows ACM drivers to perform conversions on the audio data.

5.2.9.8 SI_ChannelA

Description

This property allows selection of the soundcard channel to be mapped to channel A of the dScope's Analyzer.

Values

The valid range of values will depend on the currently selected soundcard (See SI_Soundcard). Any valid channel number, from 1 to the number of input channels on the soundcard, can be used.

5.2.9.9 SI_ChannelB

Description

This property allows selection of the soundcard channel to be mapped to channel B of the dScope's Analyzer.

Values

The valid range of values will depend on the currently selected soundcard (See SI_Soundcard). Any valid channel number, from 1 to the number of input channels on the soundcard, can be used.

5.2.9.10 SI_NoInput

Description

This **read-only** property indicates whether an input has been detected currently selected input soundcard. This property may be set, for example, if the currently selected soundcard is unplugged while soundcard analysis is being performed.

Values

True	No valid input has been detected.
False	A valid input has been detected.

5.2.10 Monitor Outputs

The Monitor Outputs section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"MonitorOutputs."**

Properties

MO_Mute
MO_GenBNC1
MO_GenBNC1Pulse
MO_GenBNC2
MO_GenBNC2Pulse
MO_DOMOnly
MO_AnaGain
MO_AnaBNC1Clipped
MO_AnaBNC2Clipped
MO_AnaBNC1
MO_AnaBNC1Pulse
MO_AnaBNC2
MO_AnaBNC2Pulse
MO_CarrierWaveform
MO_CarrierBNC2
MO_HeadphonesAndSpeaker

Methods

There are no methods available to control the dScope Monitor Outputs.

5.2.10.1 Properties

5.2.10.1.1 MO_Mute

Description

This property is used to mute or un-mute the Monitor Outputs.

This affects both the BNC connectors, and the headphones and speaker.

Values

True	Mute the Monitor Outputs.
False	Un-mute the Monitor Outputs.

5.2.10.1.2 MO_GenBNC1

Description

This property allows selection of the signal to be routed to Generator BNC 1.

Values

MO_GENBNC_CHA	Routes the output of channel A of the analogue generator to the BNC connector.
MO_GENBNC_CHB	Routes the output of channel B of the analogue generator to the BNC connector.

NB: **The Signal Generator generates the same signal in both the analogue and digital domains. It is the analogue domain signal that is routed to the Generator BNC connectors of the Monitor Outputs.**

5.2.10.1.3 MO_GenBNC1Pulse

Description

This property is used to select whether the signal routed to Generator BNC 1 should be a pulse signal or the signal as generated.

Values

True	Convert output to a pulse signal.
False	Route the output to the BNC, without converting to a pulse.

5.2.10.1.4 MO_GenBNC2

Description

This property allows selection of the signal to be routed to Generator BNC 2.

Values

MO_GENBNC_CHA	Routes the output of channel A of the analogue generator to the BNC connector.
MO_GENBNC_CHB	Routes the output of channel B of the analogue generator to the BNC connector.

NB: **The Signal Generator generates the same signal in both the analogue and digital domains. It is the analogue domain signal that is routed to the Generator BNC**

connectors of the Monitor Outputs.

5.2.10.1.5 MO_GenBNC2Pulse

Description

This property is used to select whether the signal routed to Generator BNC 2 should be a pulse signal or the signal as generated.

Values

True	Convert output to a pulse signal.
False	Route the output to the BNC, without converting to a pulse.

5.2.10.1.6 MO_DOMOnly

Description

This property is used to select whether the signal routed to the Generator BNCs should be the generated audio signal, or the Digital Output Modulation signals.

If DOM Only is selected, BNC 1 monitors the Digital Output common-mode interference signal and BNC 2 monitors the jitter modulation signal (if it is in the audio band).

Values

True	Generator BNCs monitor the Digital Output common-mode interference and jitter modulation signal as described above.
False	Generator BNCs monitor the generated audio signal.

5.2.10.1.7 MO_AnaGain

Description

This property allows selection of the gain to be applied to the Analyzer Monitor Outputs. The gain may be automatically gain ranged, or selected between 0dB and 120dB.

Values

MO_ANAGAIN_AUTO	Sets the Analyzer Monitor Outputs to automatically gain-range.
MO_ANAGAIN_0DB	Sets a fixed gain of 0dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_6DB	Sets a fixed gain of 6dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_12DB	Sets a fixed gain of 12dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_18DB	Sets a fixed gain of 18dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_24DB	Sets a fixed gain of 24dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_30DB	Sets a fixed gain of 30dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_36DB	Sets a fixed gain of 36dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_42DB	Sets a fixed gain of 42dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_48DB	Sets a fixed gain of 48dB on the Analyzer Monitor Outputs.

MO_ANAGAIN_54DB	Sets a fixed gain of 54dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_60DB	Sets a fixed gain of 60dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_66DB	Sets a fixed gain of 66dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_72DB	Sets a fixed gain of 72dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_78DB	Sets a fixed gain of 78dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_84DB	Sets a fixed gain of 84dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_90DB	Sets a fixed gain of 90dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_96DB	Sets a fixed gain of 96dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_102DB	Sets a fixed gain of 102dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_108DB	Sets a fixed gain of 108dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_114DB	Sets a fixed gain of 114dB on the Analyzer Monitor Outputs.
MO_ANAGAIN_120DB	Sets a fixed gain of 120dB on the Analyzer Monitor Outputs.

5.2.10.1.8 MO_AnaBNC1Clipped

Description

This **read-only** property is used to determine whether the signal routed to Analyzer BNC 1 is clipped.

Values

True	The signal routed to Analyzer BNC 1 is clipped.
False	The signal routed to Analyzer BNC 1 is not clipped.

5.2.10.1.9 MO_AnaBNC2Clipped

Description

This **read-only** property is used to determine whether the signal routed to Analyzer BNC 2 is clipped.

Values

True	The signal routed to Analyzer BNC 2 is clipped.
False	The signal routed to Analyzer BNC 2 is not clipped.

5.2.10.1.10 MO_AnaBNC1

Description

This property allows selection of the signal to be routed to Analyzer BNC 1.

Values

MO_ANABNC_CHA	Routes the input on channel A of the analyzer to the BNC connector.
MO_ANABNC_CHB	Routes the input on channel B of the analyzer to the BNC connector.
MO_ANABNC_SEL	When a single channel is being analyzed (See SA_Channel), routes the input of the selected channel to the BNC connector.
MO_ANABNC_NONSEL	When a single channel is being analyzed (See SA_Channel), routes the input of the *other* (i.e. non-selected) channel to the BNC

	connector.
MO_ANABNC_CTD_CHA	Routes the output of the Continuous-Time Detector (channel A) to the BNC connector.
MO_ANABNC_CTD_CHB	Routes the output of the Continuous-Time Detector (channel B) to the BNC connector.
MO_ANABNC_CTD_SEL	When a single channel is being analyzed (See SA_Channel), routes the output of the Continuous-Time Detector for the selected channel to the BNC connector.
MO_ANABNC_CTD_NONSEL	When a single channel is being analyzed (See SA_Channel), routes the output of the Continuous-Time Detector for the *other* (i.e. non-selected) channel to the BNC connector.

5.2.10.1.11 MO_AnaBNC1Pulse

Description

This property is used to select whether the signal routed to Analyzer BNC 1 should be a pulse signal or the input signal as selected.

Values

True	Convert output to a pulse signal.
False	Route the output to the BNC, without converting to a pulse.

5.2.10.1.12 MO_AnaBNC2

Description

This property allows selection of the signal to be routed to Analyzer BNC 2.

Values

MO_ANABNC_CHA	Routes the input on channel A of the analyzer to the BNC connector.
MO_ANABNC_CHB	Routes the input on channel B of the analyzer to the BNC connector.
MO_ANABNC_SEL	When a single channel is being analyzed (See SA_Channel), routes the input of the selected channel to the BNC connector.
MO_ANABNC_NONSEL	When a single channel is being analyzed (See SA_Channel), routes the input of the *other* (i.e. non-selected) channel to the BNC connector.
MO_ANABNC_CTD_CHA	Routes the output of the Continuous-Time Detector (channel A) to the BNC connector.
MO_ANABNC_CTD_CHB	Routes the output of the Continuous-Time Detector (channel B) to the BNC connector.
MO_ANABNC_CTD_SEL	When a single channel is being analyzed (See SA_Channel), routes the output of the Continuous-Time Detector for the selected channel to the BNC connector.
MO_ANABNC_CTD_NONSEL	When a single channel is being analyzed (See SA_Channel), routes the output of the Continuous-Time Detector for the *other* (i.e. non-selected) channel to the BNC connector.

5.2.10.1.13 MO_AnaBNC2Pulse

Description

This property is used to select whether the signal routed to Analyzer BNC 2 should be a pulse signal or the input signal as selected.

Values

True	Convert output to a pulse signal.
False	Route the output to the BNC, without converting to a pulse.

5.2.10.1.14 MO_CarrierWaveform

Description

This property is used to select whether the signal routed to the Analyzer BNCs should be the analyzed audio signal, or the carrier waveform.

If "Carrier waveform" is selected, BNC 1 monitors the waveform of the Digital Input Carrier (whichever is selected using DI_Source). In this mode, BNC 2 outputs a synchronization pulse (typically for oscilloscope triggering) as defined using the MO_CarrierBNC2 property.

Values

True	Analyzer BNCs monitor the carrier waveform as described above.
False	Analyzer BNCs monitor the analyzed audio signal.

5.2.10.1.15 MO_CarrierBNC2

Description

When monitoring the carrier waveform on BNC 1 (as specified using MO_CarrierWaveform), this property allows selection of the signal to be routed to Analyzer BNC 2.

Values

MO_CARRIERBNC2_X	Routes synchronization pulse from carrier X preamble to Analyzer BNC 2.
MO_CARRIERBNC2_Y	Routes synchronization pulse from carrier Y preamble to Analyzer BNC 2.
MO_CARRIERBNC2_BITCLOCK	Routes Bitclock to Analyzer BNC 2.
MO_CARRIERBNC2_GEN	Routes the selected Generator Reference Sync (see DO_RefSyncSource) to Analyzer BNC 2.

5.2.10.1.16 MO_CarrierBNC2VidDiv

Description

This property is used to select whether the output frame rate of a video Reference Sync signal selected on Carrier BNC 2 (See MO_CarrierBNC2) should be divided to ensure that output is triggered only when the start of the video frame is synchronized with the start of the audio frame.

NB: **This property is ignored unless the Digital Outputs Reference Sync (DO_RefSyncSource) is set to Video, and the output on Carrier BNC 2 (MO_CarrierBNC2) is set to the Generator Ref Sync (MO_CARRIERBNC2_GEN).**

Values

True	Divide the video Ref Sync to synchronize the video frame with the start of the audio frame.
False	Trigger the carrier BNC 2 output on every video frame.

5.2.10.1.17 MO_HeadphonesAndSpeaker

Description

This property allows selection of the signal to be routed to the headphones and speaker.

Values

MO_HEADPHONES_GENBNC1	Routes signal from Generator BNC 1 to the left and right headphone output, and to the speaker.
MO_HEADPHONES_GENBNC2	Routes signal from Generator BNC 2 to the left and right headphone output, and to the speaker.
MO_HEADPHONES_GENBNC1AND2	Routes signal from Generator BNC 1 to the left headphone output, the signal from Generator BNC 2 to the right headphone output, and a mono mix of the two to the speaker.
MO_HEADPHONES_ANABNC1	Routes signal from Analyzer BNC 1 to the left and right headphone output, and to the speaker.
MO_HEADPHONES_ANABNC2	Routes signal from Analyzer BNC 2 to the left and right headphone output, and to the speaker.
MO_HEADPHONES_ANABNC1AND2	Routes signal from Analyzer BNC 1 to the left headphone output, the signal from Analyzer BNC 2 to the right headphone output, and a mono mix of the two to the speaker.

NB: **Note that the Analyzer carrier mode is not reflected by the headphones and speaker, which continue to reflect the selections of the main Analyzer monitor; similarly, the 'pulse' mode does not affect them.**

5.3 Analyzer

The Analyzer section of this reference contains details of the following properties and methods.

Properties

There are no properties available to control the dScope Analyzer.

Methods

CreateFFTDetector
SetFFTDetector
RemoveFFTDetector

See the section below on Creating and Accessing FFT Detectors, for details on how to use these methods.

Parts of the Analyzer interface

Signal Analyzer
FFT Parameters
Continuous-Time Detector
FFT Detector

The Analyzed Channel Status is covered in the Channel Status section.

Creating and accessing FFT Detectors

The dScope software allows creation of up to 40 FFT Detectors, each of which can have a different analysis function. In this way, several different measurements can be taken at the same time, allowing for faster testing. FFT Detectors can also be set up to perform user-defined calculations, thus giving a great improvement in flexibility *and* speed.

FFT Detectors can be created and manipulated via scripts.

However, the dScope program only has a single interface for FFT Detectors, and so must provide a way of allowing access to several different Detectors through this one interface. It does this by always having a "current" FFT Detector, and any operations performed using the FFT Detector interface affect the current Detector.

To manipulate several different Detectors, a dScope script would have to use code like the following:

```
' This code snippet assumes that four FFT Detectors
' have already been opened, and they have ID numbers
' (as shown in their title bars) of 1 - 4.
Analyzer.SetFFTDetector(1)
FFTDetector.FFTD_Function = "Amplitude"

Analyzer.SetFFTDetector(2)
FFTDetector.FFTD_Function = "Gain"

Analyzer.SetFFTDetector(3)
FFTDetector.FFTD_Function = "THD"

Analyzer.SetFFTDetector(4)
FFTDetector.FFTD_Function = "Balance"
```

Note that before taking any action on each Detector, it must be set as the current Detector.

(Creating an FFT Detector from a script, using CreateFFTDetector(), will automatically set the Detector just created to be the current Detector, so there is no need to call SetFFTDetector(...) explicitly before performing operations on a Detector just created).

5.3.1 Methods

5.3.1.1 CreateFFTDetector

sDetectorID = CreateFFTDetector()

This method creates and opens an FFT Detector in the dScope, which can then be used for one of a number of analysis functions, including user-defined calculations allowing access to the full Sample and FFT buffers. By creating more than one FFT Detector, you can make several simultaneous measurements from a single FFT buffer.

NB: **Creating an FFT Detector will automatically set the Detector to be the current FFT Detector, so a call to SetFFTDetector is not necessary immediately after this call.**

Parameters

This method takes no parameters

Return Value

The return value is the ID of the FFT Detector. This is the same ID as is shown in the title bar of the Detector, and is unique amongst all the FFT Detectors currently open.

If the function fails for some reason (either there is not enough memory, or there are already 40 FFT Detectors open), then this method will return **-1.**

The FFT Detector ID returned should be used in calls to SetFFTDetector and RemoveFFTDetector, to start using this Detector or to remove it from the system.

This FFT Detector ID will no longer be valid after the FFT Detector has been removed by calling RemoveFFTDetector with this Detector ID.

NB: **This method must NOT be called from within an FFT Detector Calculation Script.**

5.3.1.2 SetFFTDetector

bRet = SetFFTDetector (sDetectorID)

This method should be called to set the current FFT Detector. All subsequent methods called on the **FFTDetector** part of the dScope will then act on this Detector, until it is called with a different Detector ID.

Parameters

sDetectorID	Pass the Detector ID of the Detector that you wish to set as the current Detector. This will be the ID returned by CreateFFTDetector(), or (for a Detector loaded in a Configuration), the ID number shown in the Detector's title bar.

Return value

This method will return **True** if the Detector exists and was successfully set as the current Detector, or **False** if it doesn't exist.

NB:

This method must NOT be called from within an FFT Detector Calculation Script.

5.3.1.3 RemoveFFTDetector

bRet = RemoveFFTDetector (sDetectorID)

This method should be called to remove the FFT Detector with the given ID.

NB:

If any Readings have been created from the Detector with the given ID, they will be automatically closed when this is called.

Also, if the Sweep settings have been set up to sweep the Result of the FFT Detector that is being removed, the Result will be set to "- None -" on the Sweep Setup window. If a Sweep is currently in progress, it will be stopped automatically before this occurs.

Parameters

sDetectorID	Pass the Detector ID of the Detector that you wish to remove. This will be the ID returned by CreateFFTDetector(), or (for a Detector loaded in a Configuration), the ID number shown in the Detector's title bar.

Return value

This method will return **True** if the Detector exists and was successfully removed, or **False** if it didn't exist.

NB:

This method must NOT be called from within an FFT Detector Calculation Script.

5.3.2 Signal Analyzer

The Signal Analyzer section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"SignalAnalyzer."**

Properties

SA_Source
SA_Channel
SA_UpdateRate
SA_ChARMSAmpl
SA_ChBRMSAmpl
SA_RMSAmplUnit
SA_ChAFreq
SA_ChBFreq

Methods

5.3.2.1 Properties

5.3.2.1.1 SA_Source

Description

This property allows selection of whether the Signal Analyzer is to analyze the Analogue or Digital Inputs.

The Digital Input is as selected using DI_Source, and the Analogue Input as selected by AI_Source.

Values

SA_DIGITAL	Selects the Digital Inputs to be analyzed.
SA_ANALOGUE	Selects the Analogue Inputs to be analyzed.
SA_SOUNDCARD	Selects the Soundcard Inputs to be analyzed. NB: This option will only be available if a valid Soundcard has been set up for the Soundcard Inputs (see SI_Soundcard)

5.3.2.1.2 SA_Channel

Description

This property allows selection of the channel(s) to analyze.

Note that this will affect which channel is analyzed on the Trace Window and by FFT Detectors. The Signal Analyzer and Continuous-Time Detector will *always* analyze both channels.

Values

SA_CHA	Selects analysis of channel A only.
SA_CHB	Selects analysis of channel B only.
SA_CHBOTH	Selects analysis of both channels.

Note that the following values (which can be applied throughout the system where a channel selection is required) can also be used :

CHANNEL_A	Selects analysis of channel A only.
CHANNEL_B	Selects analysis of channel B only.
CHANNEL_BOTH	Selects analysis of both channels.

5.3.2.1.3 SA_UpdateRate

Description

This property allows selection of the update rate for the Signal Analyzer and Continuous-Time Detector. This is the rate at which the software attempts to read data from the hardware.

An automatic update rate (**SA_UPDATERATE_AUTO**) is usually satisfactory; however lower rates may be selected to try and reduce Result variations for complex input signals with low frequency content such as noise.

Values

SA_UPDATERATE_AUTO	Selects the update rate to be automatic (default).
SA_UPDATERATE_4	Selects an update rate of 4 measurements per second.
SA_UPDATERATE_8	Selects an update rate of 8 measurements per second.
SA_UPDATERATE_16	Selects an update rate of 16 measurements per second.
SA_UPDATERATE_32	Selects an update rate of 32 measurements per second.

5.3.2.1.4 SA_ChARMSAmpl

Description

This **read-only** property represents the current RMS Amplitude of the analyzed signal on channel A.

The value is returned in the current RMS amplitude unit, as selected by SA_RMSAmplUnit.

Values

The RMS amplitude is represented as a double-precision floating point value.

5.3.2.1.5 SA_ChBRMSAmpl

Description

This **read-only** property represents the current RMS Amplitude of the analyzed signal on channel B.

The value is returned in the current RMS amplitude unit, as selected by SA_RMSAmplUnit.

Values

The RMS amplitude is represented as a double-precision floating point value.

5.3.2.1.6 SA_RMSAmplUnit

Description

This property allows selection of the unit for measurement of the RMS amplitude of the analyzed signal on channels A and B.

This specifies which unit the values returned by SA_ChARMSAmpl and SA_ChBRMSAmpl will be returned in.

Values

Under normal analysis, the following values are allowed:

UNIT_DBFS	Sets RMS Amplitude unit to dBFS.
UNIT_PERCENTFS	Sets RMS Amplitude unit to %FS (percentage of full scale).
UNIT_FFS	Sets RMS Amplitude unit to FFS (fraction of full scale).
UNIT_HEX	Sets RMS Amplitude unit to Hex.
UNIT_V	Sets RMS Amplitude unit to V.
UNIT_DBU	Sets RMS Amplitude unit to dBu.
UNIT_DBV	Sets RMS Amplitude unit to dBV.
UNIT_DBM	Sets RMS Amplitude unit to dBm.
UNIT_W	Sets RMS Amplitude unit to W.
UNIT_DBSPL	Sets RMS Amplitude unit to dBSPL.
UNIT_DBR	Sets RMS Amplitude unit to dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Sets RMS Amplitude unit to percentage of the reference amplitude (SA_RefAmpl).

If the analyzer is currently set up to analyze the demodulated jitter signal through the Analogue Inputs (See AI_Source for further details), then the following values are allowed:

UNIT_JITTER_NS	Sets RMS Amplitude unit for jitter values to ns.
UNIT_JITTER_UI	Sets RMS Amplitude unit for jitter values to UI.

5.3.2.1.7 SA_ChAFreq

Description

This **read-only** property represents the current frequency of the analyzed signal on channel A.

The value is returned in the current frequency unit, as selected by SA_FreqUnit.

Values

The frequency is represented as a double-precision floating point value.

5.3.2.1.8 SA_ChBFreq

Description

This **read-only** property represents the current frequency of the analyzed signal on channel B.

The value is returned in the current frequency unit, as selected by SA_FreqUnit.

Values

The frequency is represented as a double-precision floating point value.

5.3.2.1.9 SA_FreqUnit

Description

This property allows selection of the unit for measurement of the frequency of the input signal on channel A and B.

This specifies which unit the values returned by SA_ChAFreq and SA_ChBFreq will be returned in.

Values

UNIT_FREQ_HZ	Sets frequency unit to Hz.
UNIT_FREQ_OFFSET	Sets frequency unit to be an offset from the reference frequency (See SA_RefFreq), in Hz.
UNIT_FREQ_RATIO	Sets frequency unit to be a ratio of the reference frequency (see SA_RefFreq).

5.3.2.1.10 SA_Phase

Description

This **read-only** property represents the current inter-channel phase, in the unit specified by SA_PhaseUnit.

Values

The inter-channel phase is represented as a double-precision floating point value.

5.3.2.1.11 SA_PhaseUnit

Description

This property allows selection of the unit for measurement of the inter-channel phase.

This specifies which unit the value returned by SA_Phase will be returned in.

Values

UNIT_PHASE_DEGREES	Sets unit for inter-channel phase measurement to degrees.
UNIT_PHASE_RADIANS	Sets unit for inter-channel phase measurement to radians.
UNIT_PHASE_US	Sets unit for inter-channel phase measurement to µs (microseconds).
UNIT_PHASE_SAMPLES	Sets unit for phase measurement to samples.

NB: **UNIT_PHASE_SAMPLES is only available if the currently analyzed signal is digital (See SA_Source).**

5.3.2.1.12 SA_RefAmpl

Description

This property allows specification of the reference amplitude used throughout the dScope Analyzer for measurements in dBr (**UNIT_DBR**) and **% ref** (**UNIT_PERCENTREF**).

Note that this includes Trace scales, and measurements for Sweeps.

The value must be specified in the unit selected by SA_RefAmplUnit.

Changing this property will set the reference amplitude of both channel A and B (SA_ChARefAmpl and SA_ChBRefAmpl), and will also tie the reference amplitudes together (see SA_RefAmplTied).

NB: **If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the generator reference amplitude (SG_RefAmpl).**

Values

The reference amplitude is represented as a double-precision floating point value.

5.3.2.1.13 SA_ChARefAmpl

Description

This property allows specification of the reference amplitude used throughout the dScope Analyzer for measurements on channel A in dBr (**UNIT_DBR**) and % ref (**UNIT_PERCENTREF**).

Note that this includes Trace scales, and measurements for Sweeps.

The value must be specified in the unit selected by SA_RefAmplUnit.

NB: **If the reference amplitudes are tied together (see SA_RefAmplTied), then changing this property will also change the channel B reference amplitude (SA_ChBRefAmpl).**

If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the equivalent generator reference amplitude (SG_ChARefAmpl).

Values

The reference amplitude is represented as a double-precision floating point value.

5.3.2.1.14 SA_ChBRefAmpl

Description

This property allows specification of the reference amplitude used throughout the dScope Analyzer for measurements on channel B in dBr (**UNIT_DBR**) and % ref (**UNIT_PERCENTREF**).

Note that this includes Trace scales, and measurements for Sweeps.

The value must be specified in the unit selected by SA_RefAmplUnit.

NB:

If the reference amplitudes are tied together (see SA_RefAmplTied), then changing this property will also change the channel A reference amplitude (SA_ChARefAmpl).

If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the equivalent generator reference amplitude (SG_ChBRefAmpl).

Values

The reference amplitude is represented as a double-precision floating point value.

5.3.2.1.15 SA_RefAmplTied

Description

This property allows the Analyzer reference amplitudes to be tied together. This means that changing the channel A reference amplitude (SA_ChARefAmpl) will automatically update channel B's reference amplitude (SA_ChBRefAmpl) to be the same, and vice versa.

NB:

If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the equivalent option on the Signal Generator (see SG_RefAmplTied).

Values

True	Specifies that analyzer reference amplitudes should be tied together.
False	Specifies that analyzer reference amplitudes should be separate (not tied together).

5.3.2.1.16 SA_RefAmplUnit

Description

This property allows selection of the unit for the reference amplitude used by the Analyzer, as specified using SA_RefAmpl.

NB: **If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the generator reference amplitude's unit (SG_RefAmplUnit).**

Values

UNIT_DBFS	Sets reference amplitude unit to dBFS.
UNIT_PERCENTFS	Sets reference amplitude unit to %FS (percentage of full scale).
UNIT_FFS	Sets reference amplitude unit to FFS (fraction of full scale).
UNIT_HEX	Sets reference amplitude unit to Hex.
UNIT_VRMS	Sets reference amplitude unit to an RMS voltage.
UNIT_VP	Sets reference amplitude unit to a peak voltage.
UNIT_VPP	Sets reference amplitude unit to a peak-to-peak voltage.
UNIT_DBU	Sets reference amplitude unit to dBu.
UNIT_DBV	Sets reference amplitude unit to dBV.
UNIT_DBM	Sets reference amplitude unit to dBm.
UNIT_W	Sets reference amplitude unit to W.
UNIT_DBSPL	Sets reference amplitude unit to dBSPL.

NB: **If the reference amplitude is specified as an RMS voltage, but the measurement is a peak Result (or vice-versa), then the dScope assumes that the signal is a sine wave for purposes of conversion between RMS and peak values.**

For example, if the Continuous-Time Detector is measuring a peak amplitude, but the reference amplitude is specified in an RMS unit (e.g. dBu), then the reference amplitude will be converted to a peak amplitude (assuming a sine wave) for the Continuous-Time Detector.

5.3.2.1.17 SA_RefFreq

Description

This property allows specification of the reference frequency used throughout the dScope Signal Analyzer for measurements relative to the reference frequency (**UNIT_FREQ_OFFSET** and **UNIT_FREQ_RATIO**).

Note that this includes Trace scales, and measurements for Sweeps.

The reference frequency is specified in Hz.

NB: **If the Options settings are set up to lock together the reference frequency of the generator and analyzer (See OPT_LockRefFreq), then changing this property will also change the generator reference frequency (SG_RefFreq).**

Values

The reference frequency is represented as a double-precision floating point value.

5.3.2.1.18 SA_RefImpedance

Description

This property allows specification of the reference impedance used throughout the dScope Signal Analyzer for measurements that involve the impedance (dBm and W).

The reference impedance is specified in Ohms.

Values

The reference impedance is represented as a double-precision floating point value.

5.3.2.1.19 SA_dBSPLValue

Description

This property allows specification of the number of dBSPL that equates to the reference level specified using SA_SPLRef.

Values

The number of dBSPL equating to the reference level is represented as a double-precision floating point value.

5.3.2.1.20 SA_SPLRef

Description

This property allows specification of the reference level used throughout the dScope analyzer for the dBSPL unit. The value entered is equivalent to the number of dBSPL entered using SA_dBSPLValue.

Note that this includes Trace scales, and measurements for Sweeps.

The value must be specified in the unit selected by SA_SPLRefUnit.

Values

The dBSPL reference is represented as a double-precision floating point value.

5.3.2.1.21 SA_SPLRefUnit

Description

This property allows selection of the unit for the dBSPL reference used by the analyzer, as specified using SA_SPLRef

Values

UNIT_DBFS	Sets dBSPL reference unit to dBFS.
UNIT_PERCENTFS	Sets dBSPL reference unit to %FS (percentage of full scale).
UNIT_FFS	Sets dBSPL reference unit to FFS (fraction of full scale).
UNIT_HEX	Sets dBSPL reference unit to Hex.
UNIT_VRMS	Sets dBSPL reference unit to an RMS voltage.
UNIT_VP	Sets dBSPL reference unit to a peak voltage.
UNIT_VPP	Sets dBSPL reference unit to a peak-to-peak voltage.
UNIT_DBU	Sets dBSPL reference unit to dBu.
UNIT_DBV	Sets dBSPL reference unit to dBV.
UNIT_DBM	Sets dBSPL reference unit to dBm.
UNIT_W	Sets dBSPL reference unit to W.

NB:

If the dBSPL reference is specified as an RMS voltage, but the measurement is a peak Result (or vice-versa), then the dScope assumes that the signal is a sine wave for purposes of conversion between RMS and peak values.

For example, if the Continuous-Time Detector is measuring a peak amplitude, but the dBSPL reference is specified in an RMS unit (e.g. dBu), then the dBSPL reference will be converted to a peak amplitude (assuming a sine wave) for the Continuous-Time Detector.

5.3.2.1.22 SA_Gain

Description

This property allows specification of a pre-amplifier gain for the dScope's Analyzer. This gain is subtracted from all Analyzer results (Signal Analyzer, Continuous-Time Detector and FFT Detector) before the value is displayed. Note that this includes Trace scales, and measurements for Sweeps.

The value must be specified in the unit selected by SA_GainUnit.

Values

The gain is represented as a double-precision floating point value.

5.3.2.1.23 SA_GainUnit

Description

This property allows selection of the unit for the pre-amplifier gain used by the dScope's Analyzer, as specified using SA_Gain.

Values

UNIT_RELATIVE_DB	Sets the Analyzer gain unit to dB.
UNIT_RELATIVE_GAIN	Sets the Analyzer gain unit to a gain (where 1.0 is unity gain)

5.3.2.1.24 SA_DALineUp

Description

This property allows specification of the D/A line-up used throughout the dScope Signal Analyzer.

The value must be specified in the unit selected by SA_DALineUpUnit.

NB: **If the Options settings are set up to lock together the D/A line-up of the Signal Generator and Signal Analyzer (See OPT_LockDALineUp), then changing this property will also change the generator D/A line-up (SG_DALineUp).**

Values

The D/A line-up amplitude is represented as a double-precision floating point value.

5.3.2.1.25 SA_DALineUpUnit

Description

This property allows selection of the unit for the D/A line-up used by the Analyzer, as specified using SA_DALineUp.

NB: **If the Options settings are set up to lock together the D/A line-up of the generator and analyzer (See OPT_LockDALineUp), then changing this property will also change the generator D/A line-up's unit (SG_DALineUpUnit).**

Values

UNIT_VRMS	Sets D/A line-up unit to Volts, RMS.
UNIT_VP	Sets D/A line-up unit to Volts, peak.
UNIT_VPP	Sets D/A line-up unit to Volts, peak-to-peak.
UNIT_DBU	Sets D/A line-up unit to dBu.
UNIT_DBV	Sets D/A line-up unit to dBV.
UNIT_DBM	Sets D/A line-up unit to dBm.
UNIT_W	Sets D/A line-up unit to W.
UNIT_DBSPL	Sets D/A line-up unit to dBSPL.

5.3.2.1.26 SA_DefaultHPFilter

Description

This property allows selection of the default high-pass filter for the Continuous-Time Detector and FFT Detectors.

NB: **The default filter settings are not used in calculating the amplitude Results displayed in the Signal Analyzer panel itself. They are located centrally in the Signal Analyzer panel so that filters in the CT and FFT Detectors can be centrally switched if desired.**

Values

SA_HP_OFF	Sets default high-pass filter to off (see note below).
SA_HP_DCB	Sets default high-pass filter to a DC blocking filter.
SA_HP_10HZ	Sets default high-pass filter to 10Hz.
SA_HP_22HZ	Sets default high-pass filter to 22Hz.
SA_HP_100HZ	Sets default high-pass filter to 100Hz.
SA_HP_400HZ	Sets default high-pass filter to 400Hz.

NB: **1) The dScope analogue hardware has a built-in DC blocking filter, which can be turned on or off using a jumper on the board (see PCB jumper options for further details).**

In analogue analysis mode, if the analogue hardware is not DC coupled, then selection of "SA_HP_OFF" is disabled and the DC blocking filter is automatically selected (SA_HP_DCB).

NB: **2) To set a specific frequency for the high-pass filter, use SA_DefaultLPFilterFreq .**

5.3.2.1.27 SA_DefaultHPFilterFreq

Description

This property allows specification of the frequency of the default high-pass filter for the Continuous-Time Detector and FFT Detectors.

NB: **The default filter settings are not used in calculating the amplitude Results displayed in the Signal Analyzer panel itself. They are located centrally in the Signal Analyzer panel so that filters in the CT and FFT Detectors can be centrally switched if desired.**

Values

The default high-pass filter frequency is represented as a long integer value.

NB: **If the Default high-pass filter is one of the predefined filters (See SA_DefaultHPFilter), then this property will return the *actual* frequency that this represents.**
For example, if SA_DefaultHPFilter has been set to SA_HP_10HZ, then this property will return 10. DC Block (SA_HP_DCB) and Off (SA_HP_OFF) will both return zero.

5.3.2.1.28 SA_DefaultLPFilter

Description

This property allows selection of the default low-pass filter for the Continuous-Time Detector and FFT Detectors.

NB: **The default filter settings are not used in calculating the amplitude Results displayed in the Signal Analyzer panel itself. They are located centrally in the Signal Analyzer panel so that filters in the CT and FFT Detectors can be centrally switched if desired.**

Values

SA_LP_OFF	Sets default low-pass filter to off.
SA_LP_22KHZ	Sets default low-pass filter to 22kHz.
SA_LP_30KHZ	Sets default low-pass filter to 30kHz.
SA_LP_40KHZ	Sets default low-pass filter to 40kHz.
SA_LP_80KHZ	Sets default low-pass filter to 80kHz.
SA_LP_20KHZ_AES17	Sets the default low-pass filter to a 20kHz filter that matches the AES17 low-pass filter specification.

NB: **To set a specific frequency for the low-pass filter, use SA_DefaultLPFilterFreq.**

5.3.2.1.29 SA_DefaultLPFilterFreq

Description

This property allows specification of the frequency of the default low-pass filter for the Continuous-Time Detector and FFT Detectors.

NB: **The default filter settings are not used in calculating the amplitude Results displayed in the Signal Analyzer panel itself. They are located centrally in the Signal Analyzer panel so that filters in the CT and FFT Detectors can be centrally switched if desired.**

Values

The default low-pass filter frequency is represented as a long integer value.

NB: **If the Default low-pass filter is one of the predefined filters (See SA_DefaultLPFilter), then this property will return the *actual* frequency that this represents.**
For example, if SA_DefaultLPFilter has been set to SA_LP_30KHZ, then this property will return 30000. Off (SA_LP_OFF) will return zero.

5.3.2.1.30 SA_DefaultWeightingFilter

Description

This property allows selection of the default weighting filter for the Continuous-Time Detector and FFT Detectors.

NB: **The default filter settings are not used in calculating the amplitude Results displayed in the Signal Analyzer panel itself. They are located centrally in the Signal Analyzer panel so that filters in the CT and FFT Detectors can be centrally switched if desired.**

Values

SA_WEIGHTING_NONE	Sets default weighting filter to none (not weighted).
SA_WEIGHTING_AWEIGHTING	Sets default weighting filter to A-weighted.
SA_WEIGHTING_CWEIGHTING	Sets default weighting filter to C-weighted.
SA_WEIGHTING_CCIR468_1K	Sets default weighting filter to a CCIR-468 shape, with unity gain at 1kHz.
SA_WEIGHTING_CCIR468_2K	Sets default weighting filter to a CCIR-468 shape, with unity gain at 2kHz.

5.3.2.2 Methods

5.3.2.2.1 SA_RefAmplFromChA

SA_RefAmplFromChA()

This method can be used to set the Signal Analyzer's reference amplitude (and its unit) to the same level as the current input signal on channel A.

NB: **This method does *not* ensure that the amplitude has settled before setting it as the reference. If you wish to ensure that the result has settled, you can firstly read the amplitude using SA_ChARMSAmpl.**

Parameters

This method has no parameters.

Return value

This method has no return value.

5.3.2.2.2 SA_RefAmplFromChB

SA_RefAmplFromChB()

This method can be used to set the Signal Analyzer's reference amplitude (and its unit) to the same level as the current input signal on channel B.

NB: **This method does *not* ensure that the amplitude has settled before setting it as the reference. If you wish to ensure that the result has settled, you can firstly read the amplitude using SA_ChBRMSAmpl.**

Parameters

This method has no parameters.

Return value

This method has no return value.

5.3.2.2.3 SA_RefFreqFromChA

SA_RefFreqFromChA()

This method can be used to set the Signal Analyzer's reference frequency to the same frequency as the current input signal on channel A.

NB: **This method does *not* ensure that the frequency has settled before setting it as the reference. If you wish to ensure that the result has settled, you can firstly read the frequency using SA_ChAFreq.**

Parameters

This method has no parameters.

Return value

This method has no return value.

5.3.2.2.4 SA_RefFreqFromChB

SA_RefFreqFromChB()

This method can be used to set the Signal Analyzer's reference frequency to the same frequency as the current input signal on channel B.

NB: **This method does *not* ensure that the frequency has settled before setting it as the reference. If you wish to ensure that the result has settled, you can firstly read the frequency using SA_ChBFreq.**

Parameters

This method has no parameters.

Return value

This method has no return value.

5.3.3 FFT Parameters

The FFT Parameters section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"FFTParameters."**

Properties

FFTP_NumPoints
FFTP_WindowFunction
FFTP_UserWindowFunction
FFTP_WeightingFilter
FFTP_UserWeightingFilter
FFTP_AverageSamples
FFTP_AverageTypeSamples
FFTP_AverageTimesSamples
FFTP_AveragesDoneSamples
FFTP_Average
FFTP_AverageType
FFTP_AverageTimes
FFTP_AveragesDone
FFTP_TriggerMode
FFTP_TriggerPoint
FFTP_ThresholdMode
FFTP_Threshold
FFTP_ThresholdUnit
FFTP_ThresholdPolarity
FFTP_TriggerChannel
FFTP_TriggerOnCTDetector
FFTP_TriggerOn
FFTP_BuffersProcessed
FFTP_CalcPhaseInfo

Methods

FFTP_GetWindowSpread
FFTP_ExportSampleBuffer
FFTP_ImportSampleBuffer

5.3.3.1 Properties

5.3.3.1.1 FFTP_NumPoints

Description

This property allows selection of the number of FFT points to use.

Values

FFTP_NUMPOINTS_1K	Selects the FFT size to be 1k (1024) samples.
FFTP_NUMPOINTS_2K	Selects the FFT size to be 2k (2048) samples.
FFTP_NUMPOINTS_4K	Selects the FFT size to be 4k (4096) samples.
FFTP_NUMPOINTS_8K	Selects the FFT size to be 8k (8192) samples.
FFTP_NUMPOINTS_16K	Selects the FFT size to be 16k (16384) samples.

FFTP_NUMPOINTS_32K	Selects the FFT size to be 32k (32768) samples.
FFTP_NUMPOINTS_64K	Selects the FFT size to be 64k (65536) samples.
FFTP_NUMPOINTS_128K	Selects the FFT size to be 128k (131072) samples.
FFTP_NUMPOINTS_256K	Selects the FFT size to be 256k (262144) samples.
FFTP_NUMPOINTS_512K	Selects the FFT size to be 512k (524288) samples.
FFTP_NUMPOINTS_1M	Selects the FFT size to be 1M (1048576) samples.

NB:

This is the number of samples used to perform the FFT. The number of resulting bins is half this amount, since the FFT calculation results in positive and negative frequency values, of which the dScope only needs the positive side.

5.3.3.1.2 FFTP_WindowFunction

Description

This property allows specification of a Window function of the number of FFT points to use.

Values

FFTP_WINDOW_RECTANGULAR	Selects the FFT Window function to be rectangular. Note that this is the same as having no Window function at all, and should not be used except in circumstances where the signal period is exactly the same as the sample buffer length.
FFTP_WINDOW_FREQCORRECT_AO	Selects the FFT Window function to be rectangular (i.e. no Window function), with frequency correction at the input to ensure that tones are centred in the bins of the FFT buffer. When performing this frequency correction, this Window function assumes that the signal was originally generated at the Analogue Output.
FFTP_WINDOW_FREQCORRECT_DO	Selects the FFT Window function to be rectangular (i.e. no Window function), with frequency correction at the input to ensure that tones are centred in the bins of the FFT buffer. When performing this frequency correction, this Window function assumes that the signal was originally generated at the Digital Output.
FFTP_WINDOW_NSHOTCORRECT	Selects the FFT Window function to be rectangular, 'tilting' the input buffer to remove slight discontinuities between the first and last samples. This should only be used if the Signal Generator's waveform is playing a specified number of times, rather than continuously.
FFTP_WINDOW_TRIANGULAR	Selects the FFT Window function to be triangular.
FFTP_WINDOW_BLACKMAN	Selects the Blackman FFT Window function.
FFTP_WINDOW_HANN	Selects the Hann FFT Window function.
FFTP_WINDOW_HAMMING	Selects the Hamming FFT Window function.
FFTP_WINDOW_BH4	Selects the Blackman-Harris 4 FFT Window function.
FFTP_WINDOW_GAUSSIAN	Selects a gaussian FFT Window function.
FFTP_WINDOW_FLATTOP	Selects the Prism Sound flat-top FFT Window function. Provides a Window function with very little leakage of signal from the bin containing a tone into adjacent bins.
FFTP_WINDOW_PRISM5	Selects the Prism Sound 5-term FFT Window function. Provides optimal frequency resolution with minimum spreading of tone frequencies into adjacent bins.

FFTP_WINDOW_PRISM6	Selects the Prism Sound 6-term FFT Window function.
FFTP_WINDOW_PRISM7	Selects the Prism Sound 7-term FFT Window function. Gives the best dynamic range at the expense of a little spreading of tone frequencies into adjacent bins.
FFTP_WINDOW_USER	Selects the user-defined FFT Window function specified by FFTP_UserWindowFunction.

NB: **The optimum Window functions are Prism 5, 6 and 7 which have a very high dynamic range with minimal broadening (Prism 7 has the widest dynamic range at nearly 150dB, but the most broadening of the three). The other Window functions are inferior to the Prism Window functions in most applications, since their 'skirts' prevent sufficient dynamic range for measuring modern audio systems. They are included only to provide commonality with other analyzers and theoretical papers.**

5.3.3.1.3 FFTP_UserWindowFunction

Description

This property allows specification of a user-defined Window function for the FFT window. This will be the file name of a Window function table, or a script used to create such a table.

For further details on setting up user-defined Window functions, see FFT Detector Window functions. For a full reference of script functions available to write scripts to create Window functions, see FFT Window function reference.

Values

Any valid filename of a Window function table (*.wnd), or a dScope script file (*.dss) is allowed.

NB: **If a full file and path are not specified, the system will attempt to find the file by appending the default file extension for Window functions (*.wnd) and looking for the file in the "FFT Windows" subfolder of the dScope program folder.**

5.3.3.1.4 FFTP_WeightingFilter

Description

This property allows selection of a pre-weighting filter to be applied to the FFT Analyzer. This weighting is applied to the FFT buffer directly after calculation, and will affect all FFT Detectors as well as FFT Traces.

NB: **Any filters applied by FFT Detectors will be in ADDITION to the filter specified here. If the same filter is selected in both an FFT Detector AND the FFT Parameters, it will be applied twice, giving incorrect results.**

Values

FFTP_WEIGHTING_NONE	Sets weighting filter for the FFT Analyzer to none (not weighted).
FFTP_WEIGHTING_AWEIGHTING	Sets weighting filter for the FFT Analyzer to A-weighted.
FFTP_WEIGHTING_CWEIGHTING	Sets weighting filter for the FFT Analyzer to C-weighted.
FFTP_WEIGHTING_CCIR468_1K	Sets weighting filter for the FFT Analyzer to a CCIR-468 shape, with unity gain at 1kHz.

FFTP_WEIGHTING_CCIR468_2K	Sets weighting filter for the FFT Analyzer to a CCIR-468 shape, with unity gain at 2kHz.
FFTP_WEIGHTING_USER	Sets weighting filter for the FFT Analyzer to use the user-defined weighting filter specified by FFTP_UserWeightingFilter.

5.3.3.1.5 FFTP_UserWeightingFilter

Description

This property allows specification of a user-defined pre-weighting filter to be applied to the FFT Analyzer. This weighting is applied to the FFT buffer directly after calculation, and will affect all FFT Detectors as well as FFT Traces.

NB: **Any filters applied by FFT Detectors will be in ADDITION to the filter specified here. If the same filter is selected in both an FFT Detector AND the FFT Parameters, it will be applied twice, giving incorrect results.**

For further details on setting up user-defined weighting filters, see FFT Detector Weighting filters. For a full reference of script functions available to write scripts to create weighting filters, see the FFT Detector Weighting filter reference section.

Values

Any valid filename of a weighting filter table (*.wgt), or a dScope script file (*.dss) is allowed.

NB: **If a full file and path are not specified, the system will attempt to find the file by appending the default file extension for weighting filters (*.wgt) and looking for the file in the "FFT Detector Weighting filters" subfolder of the dScope program folder.**

5.3.3.1.6 FFTP_AverageSamples

Description

This property is used to select whether to average a number of sample buffers.

When averaging is enabled, the dScope averages the number of successively-triggered sample buffers specified by FFTP_AverageTimesSamples and then disarms the trigger. The averaging can be done on successive triggered buffers, or on contiguous data in the hardware before the data is made available to the application. See FFTP_AverageTypeSamples for further details.

The trigger must be re-armed (using FFTP_TriggerOn) to start another averaging series.

Values

True	Average the prescribed number of sample buffers.
False	Do not perform sample buffer averaging.

5.3.3.1.7 FFTP_AverageTypeSamples

Description

This property allows selection of the type of averaging to apply to the incoming sample buffers when sample buffer averaging is turned on (See FFTP_AverageSamples).

Values

FFTP_AVERAGETYPE_CONTIGUOUS	Selects contiguous averaging of sample buffers. This averaging is done in the hardware and ensures that the buffers to be averaged are contiguous, with no gaps in between them. With this type of averaging, it is important that the signal in the sample buffer contains a whole number of cycles within the buffer size, otherwise discontinuities will cause problems with the averaging. NB: This option is not available when the analyzer sample rate is 192kHz.
FFTP_AVERAGETYPE_TRIGGERED	Selects triggered averaging of the sample buffer. Each sample buffer that is read into the application will be averaged with the previous one. For this type of averaging, it is important that the signal is triggered at the same point in the signal each time.

5.3.3.1.8 FFTP_AverageTimesSamples

Description

This property is used to select how many times to average the sample buffers. This property is ignored unless sample buffer averaging is turned on using FFTP_AverageSamples.

When sample buffer averaging is on, the dScope averages the number of successively-triggered sample buffers specified and then disarms the trigger.

Values

The number of times to average can be any number between 1 and 128.

5.3.3.1.9 FFTP_AveragesDoneSamples

Description

This **read-only** property represents the number of sample buffers averaged so far. It can be any number between 0 and the number of times to average sample buffers (FFTP_AverageTimesSamples).

This property can be used to determine when averaging is finished, by checking the number of averages done against the number originally required.

Values

The number of averages done is represented as a short integer value.

5.3.3.1.10 FFTP_Average

Description

This property is used to select whether to average a number of FFT buffers.

This can be useful in order to resolve frequency components hidden in the noise of an FFT display; averaging has the effect of reducing variations in the displayed noise floor and so emphasizes real low-level components and artifacts.

When averaging is enabled, the dScope averages the number of successively-triggered FFTs specified by FFTP_AverageTimes and then disarms the trigger. The FFT is displayed after each averaging pass, so the gradual smoothing of the noise floor can be observed.

The trigger must be re-armed (using FFTP_TriggerOn) to start another averaging series.

Values

True	Average the prescribed number of FFTs
False	Do not perform FFT averaging.

5.3.3.1.11 FFTP_AverageType

Description

This property allows selection of the type of averaging to apply to the FFT buffers when FFT averaging is turned on (See FFTP_Average).

Values

FFTP_AVERAGETYPE_ONCE	Selects a single averaging of the FFT buffers. The number of averages performed can be defined using FFTP_AverageTimes. Once averaging has finished, the trigger will be disarmed and must be armed again (if required) using FFTP_TriggerOn.
FFTP_AVERAGETYPE_PSEUDOROLLING	Selects pseudo-rolling averaging of the FFT buffers. This averaging will occur continuously, using the number of buffers specified by FFTP_AverageTimes. The trigger will *not* be disarmed automatically if using this type of averaging.

5.3.3.1.12 FFTP_AverageTimes

Description

This property is used to select how many times to average the FFT. This property is ignored unless the FFT averaging is turned on using FFTP_Average.

When averaging is on, the dScope averages the number of successively-triggered FFTs specified. If the average type (see FFTP_AverageType) is set to **FFTP_AVERAGETYPE_ONCE**, the trigger is disarmed once averaging has finished.

Values

The number of times to average can be any number between 1 and 1000. A value of 1 is equivalent to a "single-shot" trigger (see FFTP_TriggerMode).

5.3.3.1.13 FFTP_AveragesDone

Description

This **read-only** property represents the number of FFT buffers averaged so far. It can be any number between 0 and the number of times to average FFT buffers (FFTP_AverageTimes).

This property can be used to determine when averaging is finished, by checking the number of averages done against the number originally required.

Values

The number of averages done is represented as a short integer value.

5.3.3.1.14 FFTP_TriggerMode

Description

This property allows selection of the trigger mode for the FFT.

Values

FFTP_TRIGGERMODE_CONTINUOUS	Selects the trigger mode to be continuous. The sample collection will be triggered immediately after the buffer has been read from the last triggering. If this option is selected, then the current threshold settings are ignored.
FFTP_TRIGGERMODE_NORMAL	Selects the normal trigger mode. This will trigger when the threshold is detected, and after a buffer has been read, will trigger again once the threshold is reached again.
FFTP_TRIGGERMODE_SINGLESHOT	Selects single-shot trigger mode. This will trigger when the threshold is detected, and then turn the trigger off. The trigger must be re-enabled using FFTP_TriggerOn.
FFTP_TRIGGERMODE_GENWAVETABLE	Selects the generator wavetable trigger mode. This is only functional when the Signal Generator function (SG_ChAFunction or SG_ChBFunction) is a wavetable, and the sample collection will be triggered when the wavetable wraps. NB: This option is not available when the analyzer sample rate is 192kHz.

5.3.3.1.15 FFTP_TriggerPoint

Description

This property allows selection of the point in the sample buffer at which the FFT will trigger.

Values

The trigger point can be a specified number of samples between 0 and one less than the number of FFT points (for example, at an FFT size of 4k, the trigger point can be between 0 and 4095).

Alternatively, one of the following constants can be used:

FFTP_TRIGGERPOINT_START	The sample buffer collected will have the trigger point at the start of the buffer.
FFTP_TRIGGERPOINT_QUARTER	The sample buffer collected will have the trigger point a quarter of the way through.
FFTP_TRIGGERPOINT_HALF	The sample buffer collected will have the trigger point half-way through.
FFTP_TRIGGERPOINT_THREEQUARTER	The sample buffer collected will have the trigger point three quarters of the way through.
FFTP_TRIGGERPOINT_END	The sample buffer collected will have the trigger point at the end.

For example, if the number of FFT points (FFTP_NumPoints) is set at 4k, and the trigger point is set to **FFTP_TRIGGERPOINT_QUARTER**, then the buffer will contain the 1k samples before the trigger point, then the point which triggered the buffer collection, followed by the next 3k-1 samples collected.

5.3.3.1.16 FFTP_ThresholdMode

Description

This property sets the threshold mode of the FFT trigger, i.e. which way the signal should be going to trigger data capture.

Values

FFTP_THRESHOLDMODE_EQUALTO	Sets the trigger to capture when the sample value is equal to the threshold value.
FFTP_THRESHOLDMODE_NEG	Sets the trigger to capture when the signal is negative-going, through the threshold value.
FFTP_THRESHOLDMODE_POS	Sets the trigger to capture when the signal is positive-going, through the threshold value.
FFTP_THRESHOLDMODE_NOTEQUALTO	Sets the trigger to capture when the sample value is not equal to the threshold value.

5.3.3.1.17 FFTP_Threshold

Description

This property allows specification of the threshold value at which the trigger should start capturing the sample buffer.

The value must be specified in the unit selected by FFTP_ThresholdUnit.

Note that if a threshold is entered in a non-logarithmic unit (e.g. Volts) then it must be entered as a positive voltage, and the polarity of the signal must be specified using FFTP_ThresholdPolarity.

Values

The FFT trigger threshold is represented as a double-precision floating point value.

5.3.3.1.18 FFTP_ThresholdUnit

Description

This property allows selection of the unit for the FFT trigger threshold, as specified using FFTP_Threshold.

Values

UNIT_DBFS	Sets trigger threshold amplitude unit to dBFS.
UNIT_PERCENTFS	Sets trigger threshold amplitude unit to %FS (percentage of full scale).
UNIT_FFS	Sets trigger threshold amplitude unit to FFS (fraction of full scale).
UNIT_HEX	Sets trigger threshold amplitude unit to Hex.
UNIT_VRMS	Sets trigger threshold amplitude unit to an RMS voltage.
UNIT_VP	Sets trigger threshold amplitude unit to a peak voltage.
UNIT_VPP	Sets trigger threshold amplitude unit to a peak-to-peak voltage.
UNIT_DBU	Sets trigger threshold amplitude unit to dBu.
UNIT_DBV	Sets trigger threshold amplitude unit to dBV.
UNIT_DBM	Sets trigger threshold amplitude unit to dBm.
UNIT_W	Sets trigger threshold amplitude unit to W.
UNIT_DBSPL	Sets trigger threshold amplitude unit to dBSPL.

NB: **The FFT trigger checks actual sample values. If the trigger threshold unit is specified as an RMS voltage, then the dScope will assume that the incoming signal is a sine wave for purposes of conversion between RMS and peak values.**

5.3.3.1.19 FFTP_ThresholdPolarity

Description

This property allows selection of the signal's polarity at the trigger point.

Values

FFTP_THRESHOLDPOLARITY_NEG	Specifies that the trigger threshold is negative.
FFTP_THRESHOLDPOLARITY_POS	Specifies that the trigger threshold is positive.

5.3.3.1.20 FFTP_TriggerChannel

Description

This property allows selection of which channel the FFT buffer aquisition should trigger on.

Values

FFTP_TRIGGERCHANNEL_A	Specifies that sample collection should be triggered when channel A meets the specified trigger threshold. Channel B will be triggered at the same time, regardless of the signal.
FFTP_TRIGGERCHANNEL_B	Specifies that sample collection should be triggered when channel B meets the specified trigger threshold. Channel A will be triggered at the same time, regardless of the signal.
FFTP_TRIGGERCHANNEL_INDEPENDENT	Specifies that sample collection will begin independently for each channel, when the signal on each channel meets the specified trigger threshold. No synchronisation of sample collection between channels will occur.

5.3.3.1.21 FFTP_TriggerOnCTDetector

Description

This property allows you to specify that the trigger should work on the signal buffer *after* it has been through the Continuous-Time Detector, rather than the incoming sample buffer. This is useful for catching glitches in the signal which show up on the Continuous-Time Detector but not the Signal Analyzer.

Values

True	Specifies that buffer capture should be triggered by the output from the Continuous-Time Detector.
False	Specifies that buffer capture should be triggered by the input signal.

5.3.3.1.22 FFTP_TriggerOn

Description

This property is used to turn the FFT trigger on and off.

Values

True	Turns the trigger on.
False	Turns the trigger off.

5.3.3.1.23 FFTP_BuffersProcessed

Description

This property is used to determine whether processing of FFT buffers has finished. This property can be used instead of the FFT Buffer Processed event, if the Event Manager is not available for the current Model Number.

To use this from a script, reset this property to **False** before triggering the FFT. Then, you can use the following code to wait until the FFT buffer has been processed:

```
While Not FFTParameters.FFTP_BuffersProcessed
    Sleep(1)
Wend
```

Values

True	This property is set to True once all FFT buffer processing is done on both channels.
False	Used to reset this property before triggering the FFT.

5.3.3.1.24 FFTP_CalcPhaseInfo

Description

This property can be set or read via automation only (it is not available from the dScope's user interface). It is used to specify that phases as well as magnitudes should be calculated from the FFT.

Once this property has been set, the **FFTD_BUFFER_PHASE** parameter can be used as a parameter to the FFT Detector functions that retrieve buffer information (FFTD_GetBufferSize, FFTD_GetBufferValueAt, FFTD_GetBufferHighestAmplToneBin, FFTD_GetBufferLowestAmplToneBin and FFTD_GetBuffer).

Values

True	Specifies that phase information should be calculated from the FFT.
False	Specifies that phase information should not be calculated from the FFT.

5.3.3.2 Methods

5.3.3.2.1 FFTP_GetWindowSpread

FFTP_GetWindowSpread()

This method can be used to return the number of FFT bins that the tone will spread into when using the currently selected Window function (See FFTP_WindowFunction). This can be used, for example, from FFT Detector Calculation Scripts to ensure that the entire spreading of a tone is taken into account when summing bins that constitute the tone.

Parameters

This method has no parameters.

Return value

This method returns the number of bins of window spread, as a short integer.

5.3.3.2.2 FFTP_ExportSampleBuffer

bRet = FFTP_ExportSampleBuffer(strFileName, sChannel, bCTBuffer)

This method can be used to export the current sample buffer (as captured by the FFT trigger) to a WAV-compatible file.

NB: **If an impulse response is currently being displayed (i.e. ImpulseResponse. IR_ImpulseResponse is set to True), then the buffer exported will be the impulse response and not the sample buffer.**

Parameters

strFileName The file name to export to. Any valid filename of a Windows WAV file (*.wav) is allowed.

If a full path name is specified, the system will save the file as specified.
If a file name only is specified, then the system will save the file in the folder specified in the Options dialogue box for Sample buffers (See OPT_SampleBuffersFolder).

If a file extension is not specified, the system will automatically append a file extension of ".wav" (Windows WAV file).

sChannel The channel to export the sample buffer for (**CHANNEL_A** or **CHANNEL_B**).

bCTBuffer **True** to export the residual Continuous-Time buffer, or **False** to export the sample buffer.
NB: To export the Continuous-Time Detector buffer, it must be currently being captured; this will happen if the Trace window is showing the Trace of this residual buffer.

Return value

This method returns **True** if the buffer was exported successfully, or **False** otherwise.

5.3.3.2.3 FFTP_ImportSampleBuffer

bRet = FFTP_ImportSampleBuffer(strFileName, sChannel, bCTBuffer, sOptions)

This method can be used to import a sample buffer from a previously exported dScope sample buffer (See FFTP_ExportSampleBuffer) or a Windows WAV file.

NB: **If the buffer loaded was originally saved as an impulse response, then this impulse response buffer (and not the raw sample buffer) will be imported. Please note that under these conditions, turning on the impulse response (by setting ImpulseResponse.IR_ImpulseResponse to True) will result in strange data in the sample buffer!)**

Parameters

strFileName	The file name to import from. Any valid filename of a Windows WAV file (*.wav) is allowed. If a full path name is specified, the system will load the file as specified. If a file name only is specified, then the system will look for the file in the folder specified in the Options dialogue box for Sample buffers (See OPT_SampleBuffersFolder). If a file extension is not specified, the system will automatically append a file extension of ".wav" (Windows WAV file).
sChannel	The channel to import the sample buffer into (**CHANNEL_A** or **CHANNEL_B**). Note that if the specified channel is not currently being analyzed (i.e. the Signal Analyzer channel selection SA_Channel is set to the opposite channel), then you will not see the imported buffer.
bCTBuffer	**True** to import the buffer into the residual Continuous-Time buffer, or **False** to import into the sample buffer. NB: To import the Continuous-Time Detector buffer, it must be currently being captured; this will happen if the Trace window is showing the Trace of this residual buffer.
sOptions	Options describing how to treat the buffer, if it does not fit exactly into a dScope buffer size (of 2^N samples). It can be one of the values listed under Options, below.

Options

The following values are valid for the ***sOptions*** parameter. Note that these are ignored if the imported buffer is the correct size for a dScope buffer (2^N samples).

FFTP_IMPORTSAMPLEBUFFER_TRIM	Trims the buffer to the next smaller buffer size of 2^N samples. Extra samples are discarded.
FFTP_IMPORTSAMPLEBUFFER_EXPAND	Expands the buffer to the next larger buffer size of 2^N samples. The extra space is filled by repeating samples from the start of the buffer until the buffer is full.
FFTP_IMPORTSAMPLEBUFFER_EXPAND_ZEROPAD	Expands the buffer to the next larger buffer size of 2^N samples. The extra space is filled with zero samples.
FFTP_IMPORTSAMPLEBUFFER_AUTOADJUST	Either expands or trims the buffer to the nearest buffer size of 2^N samples. If this buffer size is larger than the size of the WAV file, the extra space is filled by repeating samples from the start of the buffer until the buffer is full.
FFTP_IMPORTSAMPLEBUFFER_AUTOADJUST_ZEROPAD	Either expands or trims the buffer to the nearest buffer size of 2^N samples. If this buffer size is larger than the size of the WAV file, the extra space is filled with zero samples.

Return value

This method returns **True** if the buffer was imported successfully, or **False** otherwise. If imported successfully, the FFT trigger will be turned off to prevent further data captures overwriting the imported buffer.

5.3.4 Impulse Response Parameters

The Impulse Response Parameters section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"ImpulseResponse."**

Properties

IR_ImpulseResponse
IR_ImpulseRelativity
IR_NormalizeImpulse
IR_GeneratedRangeOnly
IR_WindowFunction
IR_HalfWindow
IR_StartWindowChA
IR_MidWindowChA
IR_EndWindowChA
IR_StartWindowChB
IR_MidWindowChB
IR_EndWindowChB
IR_WindowUnit
IR_WindowTied
IR_ApplyWindow

Methods

IR_SetImpulseWindowChA
IR_SetImpulseWindowChB

5.3.4.1 Properties

5.3.4.1.1 IR_ImpulseResponse

Description

This property turns on creation of an impulse response in the Sample buffer. The sample buffer is effectively replaced by its impulse response, meaning that FFTs performed will now be FFTs of the impulse response, and the Scope Trace on the Trace window will show the impulse response.

Values

True	Turns on creation of impulse response.
False	Turns off creation of impulse response, reverting to the incoming sample buffer.

5.3.4.1.2 IR_ImpulseRelativity

Description

This property allows selection of whether to create the impulse response relative to the generated data, or relative to the incoming data from the other channel.

Values

IR_IMPULSERELATIVITY_GEN	Creates the impulse response by comparing the incoming sample buffer's FFT with the FFT of the generated data.
IR_IMPULSERELATIVITY_CHANNEL	Creates the impulse response by comparing the incoming sample buffer's FFT with the FFT of the incoming data on the other channel. NB: This option is not available when the Signal Analyzer is set to analyze both channels (SA_Channel).

5.3.4.1.3 IR_ImpulseAbsolute

Description

This property specifies that the impulse response level should be adjusted so that it is shown as an absolute level, rather than relative to the reference data (either from the Generator, or the other channel's input - see IR_ImpulseRelativity).

Values

True	Adjusts level of impulse response to be at the absolute level of the input signal.
False	Leaves results of the impulse response as a relative (gain) level.

5.3.4.1.4 IR_NormalizeImpulse

Description

This property is used to select whether to normalize the impulse response to the trigger point in the buffer. Subsequent actions on the impulse response buffer will act as if the impulse actually occurred immediately at the trigger point.

Values

True	Normalize the impulse response to the trigger point.
False	Do not normalize the impulse response.

5.3.4.1.5 IR_GeneratedRangeOnly

Description

This property is used to select whether to remove frequencies from the impulse response that were not generated by the dScope's Signal Generator.

This is relevant to the Bin centres waveform only, where certain bins of the frequency response contain almost no signal (those bins corresponding to frequencies outside the generated range). When these bins are compared to the reference signal, the large differences between the bins of the input and reference causes a noise frequency response, which shows up in the impulse response.

To avoid this noise, the dScope can ignore any bins that should not contain any signal at all (according to the generated waveform) and ensure that they are not included in the impulse response.

Values

True	Limit the frequencies included in the impulse response to those generated by the dScope's Signal Generator.
False	Include all frequencies in the impulse response.

5.3.4.1.6 IR_WindowFunction

Description

This property allows specification of the Window function to use to retrieve frequency response data (via an FFT) from an impulse response.

NB: **This property is tied across both channels, regardless of whether the Window function for the impulse response is set to be tied (IR_WindowTied).**

Values

FFTP_WINDOW_RECTANGULAR	Selects the Window function for the impulse response to be rectangular. Note that this is the same as having no Window function at all, and should not be used except in circumstances where the signal period is exactly the same as the sample buffer length.
FFTP_WINDOW_TRIANGULAR	Selects the Window function for the impulse response to be triangular.
FFTP_WINDOW_BLACKMAN	Selects the Blackman Window function for the impulse response.
FFTP_WINDOW_HANN	Selects the Hann Window function for the impulse response.
FFTP_WINDOW_HAMMING	Selects the Hamming Window function for the impulse response.
FFTP_WINDOW_BH4	Selects the Blackman-Harris 4 Window function for the impulse response.
FFTP_WINDOW_GAUSSIAN	Selects a gaussian Window function for the impulse response.
FFTP_WINDOW_FLATTOP	Selects the Prism Sound flat-top Window function for the impulse response. Provides a Window function with very little leakage of signal from the bin containing a tone into adjacent bins.

FFTP_WINDOW_PRISM5	Selects the Prism Sound 5-term Window function for the impulse response. Provides optimal frequency resolution with minimum spreading of tone frequencies into adjacent bins.
FFTP_WINDOW_PRISM6	Selects the Prism Sound 6-term Window function for the impulse response.
FFTP_WINDOW_PRISM7	Selects the Prism Sound 7-term Window function for the impulse response. Gives the best dynamic range at the expense of a little spreading of tone frequencies into adjacent bins.

5.3.4.1.7 IR_HalfWindow

Description

This property is used to specify that the Window function to use for the impulse response should be half a window, i.e. the left hand side of the window should be vertical, and only the right half will be windowed. This allows the Window function to be applied immediately before the impulse, and the windowing to apply for as much of the buffer as required after the impulse.

NB: **This property is tied across both channels, regardless of whether the Window function for the impulse response is set to be tied (IR_WindowTied).**

Values

True	Use a half Window function for the impulse response.
False	Use a full Window function for the impulse response.

5.3.4.1.8 IR_StartWindowChA

Description

This property can be used to set the start position in the channel A buffer of the Window function for the impulse response specified by IR_WindowFunction. It is entered in the unit specified by IR_WindowUnit.

If the Window function is set up to be a half window (See IR_HalfWindow), then the middle position of the Window function (IR_MidWindowChA) will be set to the same value as this one (because a half window has a vertical edge at the left). Otherwise, the middle position will be shifted to be half way between the current start and end positions.

If the Window function is set up to be tied across both channels (See IR_WindowTied), then changing this property will also change IR_StartWindowChB.

Values

This can be any number from 0 to two less than the number of FFT points (See FFTP_NumPoints). This may be restricted by the current values of IR_MidWindowChA and IR_EndWindowChA.

NB: **The value may be relative to the Trigger point (FFTP_TriggerPoint) or relative to the start of the buffer, depending on an option in the Options Settings (OPT_TriggerPointRelative).**

5.3.4.1.9 IR_MidWindowChA

Description

This property can be used to set the middle position in the channel A buffer of the Window function for the impulse response specified by IR_WindowFunction. It is entered in the unit specified by IR_WindowUnit.

If the Window function is set up to be a half window (See IR_HalfWindow), then the start position of the Window function (IR_StartWindowChA) will be set to the same value as this one (because a half window has a vertical edge at the left). Otherwise, the start and end positions will both be altered by the same distance as the middle bin is being altered by.

If the Window function is set up to be tied across both channels (See IR_WindowTied), then changing this property will also change IR_EndWindowChB.

Values

This can be any number from 0 to two less than the number of FFT points (See FFTP_NumPoints). This may be restricted by the current values of IR_StartWindowChA and IR_EndWindowChA.

NB: **The value may be relative to the Trigger point (FFTP_TriggerPoint) or relative to the start of the buffer, depending on an option in the Options Settings (OPT_TriggerPointRelative).**

5.3.4.1.10 IR_EndWindowChA

Description

This property can be used to set the end position in the channel A buffer of the Window function for the impulse response specified by IR_WindowFunction. It is entered in the unit specified by IR_WindowUnit.

If the Window function is not set up to be a half window (See IR_HalfWindow), then the middle position will be shifted to be half way between the current start and end positions.

If the Window function is set up to be tied across both channels (See IR_WindowTied), then changing this property will also change IR_EndWindowChB.

Values

This can be any number from 0 to two less than the number of FFT points (See FFTP_NumPoints). This may be restricted by the current values of IR_StartWindowChA and IR_MidWindowChA.

NB: **The value may be relative to the Trigger point (FFTP_TriggerPoint) or relative to the start of the buffer, depending on an option in the Options Settings (OPT_TriggerPointRelative).**

5.3.4.1.11 IR_StartWindowChB

Description

This property can be used to set the start position in the channel B buffer of the Window function for the impulse response specified by IR_WindowFunction. It is entered in the unit specified by

IR_WindowUnit.

If the Window function is set up to be a half window (See IR_HalfWindow), then the middle position of the Window function (IR_MidWindowChB) will be set to the same value as this one (because a half window has a vertical edge at the left). Otherwise, the middle position will be shifted to be half way between the current start and end positions.

If the Window function is set up to be tied across both channels (See IR_WindowTied), then changing this property will also change IR_StartWindowChA.

Values

This can be any number from 0 to two less than the number of FFT points (See FFTP_NumPoints). This may be restricted by the current values of IR_MidWindowChB and IR_EndWindowChB.

NB: **The value may be relative to the Trigger point (FFTP_TriggerPoint) or relative to the start of the buffer, depending on an option in the Options Settings (OPT_TriggerPointRelative).**

5.3.4.1.12 IR_MidWindowChB

Description

This property can be used to set the middle position in the channel B buffer of the Window function for the impulse response specified by IR_WindowFunction. It is entered in the unit specified by IR_WindowUnit.

If the Window function is set up to be a half window (See IR_HalfWindow), then the start position of the Window function (IR_StartWindowChB) will be set to the same value as this one (because a half window has a vertical edge at the left). Otherwise, the start and end positions will both be altered by the same distance as the middle bin is being altered by.

If the Window function is set up to be tied across both channels (See IR_WindowTied), then changing this property will also change IR_EndWindowChA.

Values

This can be any number from 0 to two less than the number of FFT points (See FFTP_NumPoints). This may be restricted by the current values of IR_StartWindowChB and IR_EndWindowChB.

NB: **The value may be relative to the Trigger point (FFTP_TriggerPoint) or relative to the start of the buffer, depending on an option in the Options Settings (OPT_TriggerPointRelative).**

5.3.4.1.13 IR_EndWindowChB

Description

This property can be used to set the end position in the channel B buffer of the Window function for the impulse response specified by IR_WindowFunction. It is entered in the unit specified by IR_WindowUnit.

If the Window function is not set up to be a half window (See IR_HalfWindow), then the middle position will be shifted to be half way between the current start and end positions.

If the Window function is set up to be tied across both channels (See IR_WindowTied), then changing this property will also change IR_EndWindowChA.

Values

This can be any number from 0 to two less than the number of FFT points (See FFTP_NumPoints). This may be restricted by the current values of IR_StartWindowChB and IR_MidWindowChB.

NB: **The value may be relative to the Trigger point (FFTP_TriggerPoint) or relative to the start of the buffer, depending on an option in the Options Settings (OPT_TriggerPointRelative).**

5.3.4.1.14 IR_WindowUnit

Description

This property allows selection of the unit for entry of positions of the start, middle and end positions of the Window function for the impulse response (See IR_StartWindowChA / IR_StartWindowChB, IR_MidWindowChA / IR_MidWindowChB and IR_EndWindowChA / IR_EndWindowChB.

Values

UNIT_MS	Sets unit for entry of Window function positons to ms.
UNIT_SAMPLES	Sets unit for entry of Window function positons to samples.

NB: **The fields entered in this unit may be shown relative to the Trigger point (FFTP_TriggerPoint) or relative to the start of the buffer, depending on an option in the Options Settings (OPT_TriggerPointRelative).**

5.3.4.1.15 IR_WindowTied

Description

This property specifies that the Window function for the impulse response is tied together across both channels. Selecting this option will force the window to remain the same on both channels, regardless of which channel is changed.

If this property is set to **False**, then the Window Function (IR_WindowFunction) and Half window (IR_HalfWindow) properties will still be shared across both channels, but the start, middle and end bins will vary independently.

Values

True	Ties the Window function across both channels.
False	Allows the start, middle and end positions of the Window function to vary between channel A and channel B.

5.3.4.1.16 IR_ApplyWindow

Description

This property allows you to specify that the Window function selected for the impulse response should be applied always, regardless of whether an impulse response is currently created.

This allows you to apply a Window function to any signal, and be able to edit the start and end points of the Window function, as well as whether it is a half-window or not.

Values

IR_APPLYWINDOW_IMPULSE	Only apply the Window function for the impulse response when an impulse response has been created in the sample buffer.
IR_APPLYWINDOW_ALWAYS	Apply the Window function for the impulse response always, regardless of whether an impulse response is currently being created.

5.3.4.2 Methods

5.3.4.2.1 IR_SetImpulseWindowChA

bRet = IR_SetImpulseWindowChA(dStart, dEnd, sWindow, bHalfWindow)

This method can be used to set all details of the Window function for the impulse response on channel A.

Parameters

dStart	The start of the window function, in the units specified by IR_WindowUnit.
dEnd	The end of the window function, in the units specified by IR_WindowUnit.
sWindow	The Window function to use. See Window functions for a full list of values.
bHalfWindow	**True** to set a half window, i.e. the left hand side of the window should be vertical, and only the right half will be windowed. This allows the Window function to be applied immediately before the impulse, and the windowing to apply for as much of the buffer as required after the impulse.

Return value

This method returns **True** if the impulse response Window function was set successfully, or **False** otherwise.

Window functions

FFTP_WINDOW_RECTANGULAR	Selects the Window function for the impulse response to be rectangular. Note that this is the same as having no Window function at all, and should not be used except in circumstances where the signal period is exactly the same as the sample buffer length.
FFTP_WINDOW_TRIANGULAR	Selects the Window function for the impulse response to be triangular.
FFTP_WINDOW_BLACKMAN	Selects the Blackman Window function for the impulse

	response.
FFTP_WINDOW_HANN	Selects the Hann Window function for the impulse response.
FFTP_WINDOW_HAMMING	Selects the Hamming Window function for the impulse response.
FFTP_WINDOW_BH4	Selects the Blackman-Harris 4 Window function for the impulse response.
FFTP_WINDOW_GAUSSIAN	Selects a gaussian Window function for the impulse response.
FFTP_WINDOW_FLATTOP	Selects the Prism Sound flat-top Window function for the impulse response. Provides a Window function with very little leakage of signal from the bin containing a tone into adjacent bins.
FFTP_WINDOW_PRISM5	Selects the Prism Sound 5-term Window function for the impulse response. Provides optimal frequency resolution with minimum spreading of tone frequencies into adjacent bins.
FFTP_WINDOW_PRISM6	Selects the Prism Sound 6-term Window function for the impulse response.
FFTP_WINDOW_PRISM7	Selects the Prism Sound 7-term Window function for the impulse response. Gives the best dynamic range at the expense of a little spreading of tone frequencies into adjacent bins.

5.3.4.2.2 IR_SetImpulseWindowChB

bRet = IR_SetImpulseWindowChB(dStart, dEnd, sWindow, bHalfWindow)

This method can be used to set all details of the Window function for the impulse response on channel B.

Parameters

dStart	The start of the window function, in the units specified by IR_WindowUnit.
dEnd	The end of the window function, in the units specified by IR_WindowUnit.
sWindow	The Window function to use. See Window functions for a full list of values.
bHalfWindow	**True** to set a half window, i.e. the left hand side of the window should be vertical, and only the right half will be windowed. This allows the Window function to be applied immediately before the impulse, and the windowing to apply for as much of the buffer as required after the impulse.

Return value

This method returns **True** if the impulse response Window function was set successfully, or **False** otherwise.

Window functions

FFTP_WINDOW_RECTANGULAR	Selects the Window function for the impulse response to be rectangular. Note that this is the same as having no Window function at all, and should not be used except in circumstances where the signal period is exactly the same as the sample buffer length.
FFTP_WINDOW_TRIANGULAR	Selects the Window function for the impulse response to be

	triangular.
FFTP_WINDOW_BLACKMAN	Selects the Blackman Window function for the impulse response.
FFTP_WINDOW_HANN	Selects the Hann Window function for the impulse response.
FFTP_WINDOW_HAMMING	Selects the Hamming Window function for the impulse response.
FFTP_WINDOW_BH4	Selects the Blackman-Harris 4 Window function for the impulse response.
FFTP_WINDOW_GAUSSIAN	Selects a gaussian Window function for the impulse response.
FFTP_WINDOW_FLATTOP	Selects the Prism Sound flat-top Window function for the impulse response. Provides a Window function with very little leakage of signal from the bin containing a tone into adjacent bins.
FFTP_WINDOW_PRISM5	Selects the Prism Sound 5-term Window function for the impulse response. Provides optimal frequency resolution with minimum spreading of tone frequencies into adjacent bins.
FFTP_WINDOW_PRISM6	Selects the Prism Sound 6-term Window function for the impulse response.
FFTP_WINDOW_PRISM7	Selects the Prism Sound 7-term Window function for the impulse response. Gives the best dynamic range at the expense of a little spreading of tone frequencies into adjacent bins.

5.3.5 Continuous-Time Detector

The Continuous-Time Detector section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"CTDetector."**

Properties

CTD_ChA
CTD_ChB
CTD_Unit
CTD_Function
CTD_BPBRMode
CTD_BPBRBandwidth
CTD_BPBRFreqMode
CTD_BPBRFreq
CTD_HPFilter
CTD_HPFilterFreq
CTD_LPFilter
CTD_LPFilterFreq
CTD_WeightingFilter
CTD_Response
CTD_Relativity

Methods

There are no methods available to control the Continuous-Time Detector.

5.3.5.1 Properties

5.3.5.1.1 CTD_ChA

Description

This **read-only** property represents the current value of the Continuous-Time Detector, channel A.

The value is returned in the current unit, as selected by CTD_Unit.

Values

The CT Detector channel A value is represented as a double-precision floating point value.

5.3.5.1.2 CTD_ChB

Description

This **read-only** property represents the current value of the Continuous-Time Detector, channel B.

The value is returned in the current unit, as selected by CTD_Unit.

Values

The CT Detector channel B value is represented as a double-precision floating point value.

5.3.5.1.3 CTD_Unit

Description

This property allows selection of the unit for measurement of the Continuous-Time Detector values on channel A and B.

This specifies which unit the values returned by CTD_ChA and CTD_ChB will be returned in.

Values

Under digital and normal analogue analysis, the available units will depend on the relativity of the CT Detector (See CTD_Relativity).

If the CT Detector relativity is set to **absolute**, then the following values are allowed:

UNIT_DBFS	Sets CT Detector unit to dBFS.
UNIT_PERCENTFS	Sets CT Detector unit to %FS (percentage of full scale).
UNIT_FFS	Sets CT Detector unit to FFS (fraction of full scale).
UNIT_HEX	Sets CT Detector unit to Hex.
UNIT_V	Sets CT Detector unit to V.
UNIT_DBU	Sets CT Detector unit to dBu.
UNIT_DBV	Sets CT Detector unit to dBV.
UNIT_DBM	Sets CT Detector unit to dBm.
UNIT_W	Sets CT Detector unit to W.
UNIT_DBSPL	Sets CT Detector unit to dBSPL.

UNIT_DBR	Sets CT Detector unit to dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Sets CT Detector unit to percentage of the reference amplitude (SA_RefAmpl).

If the CT Detector is set to be **self-relative**, **generator-relative** or **channel-relative**, then the following values are allowed :

UNIT_RELATIVE_DB	Sets CT Detector unit to dB, relative to the specified value.
UNIT_RELATIVE_PERCENT	Sets CT Detector unit to a percentage of the specified value.
UNIT_RELATIVE_GAIN	Sets CT Detector unit to a gain factor, relative to the specified value.
UNIT_RELATIVE_ANAVSGEN	Sets CT Detector unit to a special unit relating the input level to a set level on the Signal Generator. For example, if the Signal Analyzer unit (SA_RMSAmplUnit) is set to dB SPL and the Signal Generator unit (SG_ChAAmplUnit) is set to V (RMS), then this unit will show dB SPL / 1V (RMS), i.e. the input level in dB SPL that corresponds to 1V (RMS). NB: This unit is only available if the Detector's relativity (CTD_Relativity) is set to generator-relative or channel-relative.

If the analyzer is currently set up to analyze the demodulated jitter signal through the Analogue Inputs (See AI_Source for further details), then the following values are allowed.

UNIT_JITTER_NS	Sets CT Detector unit for jitter values to ns.
UNIT_JITTER_UI	Sets CT Detector unit for jitter values to UI.

5.3.5.1.4 CTD_Function

Description

This property allows selection of the function used on the Continuous-Time Detector.

NB:

The selected function is actually a script (See Detector Functions). Its sole purpose is to set up the rest of the Continuous-Time Detector fields.

After setting the function, any of the fields of the Detector can still be altered, so it's possible to end up with a Detector whose settings are completely different from the function that it purports to be!

If settings are altered, the dScope's user interface will show an asterisk (*) in the title bar of the Detector, to indicate that it has changed from the default.

During a dScope session, any changes to the settings of a particular function will be remembered (if set using the Options setting OPT_RememberDetectorDetails).

Values

The following functions are the default scripts supplied with the dScope software. You can create your own scripts to add to this list by creating a Detector function script in the correct folder (see Detector Functions).

"Amplitude"	Sets up the CT Detector details for amplitude measurement.
"Balance"	Sets up the CT Detector details for measurement of inter-channel balance.
"Band pass"	Sets up the CT Detector details for band pass measurement.

"Band reject"	Sets up the CT Detector details for band reject measurement.
"Cross-talk"	Sets up the CT Detector details for cross-talk measurement. Note that this measurment assumes that the analyzed signal originated with the dScope Signal Generator, as it tracks the generated signal on the opposite channel. For this to work successfully, one of the Generator channels must be turned off, the generated frequencies must be different on each channel.
"Gain"	Sets up the CT Detector details for measurement of gain with respect to the generator.
"IMD CCIF"	Sets up the CT Detector details for measurement of IMD according to the CCIF standard ("difference-tone" IMD).
"IMD SMPTE-DIN"	Sets up the CT Detector details for measurement of IMD according to the SMPTE-DIN standard (IMD side-bands).
"Noise (unweighted)"	Sets up the CT Detector details for RMS measurement of a noise signal, with no weighting.
"Noise (A-weighted)"	Sets up the CT Detector details for RMS measurement of a noise signal, with A-weighting.
"Noise (CCIR-468)"	Sets up the CT Detector details for RMS measurement of a noise signal, with CCIR-468 weighting (unity gain at 1kHz).
"THD+N - absolute"	Sets up the CT Detector details for measurement of total harmonic distortion and noise, in absolute units.
"THD+N - relative"	Sets up the CT Detector details for measurement of total harmonic distortion and noise, relative to the RMS amplitude of the analyzed signal.

5.3.5.1.5 CTD_BPBRMode

Description

This property allows selection of the band pass or band reject filter for the Continuous-Time Detector.

Values

CTD_BPBRMODE_OFF	Selects the Continuous-Time Detector to have no band pass or band reject filter.
CTD_BPBRMODE_BP	Selects the Continuous-Time Detector to have a band pass filter with the settings described in CTD_BPBRBandwidth, CTD_BPBRFreqMode, and CTD_BPBRFreq.
CTD_BPBRMODE_BR	Selects the Continuous-Time Detector to have a band reject filter with the settings described in CTD_BPBRBandwidth, CTD_BPBRFreqMode, and CTD_BPBRFreq.
CTD_BPBRMODE_IMD	This is a special case which disables all other fields on the Continuous-Time Detector. It puts the Detector into SMPTE-DIN IMD demodulation mode, which measures the heights of the side bands around the highest frequency of two tones.

NB:

Note that the CTD_BPBRMODE_IMD option requires the generator to be set up with a valid signal for IMD side-band analysis, since the tracked frequency is taken from the highest frequency of a twin-tone.

5.3.5.1.6 CTD_BPBRBandwidth

Description

This property allows selection of the bandwidth of the band pass or band reject filter for the Continuous-Time Detector.

This property will have no effect unless the CTD_BPBRMode property is set to band pass (**CTD_BPBRMODE_BP**) or band reject (**CTD_BPBRMODE_BR).**

Values

CTD_BPBRBANDWIDTH_3	Selects the bandwidth for the band pass or band reject filter to be 1/3 octave.
CTD_BPBRBANDWIDTH_6	Selects the bandwidth for the band pass or band reject filter to be 1/6 octave.
CTD_BPBRBANDWIDTH_12	Selects the bandwidth for the band pass or band reject filter to be 1/12 octave.
CTD_BPBRBANDWIDTH_24	Selects the bandwidth for the band pass or band reject filter to be 1/24 octave.

5.3.5.1.7 CTD_BPBRFreqMode

Description

This property allows selection of the frequency details of the band pass or band reject filter for the Continuous-Time Detector.

This determines how the dScope sets the frequency of the filter.

This property will have no effect unless the CTD_BPBRMode property is set to band pass (**CTD_BPBRMODE_BP**) or band reject (**CTD_BPBRMODE_BR).**

Values

CTD_BPBRFREQMODE_INPUT	Selects the frequency for the band pass or band reject filter to track the channel's input frequency.
CTD_BPBRFREQMODE_GEN	Selects the frequency for the band pass or band reject filter to track the generator frequency for each channel.
CTD_BPBRFREQMODE_GENOTHER	Selects the frequency for the band pass or band reject filter to track the generator frequency of the *other* channel. This is useful for cross-talk measurement.
CTD_BPBRFREQMODE_FIXED	Selects the frequency for the band pass or band reject filter to be fixed, at the frequency specified using CTD_BPBRFreq.
CTD_BPBRFREQMODE_IMDDIFF	Selects the frequency for the band pass or band reject filter to track the IMD differential frequency.

NB: **Note that the CTD_BPBRFREQMODE_IMDDIFF option requires the generator to be set up with a valid signal for IMD difference-tone analysis.**

5.3.5.1.8 CTD_BPBRFreq

Description

This property allows selection of the frequency of the band pass or band reject filter for the Continuous-Time Detector.

The frequency is specified in Hz, and is only used if the frequency mode (CTD_BPBRFreqMode) is set to **CTD_BPBRFREQMODE_FIXED**.

Values

The BP/BR frequency is represented as a double-precision floating point value.

5.3.5.1.9 CTD_HPFilter

Description

This property allows selection of the high-pass filter for the Continuous-Time Detector.

Values

CTD_HP_OFF	Sets high-pass filter for the CT Detector to off (see note below).
CTD_HP_DCB	Sets high-pass filter for the CT Detector to a DC blocking filter.
CTD_HP_10HZ	Sets high-pass filter for the CT Detector to 10Hz.
CTD_HP_22HZ	Sets high-pass filter for the CT Detector to 22Hz.
CTD_HP_100HZ	Sets high-pass filter for the CT Detector to 100Hz.
CTD_HP_400HZ	Sets high-pass filter for the CT Detector to 400Hz.
CTD_HP_DEFAULT	Sets high-pass filter for the CT Detector to follow the default set up on the Signal Analyzer (see SA_DefaultHPFilter). This allows an easy way of switching filters for multiple Detectors at the same time.

NB:

1) The dScope analogue hardware has a built-in DC blocking filter, which can be disabled by fitting jumpers to the Analogue and Converter PCBs (see PCB jumper options in the Operation Manual for further details).
In analogue analysis mode, if the analogue hardware is not DC coupled, then selection of "CTD_HP_OFF" is disabled and the DC blocking filter is automatically selected (CTD_HP_DCB).

NB:

2) To set a specific frequency for the high-pass filter, use CTD_HPFilterFreq.

5.3.5.1.10 CTD_HPFilterFreq

Description

This property allows specification of the frequency of the high-pass filter for the Continuous-Time Detector.

Values

The high-pass filter frequency is represented as a long integer value.

NB: **If the high-pass filter is one of the predefined filters (See CTD_HPFilter), then this property will return the *actual* frequency that this represents.**
For example, if CTD_HPFilter has been set to CTD_HP_22HZ, then this property will return 22. DC Block (CTD_HP_DCB) and Off (CTD_HP_OFF) will both return zero.

5.3.5.1.11 CTD_LPFilter

Description

This property allows selection of the low-pass filter for the Continuous-Time Detector.

Values

CTD_LP_OFF	Sets low-pass filter for the CT Detector to off.
CTD_LP_20KHZ_AES17	Sets low-pass filter for the CT Detector to a 20kHz filter that matches the AES17 low-pass filter specification.
CTD_LP_22KHZ	Sets low-pass filter for the CT Detector to 22kHz.
CTD_LP_30KHZ	Sets low-pass filter for the CT Detector to 30kHz.
CTD_LP_40KHZ	Sets low-pass filter for the CT Detector to 40kHz.
CTD_LP_80KHZ	Sets low-pass filter for the CT Detector to 80kHz. NB: This option is ignored if the hardware is not 192kHz-capable.
CTD_LP_DEFAULT	Sets low-pass filter for the CT Detector to follow the default set up on the Signal Analyzer (see SA_DefaultLPFilter). This allows an easy way of switching filters for multiple Detectors at the same time.

NB: **To set a specific frequency for the low-pass filter, use CTD_LPFilterFreq**

5.3.5.1.12 CTD_LPFilterFreq

Description

This property allows specification of the frequency of the low-pass filter for the Continuous-Time Detector.

Values

The low-pass filter frequency is represented as a long integer value.

NB: **If the low-pass filter is one of the predefined filters (See CTD_LPFilter), then this property will return the *actual* frequency that this represents.**
For example, if CTD_LPFilter has been set to CTD_LP_40KHZ, then this property will return 40000. Off (CTD_LP_OFF) will return zero.

5.3.5.1.13 CTD_WeightingFilter

Description

This property allows selection of the weighting filter for the Continuous-Time Detector.

Values

CTD_WEIGHTING_NONE	Sets weighting filter for the CT Detector to none (not weighted).
CTD_WEIGHTING_AWEIGHTING	Sets weighting filter for the CT Detector to A-weighted.
CTD_WEIGHTING_CWEIGHTING	Sets weighting filter for the CT Detector to C-weighted.
CTD_WEIGHTING_CCIR468_1K	Sets weighting filter for the CT Detector to a CCIR-468 shape, with unity gain at 1kHz.
CTD_WEIGHTING_CCIR468_2K	Sets weighting filter for the CT Detector to a CCIR-468 shape, with unity gain at 2kHz.
CTD_WEIGHTING_DEFAULT	Sets weighting filter for the CT Detector to follow the default set up on the Signal Analyzer (see SA_DefaultWeightingFilter). This allows an easy way of switching filters for multiple Detectors at the same time.

5.3.5.1.14 CTD_Response

Description

This property allows selection of the response for the Continuous-Time Detector.

Values

CTD_RESPONSE_RMS	Sets the response for the CT Detector to measure the RMS value of the signal.
CTD_RESPONSE_PEAK	Sets the response for the CT Detector to measure the peak value of the signal.
CTD_RESPONSE_PEAKSAMPLE	Sets the response for the CT Detector to measure the peak sample value of the signal.
CTD_RESPONSE_QPEAK	Sets the response for the CT Detector to measure the Q–peak value of the signal.

5.3.5.1.15 CTD_Relativity

Description

This property allows selection of the relativity of the Continuous-Time Detector.

The CT Detector measurement can either be made in absolute units, or relative to a signal.

Values

CTD_RELATIVITY_ABS	Sets the CT Detector to display absolute Result values.
CTD_RELATIVITY_SELF	Sets the CT Detector to display Result values relative to the incoming signal, as read by the Signal Analyzer (See SA_ChARMSAmpl and SA_ChBRMSAmpl).

	Note that in IMD demodulation mode, this is actually relative to the amplitude of the higher frequency tone, and not the RMS total of the incoming signal.
CTD_RELATIVITY_GEN	Sets the CT Detector to display Result values relative to the generated amplitude for the same channel.
CTD_RELATIVITY_CHANNEL	Sets the CT Detector to display Result values relative to the RMS amplitude of the other channel.

NB:

The list of available units will change depending on the relativity. If CTD_RELATIVITY_ABS is selected, then a list of absolute units will be available. If any of the other relativities are selected, then only dB and % are available (See CTD_Unit for details of available units).

5.3.6 FFT Detector

The FFT Detector section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"FFTDetector."**

Note that before using an FFT Detector's properties and methods, the Detector must be set as the current Detector using Analyzer.SetFFTDetector. See Creating and accessing FFT Detectors for further details.

Properties

FFTD_ID
FFTD_ChA
FFTD_ChB
FFTD_Unit
FFTD_Function
FFTD_UserScript
FFTD_BPBRMode
FFTD_BPBRBandwidth
FFTD_BPBRFreqMode
FFTD_Harmonic
FFTD_BPBRFreq
FFTD_HPFilter
FFTD_HPFilterFreq
FFTD_BrickWallHPFilter
FFTD_LPFilter
FFTD_LPFilterFreq
FFTD_BrickWallLPFilter
FFTD_WeightingFilter
FFTD_UserWeightingFilter
FFTD_Relativity

Methods

All the methods for use with FFT Detectors are used to create user-defined FFT Detector Calculation scripts. Full details of these methods can be found in the FFT Detector Calculation scripts reference section.

5.3.6.1 Properties

5.3.6.1.1 FFTD_ID

Description

This **read-only** property returns the numerical ID of this FFT Detector.

Values

The FFT Detector ID is represented as a short integer value between 1 and 40.

5.3.6.1.2 FFTD_ChA

Description

This **read-only** property represents the current value of the FFT Detector, channel A.

The value is returned in the current unit, as selected by FFTD_Unit

NB:

The way that FFT Detector values are treated as settled works slightly differently from other Results.

Under normal conditions, if settling is being used from a script (See OPT_UseSettlingsFromScripts), then the script must wait for the specified number of Results to be obtained (within the specified tolerance) before the Result is returned (See Settling Parameters). For FFT Results, this will mean that a number of FFT calculations will be performed before a settled Result can be returned.

If the FFT trigger is not currently turned on, the dScope assumes that values read from an FFT Detector have *already* been settled (for example, the script may have averaged a number of FFT calculations already, and so the script assumes that a 'settled' FFT has already been obtained). Therefore, in the case of the trigger being turned off, the last FFTD_ChA value will be returned immediately without waiting for a further new value. Under these conditions, the LastResultSettled flag will be set to False.

Values

The FFT Detector channel A value is represented as a double-precision floating point value.

5.3.6.1.3 FFTD_ChB

Description

This **read-only** property represents the current value of the FFT Detector, channel B.

The value is returned in the current unit, as selected by FFTD_Unit

NB:

The way that FFT Detector values are treated as settled works slightly differently from other Results.

Under normal conditions, if settling is being used from a script (See OPT_UseSettlingsFromScripts), then the script must wait for the specified number of Results to be obtained (within the specified tolerance) before the Result is returned (See Settling Parameters). For FFT Results, this will mean that a number of FFT calculations will be performed before a settled Result can be returned.

If the FFT trigger is not currently turned on, the dScope assumes that values read from an FFT Detector have *already* been settled (for example, the script may have averaged a number of FFT calculations already, and so the script assumes that a 'settled' FFT has already been obtained). Therefore, in the case of the trigger being turned off, the last FFTD_ChB value will be returned immediately without waiting for a further new value. Under these conditions, the LastResultSettled flag will be set to False.

Values

The FFT Detector channel B value is represented as a double-precision floating point value.

5.3.6.1.4 FFTD_Unit

Description

This property allows selection of the unit for measurement of the FFT Detector values on channels A and B.

This specifies which unit the values returned by FFTD_ChA and FFTD_ChB will be returned in.

Values

Under digital and normal analogue analysis, the available units will depend on the relativity of the FFT Detector (See FFTD_Relativity).

If the FFT Detector relativity is set to **absolute**, then the following values are allowed:

UNIT_DBFS	Sets FFT Detector unit to dBFS.
UNIT_PERCENTFS	Sets FFT Detector unit to %FS (percentage of full scale).
UNIT_FFS	Sets FFT Detector unit to FFS (fraction of full scale).
UNIT_HEX	Sets FFT Detector unit to Hex.
UNIT_V	Sets FFT Detector unit to V.
UNIT_DBU	Sets FFT Detector unit to dBu.
UNIT_DBV	Sets FFT Detector unit to dBV.
UNIT_DBM	Sets FFT Detector unit to dBm.
UNIT_W	Sets FFT Detector unit to W.
UNIT_DBSPL	Sets FFT Detector unit to dBSPL.
UNIT_DBR	Sets FFT Detector unit to dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Sets FFT Detector unit to percentage of the reference amplitude (SA_RefAmpl).

If the FFT Detector is set to be **self-relative**, **generator-relative** or **channel-relative**, then the following values are allowed :

UNIT_RELATIVE_DB	Sets FFT Detector unit to dB, relative to the specified value.
UNIT_RELATIVE_PERCENT	Sets FFT Detector unit to a percentage of the specified value.
UNIT_RELATIVE_GAIN	Sets FFT Detector unit to a gain factor, relative to the specified value.
UNIT_RELATIVE_ANAVSGEN	Sets FFT Detector unit to a special unit relating the input level to a set level on the Signal Generator. For example, if the Signal Analyzer unit (SA_RMSAmplUnit) is set to dB SPL and the Signal Generator unit (SG_ChAAmplUnit) is set to V (RMS), then this unit will show dB SPL / 1V (RMS), i.e. the input level in dB SPL that corresponds to 1V (RMS). NB: This unit is only available if the Detector's relativity (FFTD_Relativity) is set to generator-relative or channel-relative.

If the analyzer is currently set up to analyze the demodulated jitter signal through the Analogue Inputs (See AI_Source for further details), then the following values are allowed.

UNIT_JITTER_NS	Sets FFT Detector unit for jitter values to ns.
UNIT_JITTER_UI	Sets FFT Detector unit for jitter values to UI.

5.3.6.1.5 FFTD_Function

Description

This property allows selection of the function used on the FFT Detector.

NB:

The selected function is actually a script (See Detector Functions). Its sole purpose is to set up the rest of the FFT Detector fields.

After setting the function, any of the fields of the Detector can still be altered, so it's possible to end up with a Detector whose settings are completely different from the function that it purports to be!

If settings are altered, the dScope's user interface will show an asterisk (*) in the title bar of the Detector, to indicate that it has changed from the default.

During a dScope session, any changes to the settings of a particular function will be remembered (if set using the Options setting OPT_RememberDetectorDetails).

Values

The following functions are the default scripts supplied with the dScope software. You can create your own scripts to add to this list by creating a Detector function script in the correct folder (see Detector Functions).

"2nd Harmonic Distortion"	Sets up the FFT Detector details for 2nd harmonic distortion measurement.
"3rd Harmonic Distortion"	Sets up the FFT Detector details for 3rd harmonic distortion measurement.
"4th Harmonic Distortion"	Sets up the FFT Detector details for 4th harmonic distortion measurement.
"Amplitude"	Sets up the FFT Detector details for amplitude measurement.
"Balance"	Sets up the FFT Detector details for measurement of inter-channel balance.
"Band pass"	Sets up the FFT Detector details for band pass measurement.
"Band reject"	Sets up the FFT Detector details for band reject measurement.

"Cross-talk"	Sets up the FFT Detector details for cross-talk measurement. Note that this measurement assumes that the analyzed signal originated with the dScope Signal Generator, as it tracks the generated signal on the opposite channel. For this to work successfully, one of the Generator channels must be turned off, or the generated frequencies must be different on each channel.
"Gain"	Sets up the FFT Detector details for measurement of gain with respect to the generator.
"IMD CCIF"	Sets up the FFT Detector details for measurement of IMD according to the CCIF standard ("difference-tone" IMD).
"THD+N - absolute"	Sets up the FFT Detector details for measurement of total harmonic distortion and noise, in absolute units.
"THD+N - relative"	Sets up the FFT Detector details for measurement of total harmonic distortion and noise, relative to the RMS amplitude of the analyzed signal.
"THD"	Sets up the FFT Detector details for measurement of total harmonic distortion (measured relative to the RMS amplitude of the analyzed signal).

5.3.6.1.6 FFTD_UserScript

Description

This property allows you to define a user-defined FFT Calculation script for the FFT Detector.

If this option is selected, it will override the current function selection (set by FFTD_Function).

For further details on creating FFT Detector Calculation scripts, see the FFT Detector Calculation script reference section.

Values

Any valid filename of an FFT Detector Calculation script file (*.dss) is allowed.

NB: **If a full file and path are not specified, the system will attempt to find the file by appending the default file extension for dScope scripts (*.dss) and looking for the file in the "Scripts\FFT Detector Calculations" subfolder of the dScope program folder.**

5.3.6.1.7 FFTD_BPBRMode

Description

This property allows selection of the band pass or band reject filter for the FFT Detector.

Values

FFTD_BPBRMODE_OFF	Selects the FFT Detector to have no band pass or band reject filter.
FFTD_BPBRMODE_BP	Selects the FFT Detector to have a band pass filter using the settings described in FFTD_BPBRBandwidth, FFTD_BPBRFreqMode, and FFTD_BPBRFreq.
FFTD_BPBRMODE_BR	Selects the FFT Detector to have a band reject filter with the settings described in FFTD_BPBRBandwidth, FFTD_BPBRFreqMode, and FFTD_BPBRFreq.

5.3.6.1.8 FFTD_BPBRBandwidth

Description

This property allows selection of the bandwidth of the band pass or band reject filter for the FFT Detector.

This property will have no effect unless the FFTD_BPBRMode property is set to band pass (**FFTD_BPBRMODE_BP**) or band reject (**FFTD_BPBRMODE_BR).**

Values

FFTD_BPBRBANDWIDTH_3	Selects the bandwidth for the band pass or band reject filter to be 1/3 octave.
FFTD_BPBRBANDWIDTH_6	Selects the bandwidth for the band pass or band reject filter to be 1/6 octave.
FFTD_BPBRBANDWIDTH_12	Selects the bandwidth for the band pass or band reject filter to be 1/12 octave.
FFTD_BPBRBANDWIDTH_24	Selects the bandwidth for the band pass or band reject filter to be 1/24 octave.
FFTD_BPBRBANDWIDTH_NOTCH	Selects the bandwidth for the band pass or band reject filter to be a brick wall notch filter, that notches in or out the exact width of the tone peak, including any bins that are part of the "skirt" of the notch.

NB:

In order for the FFTD_BPBRBANDWIDTH_NOTCH option to work, the system needs to know how wide the bin-spillage of the currently-selected Window function is. If you are specifying a user-defined Window function (See FFTP_UserWindowFunction, you can set the width of this notch as part of the user-defined Window function options. See the FFT Window function reference section for further details.

5.3.6.1.9 FFTD_BPBRFreqMode

Description

This property allows selection of the frequency details of the band pass or band reject filter for the FFT Detector.

This determines how the dScope sets the frequency of the filter.

This property will have no effect unless the FFTD_BPBRMode property is set to band pass (**FFTD_BPBRMODE_BP**) or band reject (**FFTD_BPBRMODE_BR).**

Values

FFTD_BPBRFREQMODE_INPUT	Selects the frequency for the band pass or band reject filter to track the channel's input frequency.
FFTD_BPBRFREQMODE_GEN	Selects the frequency for the band pass or band reject filter to track the generator frequency for each channel.
FFTD_BPBRFREQMODE_GENOTHER	Selects the frequency for the band pass or band reject filter to track the generator frequency of the *other* channel. This is useful for cross-talk measurement.

FFTD_BPBRFREQMODE_FIXED	Selects the frequency for the band pass or band reject filter to be fixed, at the frequency specified using FFTD_BPBRFreq.
FFTD_BPBRFREQMODE_IMDDIFF	Selects the frequency for the band pass or band reject filter to track the IMD differential frequency.
FFTD_BPBRFREQMODE_IMDSIDE	Selects frequencies for the band pass or band reject filter to track the channel's input frequency. This option will create "notch" filters for each frequency, regardless of the bandwidth set using FFTD_BPBRBandwidth.
FFTD_BPBRFREQMODE_ALLHARM	Selects frequencies for the band pass or band reject filter to track all the harmonics of the channel's input frequency, *not* including the fundamental. The last harmonic to include is set using the FFTD_Harmonic property.This option will create "notch" filters for each frequency, regardless of the bandwidth set using FFTD_BPBRBandwidth.
FFTD_BPBRFREQMODE_ALLHARM_FUND	Selects frequencies for the band pass or band reject filter to track all the harmonics of the channel's input frequency, including the fundamental. The last harmonic to include is set using the FFTD_Harmonic property. This option will create "notch" filters for each frequency, regardless of the bandwidth set using FFTD_BPBRBandwidth.
FFTD_BPBRFREQMODE_NTHHARM	Selects the frequency for the band pass or band reject filter to track the Nth harmonic of the channel's input frequency. The value for N is set using FFTD_Harmonic property.
FFTD_BPBRFREQMODE_2NDHARM	Supported for legacy code only: Selects the frequency for the band pass or band reject filter to track the 2nd harmonic of the channel's input frequency.
FFTD_BPBRFREQMODE_3RDHARM	Supported for legacy code only: Selects the frequency for the band pass or band reject filter to track the 3rd harmonic of the channel's input frequency.
FFTD_BPBRFREQMODE_4THHARM	Supported for legacy code only: Selects the frequency for the band pass or band reject filter to track the 4th harmonic of the channel's input frequency.

NB:

Note that the FFTD_BPBRFREQMODE_IMDDIFF and FFTD_BPBRFREQMODE_IMDSIDE options require the generator to be set up with a valid signal for IMD difference-tone analysis.

5.3.6.1.10 FFTD_Harmonic

Description

This property allows specification of the harmonic to use for the current frequency mode (FFTD_BPBRFreqMode).

If the current frequency mode is Nth harmonic (**FFTD_BPBRFREQMODE_NTHHARM**), this property will be the harmonic at which to locate the band pass or band reject frequency.

If the current frequency mode is all harmonics (**FFTD_BPBRFREQMODE_ALLHARM**), or all harmonics including the fundamental (**FFTD_BPBRFREQMODE_ALLHARM_FUND**), then this property will be used to specify the last frequency to include in the list of harmonics.

Values

The harmonic is represented as a short integer value. It can be any number between 0 and 99.

5.3.6.1.11 FFTD_BPBRFreq

Description

This property allows selection of the frequency of the band pass or band reject filter for the FFT Detector.

The frequency is specified in Hz, and is only used if the frequency mode (FFTD_BPBRFreqMode) is set to **FFTD_BPBRFREQMODE_FIXED**.

Values

The BP/BR frequency is represented as a double-precision floating point value.

5.3.6.1.12 FFTD_HPFilter

Description

This property allows selection of the high-pass filter for the FFT Detector.

Values

FFTD_HP_OFF	Sets high-pass filter for the FFT Detector to off (see note below).
FFTD_HP_DCB	Sets high-pass filter for the FFT Detector to a DC blocking filter.
FFTD_HP_10HZ	Sets high-pass filter for the FFT Detector to 10Hz.
FFTD_HP_22HZ	Sets high-pass filter for the FFT Detector to 22Hz.
FFTD_HP_100HZ	Sets high-pass filter for the FFT Detector to 100Hz.
FFTD_HP_400HZ	Sets high-pass filter for the FFT Detector to 400Hz.
FFTD_HP_DEFAULT	Sets high-pass filter for the FFT Detector to follow the default set up on the Signal Analyzer (see SA_DefaultHPFilter). This allows an easy way of switching filters for multiple Detectors at the same time.

NB:

1) The dScope analogue hardware has a built-in DC blocking filter, which can be turned on or off using a jumper on the board (see PCB jumper options for further details).

In analogue analysis mode, if the analogue hardware is not DC coupled, then selection of "FFTD_HP_OFF" is disabled and the DC blocking filter is automatically selected (FFTD_HP_DCB).

NB:

2) To set a specific frequency for the high-pass filter, use FFTD_HPFilterFreq.

5.3.6.1.13 FFTD_HPFilterFreq

Description

This property allows specification of the frequency of the high-pass filter for the FFT Detector.

Values

The high-pass filter frequency is represented as a long integer value.

NB: **If the high-pass filter is one of the predefined filters (See FFTD_HPFilter), then this property will return the *actual* frequency that this represents.
For example, if FFTD_HPFilter has been set to FFTD_HP_100HZ, then this property will return 100. Off (FFTD_HP_OFF) will return zero.**

5.3.6.1.14 FFTD_BrickWallHPFilter

Description

This property is used to specify whether the FFT Detector's high-pass filter is a "brick wall" filter, i.e. that the gain drops from unity gain to zero at the frequency specified by FFTD_HPFilter or FFTD_HPFilterFreq.

Values

True	This FFT Detector's high-pass filter is a brick wall filter.
False	This FFT Detector's high-pass filter emulates the rolloff of the Continuous-Time Detector's high-pass filter.

5.3.6.1.15 FFTD_LPFilter

Description

This property allows selection of the low-pass filter for the FFT Detector.

Values

FFTD_LP_OFF	Sets low-pass filter for the FFT Detector to off.
FFTD_LP_20KHZ_AES17	Sets low-pass filter for the FFT Detector to a 20kHz filter that matches the AES17 low-pass filter specification.
FFTD_LP_22KHZ	Sets low-pass filter for the FFT Detector to 22kHz.
FFTD_LP_30KHZ	Sets low-pass filter for the FFT Detector to 30kHz.
FFTD_LP_40KHZ	Sets low-pass filter for the FFT Detector to 40kHz.
FFTD_LP_80KHZ	Sets low-pass filter for the FFT Detector to 80kHz. NB: This option is ignored if the hardware is not 192kHz-capable.
FFTD_LP_DEFAULT	Sets low-pass filter for the FFT Detector to follow the default set up on the Signal Analyzer (see SA_DefaultLPFilter). This allows an easy way of switching filters for multiple Detectors at the same time.

NB: **To set a specific frequency for the low-pass filter, use FFTD_LPFilterFreq.**

5.3.6.1.16 FFTD_LPFilterFreq

Description

This property allows specification of the frequency of the low-pass filter for the FFT Detector.

Values

The low-pass filter frequency is represented as a long integer value.

NB: **If the low-pass filter is one of the predefined filters (See FFTD_LPFilter), then this property will return the *actual* frequency that this represents.**
For example, if FFTD_LPFilter has been set to FFTD_LP_22KHZ, then this property will return 22000. Off (FFTD_LP_OFF) will return zero.

5.3.6.1.17 FFTD_BrickWallLPFilter

Description

This property is used to specify whether the FFT Detector's low-pass filter is a "brick wall" filter, i.e. that the gain drops from unity gain to zero at the frequency specified using FFTD_LPFilter or FFTD_LPFilterFreq.

Values

True	This FFT Detector's low-pass filter is a brick wall filter.
False	This FFT Detector's low-pass filter emulates the rolloff of the Continuous-Time Detector's low-pass filter.

5.3.6.1.18 FFTD_WeightingFilter

Description

This property allows selection of the weighting filter for the FFT Detector.

NB: **If a pre-weighting filter has been set up in the FFT Parameters (FFTP_WeightingFilter), a filters applied to an FFT Detector will be in ADDITION to this. If the same filter is selected in both an FFT Detector AND the FFT Parameters, it will be applied twice, giving incorrect results.**

Values

FFTD_WEIGHTING_NONE	Sets weighting filter for the FFT Detector to none (not weighted).
FFTD_WEIGHTING_AWEIGHTING	Sets weighting filter for the FFT Detector to A-weighted.
FFTD_WEIGHTING_CWEIGHTING	Sets weighting filter for the FFT Detector to C-weighted.
FFTD_WEIGHTING_CCIR468_1K	Sets weighting filter for the FFT Detector to a CCIR-468 shape, with unity gain at 1kHz.
FFTD_WEIGHTING_CCIR468_2K	Sets weighting filter for the FFT Detector to a CCIR-468 shape, with unity gain at 2kHz.
FFTD_WEIGHTING_DEFAULT	Sets weighting filter for the FFT Detector to follow the default set up on the Signal Analyzer (see SA_DefaultWeightingFilter). This allows an easy way of switching filters for multiple Detectors at the same time.
FFTD_WEIGHTING_USER	Sets weighting filter for the FFT Detector to use the user-defined weighting filter specified by FFTD_UserWeightingFilter.

5.3.6.1.19 FFTD_UserWeightingFilter

Description

This property allows specification of a user-defined weighting filter for the FFT Detector. This will be the file name of a weighting filter table, or a script used to create such a table.

For further details on setting up user-defined weighting filters, see FFT Detector Weighting filters. For a full reference of script functions available to write scripts to create weighting filters, see the FFT Detector Weighting filter reference section.

NB: **If a pre-weighting filter has been set up in the FFT Parameters (FFTP_WeightingFilter), a filters applied to an FFT Detector will be in ADDITION to this. If the same filter is selected in both an FFT Detector AND the FFT Parameters, it will be applied twice, giving incorrect results.**

Values

Any valid filename of a weighting filter table (*.wgt), or a dScope script file (*.dss) is allowed.

NB: **If a full file and path are not specified, the system will attempt to find the file by appending the default file extension for weighting filters (*.wgt) and looking for the file in the "FFT Detector Weighting filters" subfolder of the dScope program folder.**

5.3.6.1.20 FFTD_Relativity

Description

This property allows selection of the relativity of the FFT Detector.

The FFT Detector measurement can either be made in absolute units, or relative to a signal.

Values

FFTD_RELATIVITY_ABS	Sets the FFT Detector to display absolute Result values.
FFTD_RELATIVITY_SELF	Sets the FFT Detector to display Result values relative to the incoming signal, as read by the Signal Analyzer (See SA_ChARMSAmpl and SA_ChBRMSAmpl).
FFTD_RELATIVITY_GEN	Sets the FFT Detector to display Result values relative to the generated amplitude for the same channel.
FFTD_RELATIVITY_CHANNEL	Sets the FFT Detector to display Result values relative to the RMS amplitude of the other channel.
FFTD_RELATIVITY_USER	This value is only available from an FFT Detector Calculation script. It allows the user to set the FFT Detector value as either a percentage or a number of dB, so the system need have no knowledge of what the value was originally related to. (all other relativities need to know this information, so they can be displayed correctly). For further details of how to use this value, see the FFT Detector Calculation script reference section.

NB: **The list of available units will change depending on the relativity. If FFTD_RELATIVITY_ABS is selected, then a list of absolute units will be available.**

If any of the other relativities are selected, then only dB and % are available (See FFTD_Unit for details of available units).

5.3.6.2 FFT Detector Calculation script Reference

FFT Detector Calculation scripts allow the user to define complex functions for FFT Detectors, which can access data from the individual samples of the incoming sample buffer, or the bins of the FFT (before or after filtering has taken place).

If an FFT Detector has a Calculation script defined as its current function, this script will be run *every* time the FFT calculation completes after each buffer acquisition.

A Calculation script can take one of two approaches to retrieving data from an FFT buffer:
1) It can either work out itself which bins to look at, and take data from them;
2) It can leave the existing FFT Detector filters to filter bins in or out, and then access individual bins or sum a groups of bins to get its Result value.

To this end, there are basically three types of methods that an FFT Detector makes available:
1) Methods to sum a total of bins;
2) Methods to get individual values from the buffer;
3) Miscellaneous functions such as retrieving the FFT buffer size, or setting the final FFT Detector Result value.

Once an FFT Detector has performed its calculations on values taken from the buffer, it will reach a final result for a channel. This value can then be set as the FFT Detector's Result using FFTD_SetChannelA or FFTD_SetChannelB. Once this has been done, the relativity and unit can be changed from the dScope user interface, regardless of the unit and relativity that the value was set in.

Methods

The following methods are available to FFT Detectors via Calculation scripts.

FFTD_SetChannelA
FFTD_SetChannelB
FFTD_GetBufferSize
FFTD_GetBufferValueAt
FFTD_GetFFTBinPowerInUnit
FFTD_GetUnfilteredFFTBinTotal
FFTD_GetFilteredFFTBinTotal
FFTD_SumBufferBins
FFTD_SumBufferEvenBins
FFTD_SumBufferOddBins
FFTD_GetBufferHighestAmplToneBin
FFTD_GetBufferLowestAmplToneBin
FFTD_GetBuffer

NB:

When FFT Detector Calculation scripts are run, it is guaranteed that the dScope's FFT buffer will not change mid-way through the script running.
However, if you call any of the above functions from a script that is *not* an FFT Detector Calculation script, then you must ensure yourself that the FFT buffer cannot be updated mid-way through the script's access to the FFT buffer, or your results may be incorrect.

Units and FFT power calculations

FFT calculations are power calculations, and the resulting buffer contains power values. The values in individual bins are in a unit defined as a "bin power", and are not in any of the usual units available to FFT Detectors. We must therefore introduce a new unit, called **UNIT_BINPOWER**.

Methods to retrieve a value from a bin, or a sum of bins, all require a parameter to be passed which indicates the unit that the value should be returned in. The usual set of units is available (dBFS, dBu, V etc) as well as the new **UNIT_BINPOWER**.

NB:

If you need to do any basic addition or subtraction on values from an FFT buffer, this maths MUST be done with the unit set to UNIT_BINPOWER.

If any bin summing is to be done by the script itself, then the safest thing to do is to keep the unit as UNIT_BINPOWER. The FFTD_SetChannelA and FFTD_SetChannelB methods will accept this unit when setting the value after calculation is finished, so there is no need for the script to worry about unit conversion.

5.3.6.2.1 FFTD_SetChannelA

FFTD_SetChannelA (dValue, sUnit, sRelativity)

This property allows the Calculation script to set the value to be used by the FFT Detector Channel A Result.

This is the value returned by the FFTD_ChA property of the FFT Detector, although FFTD_ChA will return the value in the unit currently selected by FFTD_Unit, which may be a different unit from the ***sUnit*** parameter specified here.

Parameters

dValue	The value to use as the FFT Detector Channel A Result.
sUnit	The unit that the ***dValue*** parameter is specified in. It can have any of the values specified in the Units section below.
sRelativity	The relativity that the ***dValue*** parameter is specified in. It can have any of the values specified in the Relativity section below.

NB:

The s*Unit* and *sRelativity* parameters are *only* used to specify the entry of the *dValue* parameter, and are not necessarily the same as the current unit and relativity on the FFT Detector user interface, or returned by FFTD_Unit and FFTD_Relativity.

Return value

This method has no return value.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBFS.

UNIT_PERCENTFS	Specifies that the ***dValue*** parameter is an absolute amplitude, in % FS (percentage of full scale).
UNIT_FFS	Specifies that the ***dValue*** parameter is an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the ***dValue*** parameter is an absolute amplitude, in Hex.
UNIT_V	Specifies that the ***dValue*** parameter is an absolute amplitude, in V.
UNIT_DBU	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBm.
UNIT_W	Specifies that the ***dValue*** parameter is an absolute amplitude, in W.
UNIT_DBR	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the ***dValue*** parameter is an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the ***dValue*** parameter is in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the ***dValue*** parameter is a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the ***dValue*** parameter is specified as an absolute amplitude.
FFTD_RELATIVITY_SELF	Specifies that the ***dValue*** parameter is specified relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the ***dValue*** parameter is specified relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the ***dValue*** parameter is specified relative to the incoming RMS amplitude on channel B.
FFTD_RELATIVITY_USER	Specifies that the ***dValue*** parameter is a "relative" value (i.e. specified in % or dB) but that the dScope has no knowledge of what it is relative to.

NB:

If FFTD_RELATIVITY_USER is passed as the relativity, the relativity will be disabled on the dScope user interface. This is because the dScope has no knowledge of what the entry is relative to, and so cannot convert to and from other relativities and units.

5.3.6.2.2 FFTD_SetChannelB

FFTD_SetChannelB (dValue, sUnit, sRelativity)

This property allows the Calculation script to set the value to be used by the FFT Detector Channel B Result.

This is the value returned by the FFTD_ChB property of the FFT Detector, although FFTD_ChB will return the value in the unit currently selected by FFTD_Unit, which may be a different unit to the ***sUnit*** parameter specified here.

Parameters

dValue	The value to use as the FFT Detector Channel B Result.
sUnit	The unit that the ***dValue*** parameter is specified in. It can have any of the values specified in the Units section below.
sRelativity	The relativity that the ***dValue*** parameter is specified in. It can have any of the values specified in the Relativity section below.

NB: **The s*Unit* and *sRelativity* parameters are *only* used to specify the entry of the *dValue* parameter, and are not necessarily the same as the current unit and relativity on the FFT Detector user interface, or returned by FFTD_Unit and FFTD_Relativity.**

Return value

This method has no return value.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBFS.
UNIT_PERCENTFS	Specifies that the ***dValue*** parameter is an absolute amplitude, in % FS (percentage of full scale).
UNIT_FFS	Specifies that the ***dValue*** parameter is an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the ***dValue*** parameter is an absolute amplitude, in Hex.
UNIT_V	Specifies that the ***dValue*** parameter is an absolute amplitude, in V.
UNIT_DBU	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBm.
UNIT_W	Specifies that the ***dValue*** parameter is an absolute amplitude, in W.
UNIT_DBR	Specifies that the ***dValue*** parameter is an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the ***dValue*** parameter is an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the ***dValue*** parameter is in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the ***dValue*** parameter is a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the ***dValue*** parameter is specified as an absolute amplitude.
FFTD_RELATIVITY_SELF	Specifies that the ***dValue*** parameter is specified relative to the incoming signal on channel B, as read by the Signal Analyzer (See SA_ChBRMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the ***dValue*** parameter is specified relative to the generated amplitude for channel B.
FFTD_RELATIVITY_CHANNEL	Specifies that the ***dValue*** parameter is specified relative to the incoming RMS amplitude on channel A.
FFTD_RELATIVITY_USER	Specifies that the ***dValue*** parameter is a "relative" value (i.e. specified in % or dB) but that the dScope has no knowledge of what it is relative to.

NB: **If FFTD_RELATIVITY_USER is passed as the relativity, the relativity will be disabled on the dScope user interface. This is because the dScope has no knowledge of what the entry is relative to, and so cannot convert to and from other relativities and units.**

5.3.6.2.3 FFTD_GetBufferSize

lSize = FFTD_GetBufferSize (sBuffer)

This method returns the size of the buffer specified.

NB: **This method will fail and return 0 if no FFT has yet been done.**

Parameters

sBuffer	This parameter specifies which buffer to get the size of. It can have any of the values specified in the Buffers section below.

Return value

This method returns the number of samples in the specified buffer.

Buffers

FFTD_BUFFER_SAMPLE	The sample buffer, before FFT calculations
FFTD_BUFFER_FFT_UNFILTERED	The FFT buffer, after FFT calculations but before any filters have been applied.
FFTD_BUFFER_FFT_PREWEIGHTED	The FFT buffer, after FFT calculations and after pre-weighting, but before any FFT Detectors have been applied. If pre-weighting is not turned on in the FFT Parameters (See FFTP_WeightingFilter), this will return the unfiltered buffer.
FFTD_BUFFER_FFT_FILTERED	The FFT buffer, after FFT calculations and filters have been applied.
FFTD_BUFFER_PHASE	The buffer containing phase information from the FFT (can currently ONLY be returned in radians, with an arbitrary phase offset). This value can only be used if the FFTP_CalcPhaseInfo property has been set to **True**.
FFTD_BUFFER_CTA	The *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).
FFTD_BUFFER_CTA_FFT	The FFT of the *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).

NB:

1) The sample buffer size is the same as the number of FFT points (See FFTP_NumPoints). The FFT buffers are half this size.
If an FFT has not yet been done, then no FFT buffer will be available and this function will return 0.

2) Access to the FFTD_BUFFER_CTA and FFTD_BUFFER_CTA_FFT will fail if these buffers are not currently being displayed on the Trace window.

5.3.6.2.4 FFTD_GetBufferValueAt

dValue = FFTD_GetBufferValueAt (sBuffer, sChannel, lIndex, sUnit, sRelativity)

This method returns the value of the given bin in the buffer specified.

NB:

This method will fail if no FFT has yet been done.

Parameters

sBuffer	Specifies which buffer to get the value from. It can have any of the values specified in the Buffers section below.
sChannel	Specifies which channel's buffer to get the value from. For possible values, see the Channels section below.
lIndex	Specifies the index in the buffer to get the value from. This index is zero-based. NB: If an FFT has not yet been done, then no FFT buffer will be available and this method will display an error message.
sUnit	The unit to get the value in. For a list of possible values, see the Units section below.
sRelativity	The relativity of the value to be returned. For a list of possible values, see the Relativity section below.

Return value

The value at the given index in the buffer, in the unit specified. This value is returned as a double-precision floating point value.

Buffers

The following values are valid for the ***sBuffer*** parameter.

FFTD_BUFFER_SAMPLE	The sample buffer, before FFT calculations
FFTD_BUFFER_FFT_UNFILTERED	The FFT buffer, after FFT calculations but before any filters have been applied.
FFTD_BUFFER_FFT_PREWEIGHTED	The FFT buffer, after FFT calculations and after pre-weighting, but before any FFT Detectors have been applied. If pre-weighting is not turned on in the FFT Parameters (See FFTP_WeightingFilter), this will use the unfiltered buffer.
FFTD_BUFFER_FFT_FILTERED	The FFT buffer, after FFT calculations and filters have been applied.
FFTD_BUFFER_PHASE	The buffer containing phase information from the FFT (can currently ONLY be returned in radians, with an arbitrary phase offset). This value can only be used if the FFTP_CalcPhaseInfo property has been set to **True**.
FFTD_BUFFER_CTA	The *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).
FFTD_BUFFER_CTA_FFT	The FFT of the *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).

NB: **Access to the FFTD_BUFFER_CTA and FFTD_BUFFER_CTA_FFT will fail if these buffers are not currently being displayed on the Trace window.**

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that the value should be obtained from channel A's buffer.
CHANNEL_B	Specifies that the value should be obtained from channel B's buffer.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the value returned should be an absolute amplitude, in dBFS.
UNIT_PERCENTFS	Specifies that the value returned should be an absolute amplitude, in %FS (percentage of full scale).
UNIT_FFS	Specifies that the value returned should be an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the value returned should be an absolute amplitude, in Hex.
UNIT_V	Specifies that the value returned should be an absolute amplitude, in V.
UNIT_DBU	Specifies that the value returned should be an absolute amplitude, in dBu.

UNIT_DBV	Specifies that the value returned should be an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the value returned should be an absolute amplitude, in dBm.
UNIT_W	Specifies that the value returned should be an absolute amplitude, in W.
UNIT_DBR	Specifies that the value returned should be an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the value returned should be an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the value returned should be in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the value returned should be a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the value returned should be absolute, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the value returned should be relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the value returned should be relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the value returned should be relative to the incoming RMS amplitude on the opposite channel.

5.3.6.2.5 FFTD_GetFFTBinPowerInUnit

dValue = FFTD_GetFFTBinPowerInUnit (dValue, sChannel, sUnit, sRelativity)

This method takes a value in a unit of FFT_BINPOWER, and returns it in the unit specified.

This allows easy conversion of values directly from an FFT buffer, into a value that can be understood by the user.

For details of the FFT_BINPOWER unit, see Units and FFT power calculations.

Parameters

dValue	Specifies the value, in a unit of FFT_BINPOWER, to convert.
sChannel	Specifies which channel's buffer the value is from. This is important when calculating relativity, as most relativity units need knowledge of which channel the value is relative to. For possible values, see the Channels section below.
sUnit	The unit to get the value in. For a list of possible values, see the Units section below.
sRelativity	The relativity of the value to be returned. For a list of possible values, see the Relativity section below.

Return value

The value given, converted to the specified unit. This value is returned as a double-precision floating point value.

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that the value passed is from channel A's buffer.
CHANNEL_B	Specifies that the value passed is from channel B's buffer.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the value returned should be an absolute amplitude, in dBFS.
UNIT_PERCENTFS	Specifies that the value returned should be an absolute amplitude, in %FS (percentage of full scale).
UNIT_FFS	Specifies that the value returned should be an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the value returned should be an absolute amplitude, in Hex.
UNIT_V	Specifies that the value returned should be an absolute amplitude, in V.
UNIT_DBU	Specifies that the value returned should be an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the value returned should be an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the value returned should be an absolute amplitude, in dBm.
UNIT_W	Specifies that the value returned should be an absolute amplitude, in W.
UNIT_DBSPL	Specifies that the value returned should be an absolute amplitude, in dBSPL.
UNIT_DBR	Specifies that the value returned should be an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the value returned should be an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the value returned should be in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the value returned should be a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the value returned should be absolute, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the value returned should be relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the value returned should be relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the value returned should be relative to the incoming RMS amplitude on the opposite channel.

5.3.6.2.6 FFTD_GetUnfilteredFFTBinTotal

dValue = FFTD_GetUnfilteredFFTBinTotal(sChannel, sUnit, sRelativity)

This method returns the total sum of all bins in the unfiltered FFT buffer (after FFT calculation, but BEFORE any filters have been applied).

NB: **This method will fail if no FFT has yet been done.**

Parameters

sChannel	Specifies which channel's buffer to get the total for. For possible values, see the Channels section below.
sUnit	The unit to get the total in. For a list of possible values, see the Units section below.
sRelativity	The relativity of the value to be returned. For a list of possible values, see the Relativity section below.

Return value

The total sum of all the bins in the unfiltered FFT buffer, in the unit specified. This value is returned as a double-precision floating point value.

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that the channel A buffer's bin total should be returned.
CHANNEL_B	Specifies that the channel B buffer's bin total should be returned.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the value returned should be an absolute amplitude, in dBFS.
UNIT_PERCENTFS	Specifies that the value returned should be an absolute amplitude, in %FS (percentage of full scale).
UNIT_FFS	Specifies that the value returned should be an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the value returned should be an absolute amplitude, in Hex.
UNIT_V	Specifies that the value returned should be an absolute amplitude, in V.
UNIT_DBU	Specifies that the value returned should be an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the value returned should be an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the value returned should be an absolute amplitude, in dBm.
UNIT_W	Specifies that the value returned should be an absolute amplitude, in W.
UNIT_DBSPL	Specifies that the value returned should be an absolute amplitude, in dBSPL.
UNIT_DBR	Specifies that the value returned should be an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the value returned should be an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the value returned should be in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the value returned should be a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the value returned should be absolute, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the value returned should be relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the value returned should be relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the value returned should be relative to the incoming RMS amplitude on the opposite channel.

5.3.6.2.7 FFTD_GetFilteredFFTBinTotal

dValue = FFTD_GetFilteredFFTBinTotal(sChannel, sUnit, sRelativity)

This method returns the total sum of all bins in the filtered FFT buffer (after FFT calculation and all filters have been applied).

NB: **This method will fail if no FFT has yet been done.**

Parameters

sChannel	Specifies which channel's buffer to get the total for. For possible values, see the Channels section below.
sUnit	The unit to get the total in. For a list of possible values, see the Units section below.
sRelativity	The relativity of the value to be returned. For a list of possible values, see the Relativity section below.

Return value

The total sum of all the bins in the filtered FFT buffer, in the unit specified. This value is returned as a double-precision floating point value.

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that the channel A buffer's bin total should be returned.
CHANNEL_B	Specifies that the channel B buffer's bin total should be returned.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the value returned should be an absolute amplitude, in dBFS.
UNIT_PERCENTFS	Specifies that the value returned should be an absolute amplitude, in %FS (percentage of full scale).
UNIT_FFS	Specifies that the value returned should be an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the value returned should be an absolute amplitude, in Hex.
UNIT_V	Specifies that the value returned should be an absolute amplitude, in V.
UNIT_DBU	Specifies that the value returned should be an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the value returned should be an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the value returned should be an absolute amplitude, in dBm.
UNIT_W	Specifies that the value returned should be an absolute amplitude, in W.

UNIT_DBSPL	Specifies that the value returned should be an absolute amplitude, in dBSPL.
UNIT_DBR	Specifies that the value returned should be an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the value returned should be an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the value returned should be in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the value returned should be a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the value returned should be absolute, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the value returned should be relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the value returned should be relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the value returned should be relative to the incoming RMS amplitude on the other channel.

5.3.6.2.8 FFTD_SumBufferBins

dValue = FFTD_SumBufferBins(sBuffer, sChannel, lStartBin, lEndBin, sUnit, sRelativity)

This method returns the sum of all bins in the given buffer, between the two bins specified.

NB: **This method will fail if no FFT has yet been done.**

Parameters

sBuffer	Specifies which buffer's bins should be summed. For possible values, see the Buffers section below.
sChannel	Specifies which channel's buffer to get the total for. For possible values, see the Channels section below.
lStartBin	Specifies the first bin that should be included in the total returned. The bins are zero-indexed.
lEndBin	Specifies the last bin that should be included in the total returned. The bins are zero-indexed.
sUnit	The unit to get the bin sum in. For a list of possible values, see the Units section below.
sRelativity	The relativity of the value to be returned. For a list of possible values, see the Relativity section below.

Return value

The sum of bins in the selected buffer, between the stated bins, in the unit specified. This value is returned as a double-precision floating point value.

Buffers

The following values are valid for the ***sBuffer*** parameter.

FFTD_BUFFER_SAMPLE	The sample buffer, before FFT calculations
FFTD_BUFFER_FFT_PREWEIGHTED	The FFT buffer, after FFT calculations and after pre-weighting, but before any FFT Detectors have been applied. If pre-weighting is not turned on in the FFT Parameters (See FFTP_WeightingFilter), this will use the unfiltered buffer.
FFTD_BUFFER_FFT_UNFILTERED	The FFT buffer, after FFT calculations but before any filters have been applied.
FFTD_BUFFER_FFT_FILTERED	The FFT buffer, after FFT calculations and filters have been applied.

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that the channel A buffer's bins should be summed.
CHANNEL_B	Specifies that the channel B buffer's bins should be summed.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the value returned should be an absolute amplitude, in dBFS.
UNIT_PERCENTFS	Specifies that the value returned should be an absolute amplitude, in %FS (percentage of full scale).
UNIT_FFS	Specifies that the value returned should be an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the value returned should be an absolute amplitude, in Hex.
UNIT_V	Specifies that the value returned should be an absolute amplitude, in V.
UNIT_DBU	Specifies that the value returned should be an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the value returned should be an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the value returned should be an absolute amplitude, in dBm.
UNIT_W	Specifies that the value returned should be an absolute amplitude, in W.
UNIT_DBR	Specifies that the value returned should be an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the value returned should be an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the value returned should be in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the value returned should be a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the value returned should be absolute, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the value returned should be relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the value returned should be relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the value returned should be relative to the incoming RMS amplitude on the opposite channel.

5.3.6.2.9 FFTD_SumBufferEvenBins

dValue = FFTD_SumBufferEvenBins(sBuffer, sChannel, lStartBin, lEndBin, sUnit, sRelativity)

This method returns the sum of all EVEN bins in the given buffer, between the two bins specified.

This can be useful for measuring distortion or noise in multi-tone signals - multi-tone signals are specified such that all tones are in even-numbered bins, so all distortion will also fall in even-numbered bins, and all odd-numbered bins will contain only noise.

NB: **This method will fail if no FFT has yet been done.**

Parameters

sBuffer	Specifies which buffer's bins should be summed. For possible values, see the Buffers section below.
sChannel	Specifies which channel's buffer to get the total for. For possible values, see the Channels section below.
lStartBin	Specifies the first bin that should be included in the total returned. The bins are zero-indexed.
lEndBin	Specifies the last bin that should be included in the total returned. The bins are zero-indexed.
sUnit	The unit to get the bin sum in. For a list of possible values, see the Units section below.
sRelativity	The relativity of the value to be returned. For a list of possible values, see the Relativity section below.

NB: **If *lStartBin* is not even, then dScope will start the summing at the next bin after it. If *lEndBin* is not even, dScope will stop summing at the last bin before it.**

Return value

The sum of all EVEN bins in the selected buffer, between the stated bins, in the unit specified. This value is returned as a double-precision floating point value.

Buffers

The following values are valid for the ***sBuffer*** parameter.

FFTD_BUFFER_SAMPLE	The sample buffer, before FFT calculations
FFTD_BUFFER_FFT_UNFILTERED	The FFT buffer, after FFT calculations but before any filters have been applied.
FFTD_BUFFER_FFT_PREWEIGHTED	The FFT buffer, after FFT calculations and after pre-weighting, but before any FFT Detectors have been applied. If pre-weighting is not turned on in the FFT Parameters (See FFTP_WeightingFilter), this will use the unfiltered buffer.
FFTD_BUFFER_FFT_FILTERED	The FFT buffer, after FFT calculations and filters have been applied.

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that the channel A buffer's bins should be summed.
CHANNEL_B	Specifies that the channel B buffer's bins should be summed.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the value returned should be an absolute amplitude, in dBFS.
UNIT_PERCENTFS	Specifies that the value returned should be an absolute amplitude, in %FS (percentage of full scale).
UNIT_FFS	Specifies that the value returned should be an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the value returned should be an absolute amplitude, in Hex.
UNIT_V	Specifies that the value returned should be an absolute amplitude, in V.
UNIT_DBU	Specifies that the value returned should be an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the value returned should be an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the value returned should be an absolute amplitude, in dBm.
UNIT_W	Specifies that the value returned should be an absolute amplitude, in W.
UNIT_DBR	Specifies that the value returned should be an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the value returned should be an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the value returned should be in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the value returned should be a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the value returned should be absolute, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the value returned should be relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the value returned should be relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the value returned should be relative to the incoming RMS amplitude on the opposite channel.

5.3.6.2.10 FFTD_SumBufferOddBins

dValue = FFTD_SumBufferOddBins(sBuffer, sChannel, lStartBin, lEndBin, sUnit, sRelativity)

This method returns the sum of all ODD bins in the given buffer, between the two bins specified.

This can be useful for measuring distortion or noise in multi-tone signals - multi-tone signals are specified such that all tones are in even-numbered bins, so all distortion will also fall in even-numbered bins, and all odd-numbered bins will contain only noise.

NB: **This method will fail if no FFT has yet been done.**

Parameters

sBuffer	Specifies which buffer's bins should be summed. For possible values, see the Buffers section below.
sChannel	Specifies which channel's buffer to get the total for. For possible values, see the Channels section below.
lStartBin	Specifies the first bin that should be included in the total returned. The bins are zero-indexed.
lEndBin	Specifies the last bin that should be included in the total returned. The bins are zero-indexed.
sUnit	The unit to get the bin sum in. For a list of possible values, see the Units section below.
sRelativity	The relativity of the value to be returned. For a list of possible values, see the Relativity section below.

NB: **If *lStartBin* is not odd, then dScope will start the summing at the next bin after it. If *lEndBin* is not odd, dScope will stop summing at the last bin before it.**

Return value

The sum of all ODD bins in the selected buffer, between the stated bins, in the unit specified. This value is returned as a double-precision floating point value.

Buffers

The following values are valid for the ***sBuffer*** parameter.

FFTD_BUFFER_SAMPLE	The sample buffer, before FFT calculations
FFTD_BUFFER_FFT_UNFILTERED	The FFT buffer, after FFT calculations but before any filters have been applied.
FFTD_BUFFER_FFT_PREWEIGHTED	The FFT buffer, after FFT calculations and after pre-weighting, but before any FFT Detectors have been applied. If pre-weighting is not turned on in the FFT Parameters (See FFTP_WeightingFilter), this will use the unfiltered buffer.
FFTD_BUFFER_FFT_FILTERED	The FFT buffer, after FFT calculations and filters have been applied.

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that the channel A buffer's bins should be summed.
CHANNEL_B	Specifies that the channel B buffer's bins should be summed.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the value returned should be an absolute amplitude, in dBFS.
UNIT_PERCENTFS	Specifies that the value returned should be an absolute amplitude, in %FS (percentage of full scale).
UNIT_FFS	Specifies that the value returned should be an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the value returned should be an absolute amplitude, in Hex.
UNIT_V	Specifies that the value returned should be an absolute amplitude, in V.
UNIT_DBU	Specifies that the value returned should be an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the value returned should be an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the value returned should be an absolute amplitude, in dBm.
UNIT_W	Specifies that the value returned should be an absolute amplitude, in W.
UNIT_DBR	Specifies that the value returned should be an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the value returned should be an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the value returned should be in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the value returned should be a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the value returned should be absolute, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the value returned should be relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the value returned should be relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the value returned should be relative to the incoming RMS amplitude on the opposite channel.

5.3.6.2.11 FFTD_GetBufferHighestAmplToneBin

lBin = FFTD_GetBufferHighestAmplToneBin(sBuffer, sChannel, lStartBin, lEndBin)

This method returns the index of the bin in a buffer which contains the highest amplitude tone.

Once the bin has been found, the FFTD_GetBufferValueAt method can be used to obtain the amplitude of the tone.

NB: **This method will fail if no FFT has yet been done.**

Parameters

sBuffer	Specifies which buffer should be used. For possible values, see the Buffers section below.
sChannel	Specifies which channel's buffer should be used. For possible values, see the Channels section below.
lStartBin	Specifies the first bin that should be included in the search. The bins are zero-indexed.
lEndBin	Specifies the last bin that should be included in the search. The bins are zero-indexed.

Return value

The index of the bin containing the highest amplitude. This value is returned as a long integer value.

Buffers

The following values are valid for the ***sBuffer*** parameter.

FFTD_BUFFER_SAMPLE	The sample buffer, before FFT calculations
FFTD_BUFFER_FFT_UNFILTERED	The FFT buffer, after FFT calculations but before any filters have been applied.
FFTD_BUFFER_FFT_PREWEIGHTED	The FFT buffer, after FFT calculations and after pre-weighting, but before any FFT Detectors have been applied. If pre-weighting is not turned on in the FFT Parameters (See FFTP_WeightingFilter), this will use the unfiltered buffer.
FFTD_BUFFER_FFT_FILTERED	The FFT buffer, after FFT calculations and filters have been applied.
FFTD_BUFFER_PHASE	The buffer containing phase information from the FFT (can currently ONLY be returned in radians, with an arbitrary phase offset). This value can only be used if the FFTP_CalcPhaseInfo property has been set to **True**.
FFTD_BUFFER_CTA	The *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).
FFTD_BUFFER_CTA_FFT	The FFT of the *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).

NB: **Access to the FFTD_BUFFER_CTA and FFTD_BUFFER_CTA_FFT will fail if these buffers are not currently being displayed on the Trace window.**

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that channel A's buffer should be searched.
CHANNEL_B	Specifies that channel B's buffer should be searched.

5.3.6.2.12 FFTD_GetBufferLowestAmplToneBin

lBin = FFTD_GetBufferLowestAmplToneBin(sBuffer, sChannel, lStartBin, lEndBin, dThreshold, sThresholdUnit, sThresholdRelativity)

This method returns the index of the bin in a buffer which contains the lowest amplitude tone.

Because we are looking for one of the actual tones of a signal, we must specify a threshold under which all bins are ignored.

Once the bin has been found, the FFTD_GetBufferValueAt method can be used to obtain the amplitude of the tone.

NB: **This method will fail if no FFT has yet been done.**

Parameters

sBuffer	Specifies which buffer should be used. For possible values, see the Buffers section below.
sChannel	Specifies which channel's buffer should be used. For possible values, see the Channels section below.

lStartBin	Specifies the first bin that should be included in the search. The bins are zero-indexed.
lEndBin	Specifies the last bin that should be included in the search. The bins are zero-indexed.
dThreshold	Specifies the threshold below which tones should be ignored. This is specified with the ***sUnit*** and ***sRelativity*** parameters.
sThresholdUnit	The unit that the dThreshold parameter is specified in. For a list of possible values, see the Units section below.
sThresholdRelativity	The relativity of the dThreshold parameter. For a list of possible values, see the Relativity section below.

Return value

The index of the bin containing the lowest amplitude which is above the threshold value passed. This value is returned as a long integer value.

Buffers

The following values are valid for the ***sBuffer*** parameter.

FFTD_BUFFER_SAMPLE	The sample buffer, before FFT calculations
FFTD_BUFFER_FFT_UNFILTERED	The FFT buffer, after FFT calculations but before any filters have been applied.
FFTD_BUFFER_FFT_PREWEIGHTED	The FFT buffer, after FFT calculations and after pre-weighting, but before any FFT Detectors have been applied. If pre-weighting is not turned on in the FFT Parameters (See FFTP_WeightingFilter), this will use the unfiltered buffer.
FFTD_BUFFER_FFT_FILTERED	The FFT buffer, after FFT calculations and filters have been applied.
FFTD_BUFFER_PHASE	The buffer containing phase information from the FFT (can currently ONLY be returned in radians, with an arbitrary phase offset). This value can only be used if the FFTP_CalcPhaseInfo property has been set to **True**.
FFTD_BUFFER_CTA	The *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).
FFTD_BUFFER_CTA_FFT	The FFT of the *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).

NB: **Access to the FFTD_BUFFER_CTA and FFTD_BUFFER_CTA_FFT will fail if these buffers are not currently being displayed on the Trace window.**

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that channel A's buffer should be searched.
CHANNEL_B	Specifies that channel B's buffer should be searched.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the threshold specified is an absolute amplitude, in

	dBFS.
UNIT_PERCENTFS	Specifies that the threshold specified is an absolute amplitude, in % FS (percentage of full scale).
UNIT_FFS	Specifies that the threshold specified is an absolute amplitude, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the threshold specified is an absolute amplitude, in Hex.
UNIT_V	Specifies that the threshold specified is an absolute amplitude, in V.
UNIT_DBU	Specifies that the threshold specified is an absolute amplitude, in dBu.
UNIT_DBV	Specifies that the threshold specified is an absolute amplitude, in dBV.
UNIT_DBM	Specifies that the threshold specified is an absolute amplitude, in dBm.
UNIT_W	Specifies that the threshold specified is an absolute amplitude, in W.
UNIT_DBR	Specifies that the threshold specified is an absolute amplitude, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the threshold specified is an absolute amplitude, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the threshold specified is in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the threshold specified is a percentage of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the threshold specified is an absolute amplitude, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the threshold specified is relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the threshold specified is relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the threshold specified is relative to the amplitude of the other channel.

5.3.6.2.13 FFTD_GetBuffer

bRet = FFTD_GetBuffer (sBuffer, sChannel, lStartIndex, lEndIndex, sUnit, sRelativity, pBuffer)

This method reads the specified buffer of data into an array. This array can then be used directly to perform calculations on. This is much faster than repeatedly calling a function to access each value from the buffer in turn.

NB: **This method will fail if no FFT has yet been done.**

Parameters

sBuffer	Specifies which buffer to read. It can have any of the values specified in the Buffers section below.
sChannel	Specifies which channel's buffer to read. For possible values, see the Channels section below.
lStartIndex	Specifies the index of the first item to read from the buffer.
lEndIndex	Specifies the index of the last item to read from the buffer.
sUnit	The unit to get the buffer values in. For a list of possible values, see the Units section below.
sRelativity	The relativity of the values to be returned. For a list of possible values, see the Relativity section below.
pBuffer	The buffer of values to retrieve. This must be defined as an array of the correct number of values (see Example, below).

Return value

This method returns **True** if the buffer was read successfully, **False** otherwise.

Buffers

The following values are valid for the ***sBuffer*** parameter.

FFTD_BUFFER_SAMPLE	Specifies that the buffer to be read is the sample buffer, before FFT calculations
FFTD_BUFFER_FFT_UNFILTERED	Specifies that the buffer to be read is the FFT buffer, after FFT calculations but before any filters have been applied.
FFTD_BUFFER_FFT_PREWEIGHTED	The FFT buffer, after FFT calculations and after pre-weighting, but before any FFT Detectors have been applied. If pre-weighting is not turned on in the FFT Parameters (See FFTP_WeightingFilter), this will use the unfiltered buffer.
FFTD_BUFFER_FFT_FILTERED	Specifies that the buffer to be read is the FFT buffer, after FFT calculations and filters have been applied.
FFTD_BUFFER_PHASE	Specifies that the buffer to be read is the buffer containing phase information from the FFT (can currently ONLY be returned in radians, with an arbitrary phase offset). This value can only be used if the FFTP_CalcPhaseInfo property has been set to **True**.
FFTD_BUFFER_CTA	Specifies that the buffer to be read is the *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).
FFTD_BUFFER_CTA_FFT	Specifies that the buffer to be read is the FFT of the *output* buffer from the Continuous-Time Analyzer (i.e. after Continuous-Time Detector filters have been applied).

NB: **Access to the FFTD_BUFFER_CTA and FFTD_BUFFER_CTA_FFT will fail if these buffers are not currently being displayed on the Trace window.**

Channels

The following values are valid for the ***sChannel*** parameter.

CHANNEL_A	Specifies that channel A's buffer should be read.
CHANNEL_B	Specifies that channel B's buffer should be read.

Units

If the ***sRelativity*** parameter is set to **FFTD_RELATIVITY_ABS**, the following values are allowed for the ***sUnit*** parameter:

UNIT_DBFS	Specifies that the values returned should be absolute amplitudes, in dBFS.
UNIT_PERCENTFS	Specifies that the values returned should be an absolute amplitudes, in %FS (percentage of full scale).
UNIT_FFS	Specifies that the values returned should be an absolute amplitudes, in FFS (fraction of full scale).
UNIT_HEX	Specifies that the values returned should be an absolute amplitudes, in Hex.
UNIT_V	Specifies that the values returned should be an absolute amplitudes, in V.
UNIT_DBU	Specifies that the values returned should be an absolute amplitudes, in dBu.
UNIT_DBV	Specifies that the values returned should be an absolute amplitudes, in dBV.
UNIT_DBM	Specifies that the values returned should be an absolute amplitudes, in dBm.
UNIT_W	Specifies that the values returned should be an absolute amplitudes, in W.
UNIT_DBR	Specifies that the values returned should be an absolute amplitudes, in dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Specifies that the values returned should be an absolute amplitudes, as a percentage of the reference amplitude (SA_RefAmpl).

If the ***sRelativity*** parameter is set to any other value (i.e. relative to another amplitude), then the following values are allowed for the ***sUnit*** parameter:

UNIT_RELATIVE_DB	Specifies that the values returned should be in dB, relative to the value indicated by the ***sRelativity*** parameter.
UNIT_RELATIVE_PERCENT	Specifies that the values returned should be percentages of the value indicated by the ***sRelativity*** parameter.

Relativity

The following values are valid for the ***sRelativity*** parameter:

FFTD_RELATIVITY_ABS	Specifies that the values returned should be absolute, in the unit specified by the ***sUnit*** parameter.
FFTD_RELATIVITY_SELF	Specifies that the values returned should be relative to the incoming signal on channel A, as read by the Signal Analyzer (See SA_ChARMSAmpl).
FFTD_RELATIVITY_GEN	Specifies that the values returned should be relative to the generated amplitude for channel A.
FFTD_RELATIVITY_CHANNEL	Specifies that the values returned should be relative to the incoming RMS amplitude on the opposite channel.

Example

The following example reads the first 100 values from the filtered FFT buffer.

```
Dim Buffer(100)
FFTDetector.FFTD_GetBuffer(FFTD_BUFFER_FFT_FILTERED, CHANNEL_A, 0, 99, UNIT_DBFS,
```

FFTD_RELATIVITY_ABS, Buffer)

5.4 Generator

The Generator section of this reference contains details of all the properties and methods of the following areas of the dScope:

Signal Generator

The generated Channel Status is covered in the Channel Status section.

5.4.1 Signal Generator

The Signal Generator section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"SignalGenerator."**

Properties

SG_GenMode
SG_ChAOn
SG_ChAPhaseInvert
SG_ChAFunction
SG_ChAAmpl
SG_ChAAmplUnit
SG_ChAFreq
SG_ChAFreqUnit
SG_ChADutyCycle
SG_ChADutyCycleUnit
SG_ChAPolarity
SG_ChAUserWaveform
SG_ChAUserWaveformRepeat
SG_ChA2ndFreqOffset
SG_ChA2ndFreq
SG_ChA2ndAmplOffset
SG_ChA2ndAmpl
SG_ChAPulseNumMarks
SG_ChAPulseSpacePeriod
SG_ChABurstMode
SG_ChABurstAmplDuration
SG_ChABurst2ndAmplDuration
SG_ChABurstNumPeriods
SG_ChABurstSpacePeriod
SG_ChANumSamples
SG_ChAStartFreq
SG_ChAStopFreq
SG_ChALog
SG_ChATrailSpace
SG_ChARampUp
SG_ChARampDown
SG_ChASweptSineUnit
SG_ChAPink
SG_ChAPhases
SG_ChBOn
SG_ChBPhaseInvert
SG_ChBFunction

SG_ChBAmpl
SG_ChBAmplUnit
SG_ChBFreq
SG_ChBFreqUnit
SG_ChBDutyCycle
SG_ChBDutyCycleUnit
SG_ChBPolarity
SG_ChBUserWaveform
SG_ChBUserWaveformRepeat
SG_ChB2ndFreqOffset
SG_ChB2ndFreq
SG_ChB2ndAmplOffset
SG_ChB2ndAmpl
SG_ChBPulseNumMarks
SG_ChBPulseSpacePeriod
SG_ChBBurstMode
SG_ChBBurstAmplDuration
SG_ChBBurst2ndAmplDuration
SG_ChBBurstNumPeriods
SG_ChBBurstSpacePeriod
SG_ChBNumSamples
SG_ChBStartFreq
SG_ChBStopFreq
SG_ChBLog
SG_ChBTrailSpace
SG_ChBRampUp
SG_ChBRampDown
SG_ChBSweptSineUnit
SG_ChBPink
SG_ChBPhases
SG_RefAmpl
SG_ChARefAmpl
SG_ChBRefAmpl
SG_RefAmplTied
SG_RefAmplUnit
SG_RefFreq
SG_RefImpedance
SG_SPLRef
SG_SPLRefUnit
SG_AmplStepMode
SG_AmplStep
SG_FreqStepMode
SG_FreqStep
SG_DALineUp
SG_DALineUpUnit

Methods

SG_ChACopy
SG_ChBCopy
SG_RefAmplFromChA
SG_RefAmplFromChB
SG_RefFreqFromChA
SG_RefFreqFromChB
SG_UserWaveformPlay

5.4.1.1 Properties

5.4.1.1.1 SG_GenMode

Description

This property allows selection of the current generator mode for the dScope.

Values

SG_GENMODE_TIED	Sets the Signal Generator to work in "tied" mode, i.e. both channels use the same output signal.
SG_GENMODE_SPLIT	Sets the Signal Generator to work in "split" mode, i.e. each channel's settings can be specified independently of the other channel.

5.4.1.1.2 SG_ChAOn

Description

This property is used to turn channel A of the Signal Generator on or off.

Values

True	Turns on channel A of the Signal Generator.
False	Turns off channel A of the Signal Generator.

NB:

This property turns off channel A of both the digital and Analogue Outputs. Channel A of the Digital Outputs or the Analogue Outputs can be independently muted using DO_MuteChA or AO_MuteChA respectively.

Options selected for the Digital Outputs (for example, dither) will still be generated unless the Digital Outputs are muted as well as turning the signal off.

If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's signal will be turned on or off with channel A.

5.4.1.1.3 SG_ChAPhaseInvert

Description

This property is used to invert the polarity of the signal on channel A of the Signal Generator.

Values

0 (or False)	Do not phase-invert the signal on channel A.
1 (or True)	Phase-invert the signal on channel A.

NB:

The polarity of the signal on channel B is determined by the SG_ChBPhaseInvert property. This may be tied to the phase inversion of channel A, or independent, and unlike other Signal Generator properties, is not dependent on the generator mode (SG_GenMode).

5.4.1.1.4 SG_ChAFunction

Description

This property allows selection of the current generator function, i.e. the waveform to be generated, for channel A of the Signal Generator.

NB: **The selected function will determine which fields are shown and hidden on the Signal Generator dialogue box, and also which script-controlled properties are relevant.**

Values

SG_FUNCTION_SINE	Causes channel A of the Signal Generator to generate a sine wave.
SG_FUNCTION_SQUARE_ANALYTICAL	Causes channel A of the Signal Generator to generate an "analytical" square wave - i.e. the square wave is not band-limited, and contains only two different sample values.
SG_FUNCTION_RAMP	Causes channel A of the Signal Generator to generate a ramp.
SG_FUNCTION_BURST	Causes channel A of the Signal Generator to generate a sine burst of a specified number of periods, followed by silence for a given period.
SG_FUNCTION_WHITENOISE	Causes channel A of the Signal Generator to generate white noise.
SG_FUNCTION_PINKNOISE	Causes channel A of the Signal Generator to generate pink noise (6dB/octave).
SG_FUNCTION_PULSE	Causes channel A of the Signal Generator to generate a pulse of a specified number of samples, followed by sample values of zero for a given period.
SG_FUNCTION_SWEPTSINE	Causes channel A of the Signal Generator to generate a Swept sine (chirp) signal.
SG_FUNCTION_BINCENTRES	Causes channel A of the Signal Generator to generate a signal containing a tone in the centre of every FFT bin. NB: To successfully analyze this signal, the number of FFT points (FFTP_NumPoints) will need to be set to the same number of samples as the signal (SG_ChANumSamples), and the Window function (FFTP_WindowFunction) will need to be set to "None".
SG_FUNCTION_TWINTONE	Causes channel A of the Signal Generator to generate a twin-tone signal.
SG_FUNCTION_USER	Causes channel A of the Signal Generator to generate a user-defined waveform, as specified by SG_ChAUserWaveform.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's function will be set to the same as channel A.**

5.4.1.1.5 SG_ChAUserWaveform

Description

This property allows selection of a user-defined waveform as the output for channel A of the Signal Generator.

NB: **The Signal Generator function (SG_ChAFunction) must be set to SG_FUNCTION_USER for this property to be used.**

Values

Any valid file name can be used, enclosed in double quotation marks ("..."). This should be the file name of a dScope user-defined wavetable (*.wfm) or a dScope script that is used to create a wavetable (*.dss).

Note that using the wavetable file rather than a script is quicker to load, but less flexible, as a script can query other aspects of the dScope's settings (for example, Digital Output frame rate) as it creates the table.

If a full path name is specified, the system will look for this exact file. If a file name only is specified, then the system will look in the "User Wavetables" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default).

If necessary, the system will automatically append the correct filename extension (".wfm" for user-defined waveform files).

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's user waveform will be set to the same as channel A.**

If the waveform specified is a script, this still enables different signals on each channel, since the script can ask the Signal Generator which channel it is running for, and create a different signal for each channel if necessary.

5.4.1.1.6 SG_ChAUserWaveformRepeat

Description

This property allows specification of the number of times to play the selected waveform on channel A.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to User waveform (SG_FUNCTION_USER) or Swept sine (SG_FUNCTION_SWEPTSINE).**

Values

The user waveform repeat count is represented as a short integer value. Any value from 1 to 128 can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's waveform repeat count will be set to the same as channel A.**

5.4.1.1.7 SG_ChAAmpl

Description

This property allows specification of the amplitude of the signal to be generated on channel A.

The value must be specified in the unit selected by SG_ChAAmplUnit.

Values

The channel A amplitude is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's amplitude will be set to the same as channel A.**

5.4.1.1.8 SG_ChAAmplUnit

Description

This property allows selection of the unit for the amplitude of the signal on channel A of the Signal Generator (specified using SG_ChAAmpl).

NB: **If the unit is a digital unit (dBFS, Hex etc) then the digital signal will be generated at this amplitude and the analogue signal will be generated at the amplitude implied by the current D/A line-up (SG_DALineUp). If specified as an analogue unit (V, dBu, etc) then the analogue signal will be generated at this amplitude and the digital signal at the amplitude implied by the D/A line-up.**

NB: **The channel A and channel B amplitude units are ganged together - so setting one will also automatically set the other to the same unit.**

Values

UNIT_DBFS	Sets the channel A amplitude unit to dBFS.
UNIT_PERCENTFS	Sets the channel A amplitude unit to %FS (percentage of full scale).
UNIT_FFS	Sets the channel A amplitude unit to FFS (fraction of full scale).
UNIT_HEX	Sets the channel A amplitude unit to Hex.
UNIT_VRMS	Sets the channel A amplitude unit to an RMS voltage.
UNIT_VP	Sets the channel A amplitude unit to a peak voltage.
UNIT_VPP	Sets the channel A amplitude unit to a peak-to-peak voltage.
UNIT_DBU	Sets the channel A amplitude unit to dBu.
UNIT_DBV	Sets the channel A amplitude unit to dBV.
UNIT_DBM	Sets the channel A amplitude unit to dBm.
UNIT_W	Sets the channel A amplitude unit to W.
UNIT_DBSPL	Sets the channel A amplitude unit to dBSPL.

NB: **When an RMS amplitude unit is specified, the dScope will calculate its peak output amplitude on the assumption that the signal is a sine wave. In this way, changing the Signal Generator function will leave the peak output amplitude the same.**

5.4.1.1.9 SG_ChAFreq

Description

This property allows specification of the frequency of the signal to be generated on channel A.

The value must be specified in the unit selected by SG_ChAFreqUnit.

NB: **This property is ignored unless the selected function (SG_ChAFunction) requires a frequency (sine, square, ramp, burst or twin-tone).**

Values

The channel A frequency is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's frequency will be set to the same as channel A.**

5.4.1.1.10 SG_ChAFreqUnit

Description

This property allows selection of the unit for the frequency of the signal on channel A of the Signal Generator (specified using SG_ChAFreq).

NB: **The channel A and channel B frequency units are ganged together - so setting one will also automatically set the other to the same unit.**

Values

UNIT_FREQ_HZ	Sets the channel A frequency unit to Hz.
UNIT_FREQ_OFFSET	Sets the channel A frequency unit to an offset from the reference frequency (See SG_RefFreq).
UNIT_FREQ_RATIO	Sets the channel A frequency unit to a ratio of the reference frequency (See SG_RefFreq)

5.4.1.1.11 SG_ChADutyCycle

Description

This property allows specification of the duty cycle of the signal to be generated on channel A.

The value must be specified in the unit selected by SG_ChADutyCycleUnit.

NB: **This property is ignored unless the selected function (SG_ChAFunction) requires a duty cycle (square or ramp).**

Values

The channel A duty cycle is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's duty cycle will be set to the same as channel A.**

5.4.1.1.12 SG_ChADutyCycleUnit

Description

This property allows selection of the unit for the duty cycle of the signal on channel A of the Signal Generator (specified using SG_ChADutyCycle).

NB: **This property is ignored unless the selected function (SG_ChAFunction) requires a duty cycle (square or ramp).**

Values

UNIT_DUTYCYCLE_PERCENT	Sets the channel A duty cycle unit to percent.
UNIT_DUTYCYCLE_SAMPLES	Sets the channel A duty cycle unit to a number of Digital Output samples.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's duty cycle unit will be set to the same as channel A.**

5.4.1.1.13 SG_ChAPolarity

Description

This property allows specification of the polarity of the signal to be generated on channel A.

NB: **This property is ignored unless the selected function (SG_ChAFunction) requires a polarity (square, ramp or pulse).**

Values

SG_POLARITY_NEG	Specifies that the channel A signal should have negative polarity.
SG_POLARITY_BOTH	Specifies that the channel A signal should have both negative and positive polarity.
SG_POLARITY_POS	Specifies that the channel A signal should have positive polarity.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's polarity will be set to the same as channel A.**

5.4.1.1.14 SG_ChA2ndFreqOffset

Description

This property allows specification of how the second frequency of a twin-tone signal on channel A should be calculated with respect to the first frequency.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to twin-tone.**

Values

SG_2NDFREQOFFSET_ABS	Specifies that channel A's second frequency should be treated as an absolute frequency, in Hz.
SG_2NDFREQOFFSET_OFFSET	Specifies that channel A's second frequency should be treated as an offset from the first frequency, in Hz.
SG_2NDFREQOFFSET_RATIO	Specifies that channel A's second frequency should be treated as a ratio of the first frequency.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's second frequency offset will be set to the same as channel A.**

5.4.1.1.15 SG_ChA2ndFreq

Description

This property allows specification of the second frequency of a twin-tone signal on channel A.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to twin-tone.**

Values

If the second frequency offset (SG_ChA2ndFreqOffset) is set to be a ratio (**SG_2NDFREQOFFSET_RATIO**), then this value must be specified as a ratio between 0.01 and 100.0.

If the second frequency offset is set to be an absolute or offset value (**SG_2NDFREQOFFSET_ABS** or **SG_2NDFREQOFFSET_OFFSET**), then this value must be specified in Hz.

The channel A second frequency is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's second frequency will be set to the same as channel A.**

5.4.1.1.16 SG_ChA2ndAmplOffset

Description

This property allows specification of how the second amplitude of a twin-tone signal on channel A should be calculated with respect to the first frequency.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to twin-tone.**

Values

SG_2NDAMPLOFFSET_ABS	Specifies that channel A's second amplitude should be treated as an absolute amplitude, in the unit specified by SG_ChAAmplUnit.
SG_2NDAMPLOFFSET_OFFSET	Specifies that channel A's second amplitude should be treated as an offset from the first amplitude, in the unit specified by SG_ChAAmplUnit.
SG_2NDAMPLOFFSET_RATIO	Specifies that channel A's second amplitude should be treated as a ratio of the first amplitude.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's second amplitude offset will be set to the same as channel A.**

5.4.1.1.17 SG_ChA2ndAmpl

Description

This property allows specification of the second amplitude of a twin-tone signal on channel A.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to twin-tone.**

Values

If the second amplitude offset (SG_ChA2ndAmplOffset) is set to be a ratio (**SG_2NDAMPLOFFSET_RATIO**), then this value must be specified as a ratio between 0.01 and 100.0.

If the second amplitude offset is set to be an absolute or offset value (**SG_2NDAMPLOFFSET_ABS** or **SG_2NDAMPLOFFSET_OFFSET**), then this value must be specified in the unit selected by SG_ChAAmplUnit.

The channel A second amplitude is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's second amplitude will be set to the same as channel A.**

5.4.1.1.18 SG_ChAPulseNumMarks

Description

This property allows specification of the length of the pulse signal to be generated on channel A, in samples.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to pulse.**

Values

The channel A pulse length is represented as a short integer value. Any value between 1 and 50 can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's pulse length will be set to the same as channel A.**

5.4.1.1.19 SG_ChAPulseSpacePeriod

Description

This property allows specification of the period of space between pulse signals to be generated on channel A, in samples.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to pulse.**

Values

The channel A space period is represented as a short integer value. Any value between 1 and 512k can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's pulse length will be set to the same as channel A.**

5.4.1.1.20 SG_ChABurstMode

Description

This property allows specification of whether the duration of each amplitude of a burst signal on channel A should be entered as a number of periods, or a time in ms. This affects the properties SG_ChABurstAmplDuration and SG_ChABurst2ndAmplDuration.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to burst.**

Values

SG_BURSTMODE_NUMPERIODS	Specifies that the duration of each amplitude of a burst signal on channel A should be entered as a number of periods of the signal.
SG_BURSTMODE_TIMEPERIOD	Specifies that the duration of each amplitude of a burst signal on channel A should be entered as a time, in ms.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's burst mode will be set to the same as channel A.**

5.4.1.1.21 SG_ChABurstAmplDuration

Description

This property allows specification of the duration of the first amplitude of a burst signal on channel A, in number of periods of the signal or in ms (as determined by the burst mode, SG_ChABurstMode).

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to burst.**

Values

The duration of the first amplitude of the channel A burst signal is represented as a long integer value. Any value between 1 and 1000 periods, or 1 and 5000 ms, can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then the duration of the first amplitude of channel B's burst signal will be set to the same as channel A.**

5.4.1.1.22 SG_ChABurst2ndAmplDuration

Description

This property allows specification of the duration of the second amplitude of a burst signal on channel A, in ms or number of signal periods (as determined by the burst mode, SG_ChABurstMode).

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to burst.**

Values

The duration of the second amplitude of the channel A burst signal is represented as a long integer value. Any value between 1 and 1000 periods, or 1 and 5000 ms, can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then the duration of the second amplitude of channel B's burst signal will be set to the same as channel A.**

5.4.1.1.23 SG_ChABurstNumPeriods

Description

This property allows specification of the number of periods of sine burst signal to be generated on channel A.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to burst.**

Values

The number of channel A burst periods is represented as a short integer value. Any value between 1 and 1000 can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's number of burst periods will be set to the same as channel A.**

5.4.1.1.24 SG_ChABurstSpacePeriod

Description

This property allows specification of the period of space between burst signals to be generated on channel A, in ms.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to burst.**

Values

The channel A space period is represented as a short integer value. Any value between 1 and 5000 can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's space period will be set to the same as channel A.**

5.4.1.1.25 SG_ChANumSamples

Description

This property allows selection of the number of samples to use for the Swept sine or Bin centres functions on channel A.

Values

SG_NUMSAMPLES_1K	Selects the signal size to be 1k (1024) samples.
SG_NUMSAMPLES_2K	Selects the signal size to be 2k (2048) samples.
SG_NUMSAMPLES_4K	Selects the signal size to be 4k (4096) samples.
SG_NUMSAMPLES_8K	Selects the signal size to be 8k (8192) samples.
SG_NUMSAMPLES_16K	Selects the signal size to be 16k (16384) samples.
SG_NUMSAMPLES_32K	Selects the signalsize to be 32k (32768) samples.
SG_NUMSAMPLES_64K	Selects the signal size to be 64k (65536) samples.
SG_NUMSAMPLES_128K	Selects the signal size to be 128k (131072) samples.
SG_NUMSAMPLES_256K	Selects the signal size to be 256k (262144) samples.

NB: **For best results, the number of samples selected should be the same as the number of FFT points (FFTP_NumPoints) that will be used for analysis.**

If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's number of samples will be set to the same as channel A.

5.4.1.1.26 SG_ChAStartFreq

Description

This property allows specification of the start frequency for a Swept sine or Bin centres signal on channel A, in Hz.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to Swept sine or Bin centres.**

Values

The start frequency is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's start frequency will be set to the same as channel A.**

5.4.1.1.27 SG_ChAStopFreq

Description

This property allows specification of the stop frequency for a Swept sine or Bin centres signal on channel A, in Hz.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to Swept sine or Bin centres.**

Values

The stop frequency is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's stop frequency will be set to the same as channel A.**

5.4.1.1.28 SG_ChALog

Description

This property is used to specify whether the frequencies of the Swept sine signal on channel A should increase linearly or logarithmically.

Values

False	Increase Swept sine frequencies linearly.
True	Increase Swept sine frequencies logarithmically (default).

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's log flag will be set to the same as channel A.**

5.4.1.1.29 SG_ChATrailSpace

Description

This property allows specification of the trailing space after the end of the Swept sine signal on channel A. It is entered in the unit specified by SG_ChASweptSineUnit.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to Swept sine.**

Values

The trailing space is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's trailing space will be set to the same as channel A.**

5.4.1.1.30 SG_ChARampUp

Description

This property allows specification of the time at the start of a Swept sine signal on channel A for the signal to reach its full amplitude. It is entered in the unit specified by SG_ChASweptSineUnit.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to Swept sine.**

Values

The ramping up period is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's ramping up period will be set to the same as channel A.**

5.4.1.1.31 SG_ChARampDown

Description

This property allows specification of the time at the end of a Swept sine signal on channel A for the signal to get from its full amplitude down to zero. It is entered in the unit specified by SG_ChASweptSineUnit.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to Swept sine.**

Values

The ramping down period is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's ramping down period will be set to the same as channel A.**

5.4.1.1.32 SG_ChASweptSineUnit

Description

This property allows selection of the unit for entry of trailing space and ramping up or down period for a Swept sine signal on channel A of the Signal Generator.

Values

UNIT_MS	Sets the unit for parameters of the Swept sine signal to ms.
UNIT_SAMPLES	Sets the unit for parameters of the Swept sine signal to samples.

NB: **Convrsion between these units uses the currently selected ANALOGUE sample rate (See AI_SampleRate).**

5.4.1.1.33 SG_ChAPink

Description

This property is used to specify whether the Bin centres signal generated on channel A should have a pink or white frequency response.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to Bin centres.**

Values

False	Bin centres signal has a pink frequency response of 3dB per octave.
True	Bin centres signal has a white (flat) frequency response.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's frequency response will be set to the same as channel A.**

5.4.1.1.34 SG_ChAPhases

Description

This property allows specification of the phases for generation of the Bin centres signal on channel A.

NB: **This property is ignored unless the selected function (SG_ChAFunction) is set to Bin centres.**

Values

SG_PHASES_RANDOM	Generates the Bin centres signal with random phases. This makes the signal look like white noise.
SG_PHASES_NEWMAN	Generates the Bin centres signal with phases suggested by D J Newman for minimization of crest factor. This has the advantage of producing a signal with minimal crest factor (and therefore a higher maximum amplitude), but the resulting signal is no longer noise-like.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's phases will be set to the same as channel A.**

5.4.1.1.35 SG_ChBOn

Description

This property is used to turn channel B of the Signal Generator on or off.

Values

True	Turns on channel B of the Signal Generator.
False	Turns off channel B of the Signal Generator.

NB:

This property turns off channel B of both the digital and Analogue Outputs. Channel B of the Digital Outputs or the Analogue Outputs can be independently muted using DO_MuteChB or AO_MuteChB respectively.

Options selected for the Digital Outputs (for example, dither) will still be generated unless the Digital Outputs are muted as well as turning the signal off.

If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's signal will be turned on or off with channel B.

5.4.1.1.36 SG_ChBPhaseInvert

Description

This property is used to invert the polarity of the signal on channel B of the Signal Generator.

Values

0 (or False)	Do not phase-invert the signal on channel B.
1 (or True)	Phase-invert the signal on channel B.
2	Set the phase inversion of channel B to follow channel A.

NB:

This is the *only* property of channel B of the Signal Generator that is NOT tied to channel A if the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED.

5.4.1.1.37 SG_ChBFunction

Description

This property allows selection of the current generator function, i.e. the waveform to be generated, for channel B of the Signal Generator.

NB:

The selected function will determine which fields are shown and hidden on the Signal Generator dialogue box, and also which script-controlled properties are relevant.

Values

SG_FUNCTION_SINE	Causes channel B of the Signal Generator to generate a

	sine wave.
SG_FUNCTION_SQUARE_ANALYTICAL	Causes channel B of the Signal Generator to generate an "analytical" square wave - i.e. the square wave is not band-limited, and contains only two different sample values.
SG_FUNCTION_RAMP	Causes channel B of the Signal Generator to generate a ramp.
SG_FUNCTION_BURST	Causes channel B of the Signal Generator to generate a sine burst of a specified number of periods, followed by silence for a given period.
SG_FUNCTION_WHITENOISE	Causes channel B of the Signal Generator to generate white noise.
SG_FUNCTION_PINKNOISE	Causes channel B of the Signal Generator to generate pink noise (6dB/octave).
SG_FUNCTION_PULSE	Causes channel B of the Signal Generator to generate a pulse of a specified number of samples, followed by sample values of zero for a given period.
SG_FUNCTION_SWEPTSINE	Causes channel B of the Signal Generator to generate a Swept sine (chirp) signal.
SG_FUNCTION_BINCENTRES	Causes channel B of the Signal Generator to generate a signal containing a tone in the centre of every FFT bin. NB: To successfully analyze this signal, the number of FFT points (FFTP_NumPoints) will need to be set to the same number of samples as the signal (SG_ChBNumSamples), and the Window function (FFTP_WindowFunction) will need to be set to "None".
SG_FUNCTION_TWINTONE	Causes channel B of the Signal Generator to generate a twin-tone signal.
SG_FUNCTION_USER	Causes channel B of the Signal Generator to generate a user-defined waveform, as specified by SG_ChBUserWaveform.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's function will be set to the same as channel B.**

5.4.1.1.38 SG_ChBUserWaveform

Description

This property allows selection of a user-defined waveform as the output for channel B of the Signal Generator.

NB: **The Signal Generator function (SG_ChBFunction) must be set to SG_FUNCTION_USER for this property to be used.**

Values

Any valid file name can be used, enclosed in double quotation marks ("..."). This should be the file name of a dScope user-defined wavetable (*.wfm) or a dScope script that is used to create a wavetable (*.dss).

Note that using the wavetable file rather than a script is quicker to load, but less flexible, as a script can query other aspects of the dScope's settings (for example, Digital Output frame rate) as it creates the table.

If a full path name is specified, the system will look for this exact file. If a file name only is specified, then the system will look in the "User Wavetables" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default).

If necessary, the system will automatically append the correct filename extension (".wfm" for user-defined waveform files).

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's user waveform will be set to the same as channel B.**

If the waveform specified is a script, this still enables different signals on each channel, since the script can ask the generator which channel it is running for, and create a different signal for each channel if necessary.

5.4.1.1.39 SG_ChBUserWaveformRepeat

Description

This property allows specification of the number of times to play the selected waveform on channel B.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to User waveform (SG_FUNCTION_USER) or Swept sine (SG_FUNCTION_SWEPTSINE).**

Values

The user waveform repeat count is represented as a short integer value. Any value from 1 to 128 can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's waveform repeat count will be set to the same as channel B.**

5.4.1.1.40 SG_ChBAmpl

Description

This property allows specification of the amplitude of the signal to be generated on channel B.

The value must be specified in the unit selected by SG_ChBAmplUnit.

Values

The channel B amplitude is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's amplitude will be set to the same as channel B.**

5.4.1.1.41 SG_ChBAmplUnit

Description

This property allows selection of the unit for the amplitude of the signal on channel B of the Signal Generator (specified using SG_ChBAmpl).

NB: **If the unit is a digital unit (dBFS, Hex etc) then the digital signal will be generated at this amplitude and the analogue signal will be generated at the amplitude implied by the current D/A line-up (SG_DALineUp). If specified as an analogue unit (V, dBu, etc) then the analogue signal will be generated at this amplitude and the digital signal at the amplitude implied by the D/A line-up.**

NB: **The channel A and channel B amplitude units are ganged together - so setting one will also automatically set the other to the same unit.**

Values

UNIT_DBFS	Sets the channel B amplitude unit to dBFS.
UNIT_PERCENTFS	Sets the channel B amplitude unit to %FS (percentage of full scale).
UNIT_FFS	Sets the channel B amplitude unit to FFS (fraction of full scale).
UNIT_HEX	Sets the channel B amplitude unit to Hex.
UNIT_VRMS	Sets the channel B amplitude unit to an RMS voltage.
UNIT_VP	Sets the channel B amplitude unit to a peak voltage.
UNIT_VPP	Sets the channel B amplitude unit to a peak-to-peak voltage.
UNIT_DBU	Sets the channel B amplitude unit to dBu.
UNIT_DBV	Sets the channel B amplitude unit to dBV.
UNIT_DBM	Sets the channel B amplitude unit to dBm.
UNIT_W	Sets the channel B amplitude unit to W.
UNIT_DBSPL	Sets the channel B amplitude unit to dBSPL.

NB: **When an RMS amplitude unit is specified, the dScope will calculate its peak output amplitude on the assumption that the signal is a sine wave. In this way, changing the Signal Generator function will leave the peak output amplitude the same.**

5.4.1.1.42 SG_ChBFreq

Description

This property allows specification of the frequency of the signal to be generated on channel B.

The value must be specified in the unit selected by SG_ChBFreqUnit.

NB: **This property is ignored unless the selected function (SG_ChBFunction) requires a frequency (sine, square, ramp, burst or twin-tone).**

Values

The channel B frequency is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's frequency will be set to the same as channel B.**

5.4.1.1.43 SG_ChBFreqUnit

Description

This property allows selection of the unit for the frequency of the signal on channel B of the Signal Generator (specified using SG_ChBFreq).

NB: **The channel A and channel B frequency units are ganged together - so setting one will also automatically set the other to the same unit.**

Values

UNIT_FREQ_HZ	Sets the channel B frequency unit to Hz.
UNIT_FREQ_OFFSET	Sets the channel B frequency unit to an offset from the reference frequency (See SG_RefFreq).
UNIT_FREQ_RATIO	Sets the channel B frequency unit to a ratio of the reference frequency (See SG_RefFreq)

5.4.1.1.44 SG_ChBDutyCycle

Description

This property allows specification of the duty cycle of the signal to be generated on channel B.

The value must be specified in the unit selected by SG_ChBDutyCycleUnit.

NB: **This property is ignored unless the selected function (SG_ChBFunction) requires a duty cycle (square or ramp).**

Values

The channel B duty cycle is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's duty cycle will be set to the same as channel B.**

5.4.1.1.45 SG_ChBDutyCycleUnit

Description

This property allows selection of the unit for the duty cycle of the signal on channel B of the Signal Generator (specified using SG_ChBDutyCycle).

Values

UNIT_DUTYCYCLE_PERCENT	Sets the channel B duty cycle unit to percent.
UNIT_DUTYCYCLE_SAMPLES	Sets the channel B duty cycle unit to a number of Digital Output samples.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's duty cycle unit will be set to the same as channel B.**

5.4.1.1.46 SG_ChBPolarity

Description

This property allows specification of the polarity of the signal to be generated on channel B.

NB: **This property is ignored unless the selected function (SG_ChBFunction) requires a polarity (square, ramp or pulse).**

Values

SG_POLARITY_NEG	Specifies that the channel B signal should have negative polarity.
SG_POLARITY_BOTH	Specifies that the channel B signal should have both negative and positive polarity.
SG_POLARITY_POS	Specifies that the channel B signal should have positive polarity.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's polarity will be set to the same as channel B.**

5.4.1.1.47 SG_ChB2ndFreqOffset

Description

This property allows specification of how the second frequency of a twin-tone signal on channel B should be calculated with respect to the first frequency.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to twin-tone.**

Values

SG_2NDFREQOFFSET_ABS	Specifies that channel B's second frequency should be treated as an absolute frequency, in Hz.
SG_2NDFREQOFFSET_OFFSET	Specifies that channel B's second frequency should be treated as an offset from the first frequency, in Hz.
SG_2NDFREQOFFSET_RATIO	Specifies that channel B's second frequency should be treated as a ratio of the first frequency.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's second frequency offset will be set to the same as channel B.**

5.4.1.1.48 SG_ChB2ndFreq

Description

This property allows specification of the second frequency of a twin-tone signal on channel B.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to twin-tone.**

Values

If the second frequency offset (SG_ChB2ndFreqOffset) is set to be a ratio (**SG_2NDFREQOFFSET_RATIO**), then this value must be specified as a ratio between 0.01 and 100.0.

If the second frequency offset is set to be an absolute or offset value (**SG_2NDFREQOFFSET_ABS** or **SG_2NDFREQOFFSET_OFFSET**), then this value must be specified in Hz.

The channel B second frequency is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's second frequency will be set to the same as channel B.**

5.4.1.1.49 SG_ChB2ndAmplOffset

Description

This property allows specification of how the second amplitude of a twin-tone signal on channel B should be calculated with respect to the first frequency.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to twin-tone.**

Values

SG_2NDAMPLOFFSET_ABS	Specifies that channel B's second amplitude should be treated as an absolute amplitude, in the unit specified by SG_ChBAmplUnit.
SG_2NDAMPLOFFSET_OFFSET	Specifies that channel B's second amplitude should be treated as an offset from the first amplitude, in the unit specified by SG_ChBAmplUnit.
SG_2NDAMPLOFFSET_RATIO	Specifies that channel B's second amplitude should be treated as a ratio of the first amplitude.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's second amplitude offset will be set to the same as channel B.**

5.4.1.1.50 SG_ChB2ndAmpl

Description

This property allows specification of the second amplitude of a twin-tone signal on channel B.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to twin-tone.**

Values

If the second amplitude offset (SG_ChB2ndAmplOffset) is set to be a ratio (**SG_2NDAMPLOFFSET_RATIO**), then this value must be specified as a ratio between 0.01 and 100.0.

If the second amplitude offset is set to be an absolute or offset value (**SG_2NDAMPLOFFSET_ABS** or **SG_2NDAMPLOFFSET_OFFSET**), then this value must be specified in the unit selected by SG_ChBAmplUnit.

The channel B second amplitude is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's second amplitude will be the same as channel B.**

5.4.1.1.51 SG_ChBPulseNumMarks

Description

This property allows specification of the length of the pulse signal to be generated on channel B, in samples.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to pulse.**

Values

The channel B pulse length is represented as a short integer value. Any value between 1 and 50 can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's pulse length will be set to the same as channel B.**

5.4.1.1.52 SG_ChBPulseSpacePeriod

Description

This property allows specification of the period of space between pulse signals to be generated on channel B, in samples.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to pulse.**

Values

The channel B space period is represented as a short integer value. Any value between 1 and 512k can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's pulse length will be set to the same as channel B.**

5.4.1.1.53 SG_ChBBurstMode

Description

This property allows specification of whether the duration of each amplitude of a burst signal on channel B should be entered as a number of periods, or a time in ms. This affects the properties SG_ChBBurstAmplDuration and SG_ChBBurst2ndAmplDuration.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to burst.**

Values

SG_BURSTMODE_NUMPERIODS	Specifies that the duration of each amplitude of a burst signal on channel B should be entered as a number of periods of the signal.
SG_BURSTMODE_TIMEPERIOD	Specifies that the duration of each amplitude of a burst signal on channel B should be entered as a time, in ms.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's burst mode will be set to the same as channel B.**

5.4.1.1.54 SG_ChBBurstAmplDuration

Description

This property allows specification of the duration of the first amplitude of a burst signal on channel B, in number of periods of the signal or in ms (as determined by the burst mode, SG_ChBBurstMode).

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to burst.**

Values

The duration of the first amplitude of the channel B burst signal is represented as a long integer value. Any value between 1 and 1000 periods, or 1 and 5000 ms, can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then the duration of the first amplitude of channel A's burst signal will be set to the same as channel B.**

5.4.1.1.55 SG_ChBBurst2ndAmplDuration

Description

This property allows specification of the duration of the second amplitude of a burst signal on channel B, in ms or number of signal periods (as determined by the burst mode, SG_ChBBurstMode).

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to burst.**

Values

The duration of the second amplitude of the channel B burst signal is represented as a long integer value. Any value between 1 and 1000 periods, or 1 and 5000 ms, can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then the duration of the second amplitude of channel A's burst signal will be set to the same as channel B.**

5.4.1.1.56 SG_ChBBurstNumPeriods

Description

This property allows specification of the number of periods of sine burst signal to be generated on channel B.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to burst.**

Values

The number of channel B burst periods is represented as a short integer value. Any value between 1 and 1000 can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's number of burst periods will be set to the same as channel B.**

5.4.1.1.57 SG_ChBBurstSpacePeriod

Description

This property allows specification of the period of space between burst signals to be generated on channel B, in ms.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to burst.**

Values

The channel B space period is represented as a short integer value. Any value between 1 and 5000 can be entered.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's space period will be set to the same as channel B.**

5.4.1.1.58 SG_ChBNumSamples

Description

This property allows selection of the number of samples to use for the Swept sine or Bin centres functions on channel B.

Values

SG_NUMSAMPLES_1K	Selects the signal size to be 1k (1024) samples.
SG_NUMSAMPLES_2K	Selects the signal size to be 2k (2048) samples.
SG_NUMSAMPLES_4K	Selects the signal size to be 4k (4096) samples.
SG_NUMSAMPLES_8K	Selects the signal size to be 8k (8192) samples.
SG_NUMSAMPLES_16K	Selects the signal size to be 16k (16384) samples.
SG_NUMSAMPLES_32K	Selects the signalsize to be 32k (32768) samples.
SG_NUMSAMPLES_64K	Selects the signal size to be 64k (65536) samples.
SG_NUMSAMPLES_128K	Selects the signal size to be 128k (131072) samples.
SG_NUMSAMPLES_256K	Selects the signal size to be 256k (262144) samples.

NB:

For best results, the number of samples selected should be the same as the number of FFT points (FFTP_NumPoints) that will be used for analysis.

If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's number of samples will be set to the same as channel B.

5.4.1.1.59 SG_ChBStartFreq

Description

This property allows specification of the start frequency for a Swept sine or Bin centres signal on channel B, in Hz.

NB:

This property is ignored unless the selected function (SG_ChBFunction) is set to Swept sine or Bin centres.

Values

The start frequency is represented as a double-precision floating point value.

NB:

If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's start frequency will be set to the same as channel B.

5.4.1.1.60 SG_ChBStopFreq

Description

This property allows specification of the stop frequency for a Swept sine or Bin centres signal on channel B, in Hz.

NB:

This property is ignored unless the selected function (SG_ChBFunction) is set to Swept sine or Bin centres.

Values

The stop frequency is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's stop frequency will be set to the same as channel B.**

5.4.1.1.61 SG_ChBLog

Description

This property is used to specify whether the frequencies of the Swept sine signal on channel B should increase linearly or logarithmically.

Values

False	Increase Swept sine frequencies linearly.
True	Increase Swept sine frequencies logarithmically (default).

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's log flag will be set to the same as channel B.**

5.4.1.1.62 SG_ChBTrailSpace

Description

This property allows specification of the trailing space after the end of the Swept sine signal on channel B. It is entered in the unit specified by SG_ChBSweptSineUnit.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to Swept sine.**

Values

The trailing space is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's trailing space will be set to the same as channel B.**

5.4.1.1.63 SG_ChBRampUp

Description

This property allows specification of the time at the start of a Swept sine signal on channel B for the signal to reach its full amplitude. It is entered in the unit specified by SG_ChBSweptSineUnit.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to Swept sine.**

Values

The ramping up period is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's ramping up period will be set to the same as channel B.**

5.4.1.1.64 SG_ChBRampDown

Description

This property allows specification of the time at the end of a Swept sine signal on channel B for the signal to get from its full amplitude down to zero. It is entered in the unit specified by SG_ChBSweptSineUnit.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to Swept sine.**

Values

The ramping down period is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's ramping down period will be set to the same as channel B.**

5.4.1.1.65 SG_ChBSweptSineUnit

Description

This property allows selection of the unit for entry of trailing space and ramping up or down period for a Swept sine signal on channel B of the Signal Generator.

Values

UNIT_MS	Sets the unit for parameters of the Swept sine signal to ms.
UNIT_SAMPLES	Sets the unit for parameters of the Swept sine signal to samples.

NB: **Convrsion between these units uses the currently selected ANALOGUE sample rate (See AI_SampleRate).**

5.4.1.1.66 SG_ChBPink

Description

This property is used to specify whether the Bin centres signal generated on channel B should have a pink or white frequency response.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to Bin centres.**

Values

False	Bin centres signal has a pink frequency response of 3dB per octave.
True	Bin centres signal has a white (flat) frequency response.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's frequency response will be set to the same as channel B.**

5.4.1.1.67 SG_ChBPhases

Description

This property allows specification of the phases for generation of the Bin centres signal on channel B.

NB: **This property is ignored unless the selected function (SG_ChBFunction) is set to Bin centres.**

Values

SG_PHASES_RANDOM	Generates the Bin centres signal with random phases. This makes the signal look like white noise.
SG_PHASES_NEWMAN	Generates the Bin centres signal with phases suggested by D J Newman for minimization of crest factor. This has the advantage of producing a signal with minimal crest factor (and therefore a higher maximum amplitude), but the resulting signal is no longer noise-like.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel A's phases will be set to the same as channel B.**

5.4.1.1.68 SG_RefAmpl

Description

This property allows specification of the reference amplitude used in the dScope Generator for amplitudes specified in dBr (**UNIT_DBR**) and **% ref** (**UNIT_PERCENTREF**).

The value must be specified in the unit selected by SG_RefAmplUnit.

Changing this property will set the reference amplitude of both channel A and B (SG_ChARefAmpl and SG_ChBRefAmpl), and will also tie the reference amplitudes together (see SG_RefAmplTied).

NB: **If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the analyzer reference amplitude (SA_RefAmpl).**

Values

The reference amplitude is represented as a double-precision floating point value.

5.4.1.1.69 SG_ChARefAmpl

Description

This property allows specification of the reference amplitude used in the dScope Generator for channel A amplitudes specified in dBr (**UNIT_DBR**) and % ref (**UNIT_PERCENTREF**).

The value must be specified in the unit selected by SG_RefAmplUnit.

NB:

If the reference amplitudes are tied together (see SG_RefAmplTied), then changing this property will also change the channel B reference amplitude (SG_ChBRefAmpl).

If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the analyzer reference amplitude (SA_RefAmpl).

Values

The reference amplitude is represented as a double-precision floating point value.

5.4.1.1.70 SG_ChBRefAmpl

Description

This property allows specification of the reference amplitude used in the dScope Generator for channel B amplitudes specified in dBr (**UNIT_DBR**) and % ref (**UNIT_PERCENTREF**).

The value must be specified in the unit selected by SG_RefAmplUnit.

NB:

If the reference amplitudes are tied together (see SG_RefAmplTied), then changing this property will also change the channel A reference amplitude (SG_ChARefAmpl).

If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the analyzer reference amplitude (SA_RefAmpl).

Values

The reference amplitude is represented as a double-precision floating point value.

5.4.1.1.71 SG_RefAmplTied

Description

This property allows the Generator reference amplitudes to be tied together. This means that changing the channel A reference amplitude (SG_ChARefAmpl) will automatically update channel B's reference amplitude (SG_ChBRefAmpl) to be the same, and vice versa.

NB:

If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the equivalent option on the Signal Analyzer (see SA_RefAmplTied).

Values

True	Specifies that generator reference amplitudes should be tied together.
False	Specifies that generator reference amplitudes should be separate (not tied together).

5.4.1.1.72 SG_RefAmplUnit

Description

This property allows selection of the unit for the reference amplitude used by the Generator, as specified using SG_RefAmpl.

NB: **If the Options settings are set up to lock together the reference amplitude of the generator and analyzer (See OPT_LockdBr), then changing this property will also change the analyzer reference amplitude's unit (SA_RefAmplUnit).**

Values

UNIT_DBFS	Sets reference amplitude unit to dBFS.
UNIT_PERCENTFS	Sets reference amplitude unit to %FS (percentage of full scale).
UNIT_FFS	Sets reference amplitude unit to FFS (fraction of full scale).
UNIT_HEX	Sets reference amplitude unit to Hex.
UNIT_VRMS	Sets reference amplitude unit to an RMS voltage.
UNIT_VP	Sets reference amplitude unit to a peak voltage.
UNIT_VPP	Sets reference amplitude unit to a peak-to-peak voltage.
UNIT_DBU	Sets reference amplitude unit to dBu.
UNIT_DBV	Sets reference amplitude unit to dBV.
UNIT_DBM	Sets reference amplitude unit to dBm.
UNIT_W	Sets reference amplitude unit to W.
UNIT_DBSPL	Sets reference amplitude unit to dBSPL.

NB: **If the reference amplitude is specified as an RMS voltage, but the generated signal is specified in a peak unit (or vice-versa), then the dScope assumes that the signal is a sine wave for purposes of conversion between RMS and peak values.**

5.4.1.1.73 SG_RefFreq

Description

This property allows specification of the reference frequency used in the dScope Generator for amplitudes relative to the reference frequency (**UNIT_FREQ_OFFSET** and **UNIT_FREQ_RATIO**).

The reference frequency is specified in Hz.

NB: **If the Options settings are set up to lock together the reference frequency of the generator and analyzer (See OPT_LockRefFreq), then changing this property will also change the analyzer reference frequency (SA_RefFreq).**

Values

The reference frequency is represented as a double-precision floating point value.

5.4.1.1.74 SG_RefImpedance

Description

This property allows specification of the reference impedance used throughout the dScope Generator for amplitude units that involve the impedance (dBm and W).

The reference impedance is specified in Ohms.

Values

The reference impedance is represented as a double-precision floating point value.

5.4.1.1.75 SG_dBSPLValue

Description

This property allows specification of the number of dBSPL that equates to the reference level specified using SG_SPLRef.

Values

The number of dBSPL equating to the reference level is represented as a double-precision floating point value.

5.4.1.1.76 SG_SPLRef

Description

This property allows specification of the reference used throughout the dScope generator for the dBSPL unit. The value entered is equivalent to the number of dBSPL entered using SG_dBSPLValue.

The value must be specified in the unit selected by SG_SPLRefUnit.

Values

The dBSPL reference is represented as a double-precision floating point value.

5.4.1.1.77 SG_SPLRefUnit

Description

This property allows selection of the unit for the dBSPL reference used by the generator, as specified using SG_SPLRef.

Values

UNIT_DBFS	Sets dBSPL reference unit to dBFS.
UNIT_PERCENTFS	Sets dBSPL reference unit to %FS (percentage of full scale).
UNIT_FFS	Sets dBSPL reference unit to FFS (fraction of full scale).
UNIT_HEX	Sets dBSPL reference unit to Hex.
UNIT_VRMS	Sets dBSPL reference unit to an RMS voltage.
UNIT_VP	Sets dBSPL reference unit to a peak voltage.
UNIT_VPP	Sets dBSPL reference unit to a peak-to-peak voltage.
UNIT_DBU	Sets dBSPL reference unit to dBu.
UNIT_DBV	Sets dBSPL reference unit to dBV.
UNIT_DBM	Sets dBSPL reference unit to dBm.
UNIT_W	Sets dBSPL reference unit to W.

5.4.1.1.78 SG_Gain

Description

This property allows specification of a gain for the dScope's Signal Generator. If a gain is specified, then the amplitudes entered in the Signal Generator (see SG_ChAAmpl and SG_ChBAmpl) are assumed to be post-amplifier amplitudes, i.e. the gain will be subtracted from the entered amplitudes before they are output from the dScope.

The value must be specified in the unit selected by SG_GainUnit.

Values

The gain is represented as a double-precision floating point value.

5.4.1.1.79 SG_GainUnit

Description

This property allows selection of the unit for the output amplifier gain used by the dScope's Signal Generator, as specified using SG_Gain.

Values

UNIT_RELATIVE_DB	Sets the Generator gain unit to dB.
UNIT_RELATIVE_GAIN	Sets the Generator gain unit to a gain (where 1.0 is unity gain)

5.4.1.1.80 SG_AmplStepMode

Description

This property allows specification of how the generated amplitudes will be stepped when the Ctrl + PageUp or Ctrl + PageDown Hotkey combinations are pressed.

Values

SG_AMPLSTEPMODE_OFFSET	Specifies that the amplitude step Hotkeys should change the amplitude by the given offset, as specified by SG_AmplStep.
SG_AMPLSTEPMODE_RATIO	Specifies that the amplitude step Hotkeys should change the amplitude by the given ratio, as specified by SG_AmplStep.

NB: **When the Hotkeys are pressed, the specified amplitude step will only be applied if it can successfully be applied to BOTH channels.**

5.4.1.1.81 SG_AmplStep

Description

This property allows specification of the amplitude step to apply to the generated signals when the Ctrl + PageUp or Ctrl + PageDown Hotkey combinations are pressed.

Values

If the amplitude step mode (SG_AmplStepMode) is set to be a ratio (i.e. **SG_AMPLSTEPMODE_RATIO**), then this value must be specified as a ratio between 0.01 and 100.0.

If the amplitude step mode is set to be an offset (i.e. **SG_AMPLSTEPMODE_OFFSET**), then this value must be specified in the unit selected by SG_ChAAmplUnit and SG_ChBAmplUnit.

The amplitude step is represented as a double-precision floating point value.

NB: **When the Hotkeys are pressed, the specified amplitude step will only be applied if it can successfully be applied to BOTH channels.**

5.4.1.1.82 SG_FreqStepMode

Description

This property allows specification of how the generated frequencies will be stepped when the Shift + PageUp or Shift + PageDown Hotkey combinations are pressed.

Values

SG_FREQSTEPMODE_OFFSET	Specifies that the frequency step Hotkeys should change the frequency by the given offset, as specified by SG_FreqStep.
SG_FREQSTEPMODE_RATIO	Specifies that the frequency step Hotkeys should change the frequency by the given ratio, as specified by SG_FreqStep.
SG_FREQSTEPMODE_OCTAVE	Specifies that the frequency step Hotkeys should change the frequency by an octave.
SG_FREQSTEPMODE_OCTAVE2	Specifies that the frequency step Hotkeys should change the frequency by 1/2 an octave.
SG_FREQSTEPMODE_OCTAVE3	Specifies that the frequency step Hotkeys should change the frequency by 1/3 of an octave.
SG_FREQSTEPMODE_OCTAVE4	Specifies that the frequency step Hotkeys should change the frequency by 1/4 of an octave.
SG_FREQSTEPMODE_OCTAVE6	Specifies that the frequency step Hotkeys should change the frequency by 1/6 of an octave.

SG_FREQSTEPMODE_OCTAVE12	Specifies that the frequency step Hotkeys should change the frequency by 1/12 of an octave.

NB: **When the Hotkeys are pressed, the specified frequency step will only be applied if it can successfully be applied to BOTH channels.**

5.4.1.1.83 SG_FreqStep

Description

This property allows specification of the frequency step to apply to the generated signals when the Shift + PageUp or Shift + PageDown Hotkey combinations are pressed.

Values

If the frequency step mode (SG_FreqStepMode) is set to be a ratio (**SG_FREQSTEPMODE_RATIO**), then this value must be specified as a ratio between 0.01 and 100.0.

If the frequency step mode is set to be an offset (**SG_FREQSTEPMODE_OFFSET**), then this value must be specified in Hz.

If the frequency step mode is set to one of the octave values (**SG_FREQSTEPMODE_OCTAVE .. SG_FREQSTEPMODE_OCTAVE12**), then this property is ignored.

The frequency step is represented as a double-precision floating point value.

NB: **When the Hotkeys are pressed, the specified frequency step will only be applied if it can successfully be applied to BOTH channels.**

5.4.1.1.84 SG_DALineUp

Description

This property allows specification of the D/A line-up used throughout the dScope Signal Generator.

The value must be specified in the unit selected by SG_DALineUpUnit.

NB: **If the Options settings are set up to lock together the D/A line-up of the Signal Generator and Signal Analyzer (See OPT_LockDALineUp), then changing this property will also change the Signal Analyzer D/A line-up (SA_DALineUp).**

Values

The D/A line-up amplitude is represented as a double-precision floating point value.

5.4.1.1.85 SG_DALineUpUnit

Description

This property allows selection of the unit for the D/A line-up used by the Generator, as specified using SG_DALineUp.

NB: If the Options settings are set up to lock together the D/A line-up of the generator and analyzer (See OPT_LockDALineUp), then changing this property will also change the analyzer D/A line-up's unit (SA_DALineUpUnit).

Values

UNIT_VRMS	Sets D/A line-up unit to Volts, RMS.
UNIT_VP	Sets D/A line-up unit to Volts, peak.
UNIT_VPP	Sets D/A line-up unit to Volts, peak-to-peak.
UNIT_DBU	Sets D/A line-up unit to dBu.
UNIT_DBV	Sets D/A line-up unit to dBV.
UNIT_DBM	Sets D/A line-up unit to dBm.
UNIT_W	Sets D/A line-up unit to W.
UNIT_DBSPL	Sets D/A line-up unit to dBSPL.

5.4.1.2 Methods

5.4.1.2.1 SG_ChACopy

SG_ChACopy()

This method can be used to copy the current channel A Signal Generator settings to channel B.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.4.1.2.2 SG_ChBCopy

SG_ChBCopy()

This method can be used to copy the current channel B Signal Generator settings to channel A.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.4.1.2.3 SG_RefAmplFromChA

SG_RefAmplFromChA()

This method can be used to set the Signal Generator's reference amplitude (and its unit) to the same level as the current generated signal on channel A.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.4.1.2.4 SG_RefAmplFromChB

SG_RefAmplFromChB()

This method can be used to set the Signal Generator's reference amplitude (and its unit) to the same level as the current generated signal on channel B.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.4.1.2.5 SG_RefFreqFromChA

SG_RefFreqFromChA()

This method can be used to set the Signal Generator's reference frequency to the same frequency as the current generated signal on channel A.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.4.1.2.6 SG_RefFreqFromChB

SG_RefFreqFromChB()

This method can be used to set the Signal Generator's reference frequency to the same frequency as the current generated signal on channel B.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.4.1.2.7 SG_UserWaveformPlay

SG_UserWaveformPlay(short sChannel)

This method can be used to play the currently selected signal on one or both channels of the Signal Generator the number of times specified by the relevant repeat count SG_ChAUserWaveformRepeat or SG_ChBUserWaveformRepeat.

Parameters

sChannel	The channel to play the waveform on - this can be **CHANNEL_A**, **CHANNEL_B** or **CHANNEL_BOTH**.

Return value

This method has no return value.

5.5 Channel Status

NB: This part of the dScope's scripting interface may not be available, depending on the dScope model number.

The Channel Status section of this reference contains details of all the properties and methods concerned with the generation and analysis of Channel Status.

Scripts can set all bits of the generated Channel Status independently for each channel, and read the entire Channel Status of either input channel.

The following properties and methods are available for the Channel Status:

Output Channel Status

ChAOutput
ChBOutput

Properties

CS_Tied
CS_ConsSamplingFreqAuto
CS_ConsWordLengthAuto
CS_ProfFreqLockingAuto
CS_ProfSamplingFreqAuto
CS_ProfChannelModeAuto
CS_ProfWordLengthAuto
CS_SampleTimeShowHex
CS_SampleTimeSendBCD
CS_TimeOfDayShowHex
CS_TimeOfDaySendBCD

Methods

CS_SampleTimeGetCurrent
CS_TimeOfDayGetCurrent

Input Channel Status

The following properties and methods are concerned with the Input Channel Status:

ChAInput
ChBInput

Properties

CS_SampleTimeShowHex
CS_TimeOfDayShowHex

Methods

There are no methods associated with the Input Channel Status.

5.5.1 Output Channel Status

5.5.1.1 ChAOutput

This part of the OLE interface gives access to the generated Channel Status frame for channel A of the Digital Output.

To access the individual bytes of this frame, use the syntax **ChannelStatus.ChAOutput.CS_Byte...**

NB: **If the Channel Status mode is set to tied (see CS_Tied), then any changes made to this generated frame will be reflected in the Channel Status frame for channel B.**

Properties

The following properties give access to the individual bytes of the Channel Status frame:

CS_Byte[N]
CS_CRCMode

Methods

The following methods are available for the generated Channel Status frame:

CS_SetDefault

5.5.1.2 ChBOutput

This part of the OLE interface gives access to the generated Channel Status frame for channel B of the Digital Output.

To access the individual bytes of this frame, use the syntax **ChannelStatus.ChBOutput.CS_Byte...**

NB: **If the Channel Status mode is set to tied (see CS_Tied), then any changes made to this generated frame will be reflected in the Channel Status frame for channel A.**

Properties

The following properties give access to the individual bytes of the Channel Status frame:

CS_Byte[N]
CS_CRCMode

Methods

The following methods are available for the generated Channel Status frame:

CS_SetDefault

5.5.1.3 Sample Time and Time Of Day

On the advanced Professional Channel Status windows, two implementations of the timecode fields are supported. By default, the mode defined in the AES3 standard is used; the fields simply contain a hexadecimal value indicating the number of samples past midnight. The second option is a 'BCD' mode - the 32-bit field carries eight BCD digits representing the time; this format is used as a defacto standard by some broadcasting organisations.

For example, a time of 6:30am is equivalent to 23,400 seconds past midnight, or 1,123,200,000 samples at a sample rate of 48kHz. In hexadecimal, this is 0 x 42 F2 AC 00.

If "Show Hex" is selected, these bytes will be displayed, but if a BCD display is selected by turning this option off, the bytes 06 30 00 00 will be shown as the current time.

On the other hand, if the data is actually *sent* as BCD, then 6:30am will be stored as 0 x 06 30 00 00. In this case, a hexadecimal display of the generated bytes is meaningless. The Input Channel Status however must be set as "Show Hex" to show the bytes correctly, otherwise the BCD will be interpreted as a sample count.

5.5.1.4 Properties

5.5.1.4.1 CS_Tied

Description

This property is used to tie both channels of the Output Channel Status together.

Values

True	Tie both channels of the Output Channel Status.
False	Do not tie both channels of the Output Channel Status.

5.5.1.4.2 CS_ConsSampleRateAuto

Description

This property is used for Consumer Output Channel Status, to set the sample rate field to automatic. The sample rate bits will automatically be set according to the current state of the Digital Output.

Values

True	Sets sample rate field for Consumer Output Channel Status to be automatic.
False	Sets sample rate field for Consumer Output Channel Status to no longer be automatic.

NB: **This property applies to both channels of the Output Channel Status.**

5.5.1.4.3 CS_ConsWordLengthAuto

Description

This property is used for Consumer Output Channel Status, to set the wordlength field to automatic. The wordlength bits will automatically be set according to the current state of the Digital Output.

Values

True	Sets wordlength field for Consumer Output Channel Status to be automatic.
False	Sets wordlength field for Consumer Output Channel Status to no longer be automatic.

NB: **This property applies to both channels of the Output Channel Status.**

5.5.1.4.4 CS_ProfFreqLockingAuto

Description

This property is used for Professional Output Channel Status, to set the source frequency lock field to automatic. The source frequency lock will automatically be set according to the current state of the Reference Sync source - Byte 0, bit 5 will be set to 1 if unlocked, 0 if locked.

Values

True	Sets source frequency lock field for Professional Output Channel Status to be automatic.
False	Sets source frequency lock field for Professional Output Channel Status to no longer be automatic.

NB: **This property applies to both channels of the Output Channel Status.**

5.5.1.4.5 CS_ProfSampleRateAuto

Description

This property is used for Professional Output Channel Status, to set the sample rate field to automatic. The sample rate bits will automatically be set according to the current state of the Digital Output.

Values

True	Sets sample rate field for Professional Output Channel Status to be automatic.
False	Sets sample rate field for Professional Output Channel Status to no longer be automatic.

NB: **This property applies to both channels of the Output Channel Status.**

5.5.1.4.6 CS_ProfChannelModeAuto

Description

This property is used for Professional Output Channel Status, to set the channel mode field to automatic. The channel mode bits will automatically be set according to the current state of the Digital Output.

The channel mode will usually be set to "Not indicated" unless the Digital Outputs are in Split96 mode (see DO_Split96), in which case they are set to "Single channel double fs".

Values

True	Sets channel mode field for Professional Output Channel Status to be automatic.
False	Sets channel mode field for Professional Output Channel Status to no longer be automatic.

NB: **This property applies to both channels of the Output Channel Status.**

5.5.1.4.7 CS_ProfWordLengthAuto

Description

This property is used for Professional Output Channel Status, to set the wordlength field to automatic. The wordlength bits will automatically be set according to the current state of the Digital Output.

Values

True	Sets wordlength field for Professional Output Channel Status to be automatic.
False	Sets wordlength field for Professional Output Channel Status to no longer be automatic.

NB: **This property applies to both channels of the Output Channel Status.**

5.5.1.4.8 CS_SampleTimeOutputShowHex

Description

This property allows specification of whether to show the Sample Time field as BCD (Binary Coded Decimal) or as Hex.

See Sample Time and Time Of Day for more details on these formats.

Values

True	Sets the display of the sample time on the Output Channel Status to be BCD.
False	Sets the display of the sample time on the Output Channel Status to show the hex byte values directly.

5.5.1.4.9 CS_SampleTimeSendBCD

Description

This property allows specification of whether to output the Sample Time field in the normal "samples past midnight" mode, or as a BCD (Binary Coded Decimal) timecode.

See Sample Time and Time Of Day for more details on these formats.

Values

True	Selects the output format of the sample time on the Output Channel Status to be BCD.
False	Selects the output format of the sample time on the Output Channel Status to be hex bytes.

NB: **This property applies to both channels of the Output Channel Status.**

5.5.1.4.10 CS_TimeOfDayOutputShowHex

Description

This property allows specification of whether to show the Time of Day field as BCD (Binary Coded Decimal) or as Hex.

See Sample Time and Time Of Day for more details on these formats.

Values

True	Sets the display of the time of day on the Output Channel Status to be BCD.
False	Sets the display of the time of day on the Output Channel Status to show the hex byte values directly.

5.5.1.4.11 CS_TimeOfDaySendBCD

Description

This property allows specification of whether to output the Time of Day field in the normal "samples past midnight" mode, or as a BCD (Binary Coded Decimal) timecode.

See Sample Time and Time Of Day for more details on these formats.

Values

True	Selects the output format of the time of day on the Output Channel Status to be BCD.
False	Selects the output format of the time of day on the Output Channel Status to be hex bytes.

NB: **This property applies to both channels of the Output Channel Status.**

5.5.1.5 Methods

5.5.1.5.1 CS_SampleTimeLoadCurrent

CS_SampleTimeLoadCurrent ()

This method gets the current time (in samples after midnight) and inserts it into the Sample Time field of the Output Channel Status.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.5.1.5.2 CS_TimeOfDayLoadCurrent

CS_TimeOfDayLoadCurrent ()

This method gets the current time (in samples after midnight) and inserts it into the Time of Day field of the Output Channel Status.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.5.2 Input Channel Status

5.5.2.1 ChAInput

This part of the OLE interface gives access to the received Channel Status frame for channel A of the Digital Input.

To access the individual bytes of this frame, use the syntax **ChannelStatus.ChAInput.CS_Byte...**

Properties

The following properties give access to the individual bytes of the Channel Status frame:

CS_Byte[N]

Methods

There are no methods available for the received Channel Status frame.

5.5.2.2 ChBInput

This part of the OLE interface gives access to the received Channel Status frame for channel B of the Digital Input.

To access the individual bytes of this frame, use the syntax **ChannelStatus.ChBInput.CS_Byte...**

Properties

The following properties give access to the individual bytes of the Channel Status frame:

CS_Byte[N]

Methods

There are no methods available for the received Channel Status frame.

5.5.2.3 Properties

5.5.2.3.1 CS_SampleTimeInputShowHex

Description

This property allows specification of whether to show the Sample Time field as BCD (Binary Coded Decimal) or as Hex.

See Sample Time and Time Of Day for more details on these formats.

Values

True	Sets the display of the sample time on the Input Channel Status to be BCD.
False	Sets the display of the sample time on the Input Channel Status to show the hex byte values directly.

5.5.2.3.2 CS_TimeOfDayInputShowHex

Description

This property allows specification of whether to show the Time of Day field as BCD (Binary Coded Decimal) or as Hex.

See Sample Time and Time Of Day for more details on these formats.

Values

True	Sets the display of the time of day on the Input Channel Status to be BCD.
False	Sets the display of the time of day on the Input Channel Status to show the hex byte values directly.

5.5.3 Channel Status frame

Individual Channel Status frames can be accessed using the ChAOutput, ChBOutput, ChAInput and ChBInput methods of the ChannelStatus object.

Properties

CS_Byte[N]
CS_CRCMode

Methods

The following methods are available for the Channel Status frame:

CS_SetDefault

5.5.3.1 Properties

5.5.3.1.1 CS_Byte[N]

Description

This property gives access to an individual byte of the Channel Status frame. [N] can be any number from 0 to 23, for example CS_Byte0 or CS_Byte23. This property can be read or set for Output Channel Status, and read for Input Channel Status.

Values

This property is a short integer and can have a value of 0x00 to 0xFF Hex (0 to 255).

5.5.3.1.2 CS_CRCMode

Description

This property allows selection of the mode of the CRC output byte for Professional Output Channel Status.

Values

CS_CRC_CORRECT	Sets the output CRC to be correct, based on the preceding bytes in the Channel Status.
CS_CRC_INCORRECT	Sets the output CRC to be incorrect, based on the preceding bytes in the Channel Status. The correct CRC is calculated, and then the last bit is inverted.
CS_CRC_ZERO	Sets the output CRC to be always zero.
CS_CRC_STATIC	Sets the output CRC to not change when other fields in the Channel Status are changed. The value of this field will be whatever the field was last set to, or may be the value set using the **CS_Byte23** property (See CS_Byte[N] for details).

5.5.3.2 Methods

5.5.3.2.1 CS_SetDefault

CS_SetDefault ()

This method sets the Output Channel Status to its default state.

Leaving the Consumer/Professional bit (byte 0, bit 0) as it is, the rest of the Channel Status frame will be set to zeros. Any automatic fields are then set to 'Auto' mode, wherein their value is set based on the current state of the Digital Outputs - for Consumer Channel Status, this affects the sample rate and wordlength fields; for Professional, it affects the frequency locking, sample rate, channel mode and wordlength fields.

Parameters

This method has no parameters

Return value

This method has no return value.

5.6 Automation

The Automation section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"Automation."**

Properties

There are no properties available to control the Automation part of the dScope interface.

Methods

AUT_RunScript
AUT_StopScript

The Automation section of this reference also contains details of the following parts of the dScope interface:

Event Manager

5.6.1 AUT_RunScript

AUT_RunScript (strScript)

This method runs the specified script from within another script.

Parameters

strScript The file name of the script to run. Any valid script file name can be used, enclosed in double quotation marks ("...").

If a full path name is not specified, then the system will look in the "Automation" subfolder of the folder specified in the Options dialogue box for scripts (See OPT_ScriptsFolder).

If necessary, the system will automatically append the file extension for dScope script files (".dss").

Return value

This method has no return value.

5.6.2 AUT_StopScript

AUT_StopScript ()

This method stops the currently running script.

By default, a script will remain running if it has any event handlers in it (See Events for further details). This is because the dScope does not know when these events will occur, so the script is left running to enable the events to be handled. When you no longer wish to handle these events, this method should be used to stop the script.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.6.3 Event Manager

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The Event Manager section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"EventManager."**

Properties

EM_On
EM_LogFile

Methods

EM_SetEvent
EM_EventOn

5.6.3.1 Properties

5.6.3.1.1 EM_On

Description

This property specifies whether the Event Manager is on, i.e. whether events are handled at all. This determines whether events are managed by the Event Manager or are fired to any scripts that are currently running.

Values

True	Turns on the Event Manager
False	Turns off the Event Manager

NB: **Individual events can also be turned on or off (see EM_EventOn).**

5.6.3.1.2 EM_LogFile

Description

Individual events in the Event Manager can be set to write to a log file when the event occurs. This property allows you to specify the name of the log file that should be used.

Values

Any valid file name can be used, enclosed in double quotation marks ("..."). This should be the file name of an event log file (*.log). If the file does not already exist, the dScope will create it.

If a full path name is specified, the system will look for this exact file. If a file name only is specified, then the system will look in the "Event Logs" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default).

If necessary, the system will automatically append the correct filename extension (".log" for event log files).

5.6.3.2 Methods

5.6.3.2.1 EM_SetEvent

EM_SetEvent (sEventID, bEventOn, bBeep, bLogToFile, strScriptName)

This method allows full details of an event to be set in the Event Manager.

If the details of an event are already set correctly, you can turn the event on or off by using EM_EventOn.

Parameters

sEventID	The event ID of the event whose details you wish to set. See Event IDs below for a list of valid IDs.
bEventOn	**True** to turn the event On; **False** to turn it off.
bBeep	**True** to set an audible alarm when the event occurs; **False** for no alarm.
bLogToFile	**True** to log this event's details to the log file when it occurs; **False** if you don't want to log the details.
strScriptName	The file name of a script file to run when the event occurs. Any valid file name can be used, enclosed in double quotation marks ("..."). This should be the file name of a dScope script file (*.dss). If a full path name is specified, the system will look for this exact file. If a file name only is specified, then the system will look in the "Scripts\Automation" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default). If necessary, the system will automatically append the correct filename extension (".dss" for dScope script files).

Return value

This method has no return value.

Event IDs

EM_EVENT_CHAVALIDBIT	Change of state of channel A Valid bit.
EM_EVENT_CHBVALIDBIT	Change of state of channel B Valid bit.
EM_EVENT_CARRIERINPUTLOCKING	Change of Digital Input Carrier locked state
EM_EVENT_CARRIERBIPHASE	Change of state of Digital Input Carrier biphase error indication.
EM_EVENT_CARRIERBLOCKLENGTH	Change of state of Digital Input Carrier block length error indication.
EM_EVENT_CARRIEREYENARROWING	Change of state of Digital Input Carrier eye-narrowing error indication.

EM_EVENT_CARRIERASYNC	Change of state of Digital Input Carrier asynchronous w.r.t. generator indication.
EM_EVENT_CHANNELCHECKFAILED_CHA	Change of state of channel A Channel Check failed indication.
EM_EVENT_CHANNELCHECKFAILED_CHB	Change of state of channel B Channel Check failed indication.
EM_EVENT_CSPROFBIT	Change of state of Channel Status Professional/ Consumer bit.
EM_EVENT_CSCOPYRIGHTBIT	Change of state of Consumer Channel Status copyright bit.
EM_EVENT_CSEMPHASIS	Change of state of Channel Status emphasis bits.
EM_EVENT_CSCHANNELMODE	Change of state of Professional Channel Status channel mode bits.
EM_EVENT_CSCRCERROR	Change of state of Professional Channel Status CRC error state.
EM_EVENT_CSANOTEQUALTOB	Change of state of Channel Status being different for each channel.
EM_EVENT_FFTTRIGGER	FFT trigger going off.
EM_EVENT_FFTBUFFERPROCESSED	FFT buffer has been processed.
EM_EVENT_READINGMINLIMIT	Lower limit of Reading being breached (change of state).
EM_EVENT_READINGMAXLIMIT	Upper limit of Reading being breached (change of state).
EM_EVENT_TRACEMINLIMIT	Lower limit of Trace being breached (change of state).
EM_EVENT_TRACEMAXLIMIT	Upper limit of Trace being breached (change of state).
EM_EVENT_SWEEPSTARTED	A Sweep has started.
EM_EVENT_SWEEPSTEPDONE	A Sweep step has completed.
EM_EVENT_SWEEPFINISHED	A Sweep has finished.
EM_EVENT_SWEEPSENSE	A sense Sweep has sensed a new source point.
EM_EVENT_KEYPRESS	Event key (F2) has been pressed.

5.6.3.2.2 EM_GetEvent

EM_GetEvent (sEventID, pbEventOn, pbBeep, pbLogToFile, pstrScriptName)

This method allows full details of an event to be retrieved from the Event Manager.

Parameters

sEventID	The event ID of the event whose details you wish to get. See Event IDs below for a list of valid IDs.
pbEventOn	Will be set to **True** if the event is On; **False** if it is off.
pbBeep	Will be set to **True** if the event is set to have an audible alarm when the event occurs; **False** if it is not set to have an alarm.
pbLogToFile	Will be set to **True** if this event's details are written to the log file when it occurs; **False** if nothing is logged to the file.
pstrScriptName	Will be set the the name of a script file that runs when the event occurs.

Return value

This method returns **True** if the event is found, and the returned parameters are valid, or **False** otherwise.

Event IDs

EM_EVENT_CHAVALIDBIT	Change of state of channel A Valid bit.
EM_EVENT_CHBVALIDBIT	Change of state of channel B Valid bit.
EM_EVENT_CARRIERINPUTLOCKING	Change of Digital Input Carrier locked state
EM_EVENT_CARRIERBIPHASE	Change of state of Digital Input Carrier biphase error indication.
EM_EVENT_CARRIERBLOCKLENGTH	Change of state of Digital Input Carrier block length error indication.
EM_EVENT_CARRIEREYENARROWING	Change of state of Digital Input Carrier eye-narrowing error indication.
EM_EVENT_CARRIERASYNC	Change of state of Digital Input Carrier asynchronous w.r.t. generator indication.
EM_EVENT_CHANNELCHECKFAILED_CHA	Change of state of channel A Channel Check failed indication.
EM_EVENT_CHANNELCHECKFAILED_CHB	Change of state of channel B Channel Check failed indication.
EM_EVENT_CSPROFBIT	Change of state of Channel Status Professional/ Consumer bit.
EM_EVENT_CSCOPYRIGHTBIT	Change of state of Consumer Channel Status copyright bit.
EM_EVENT_CSEMPHASIS	Change of state of Channel Status emphasis bits.
EM_EVENT_CSCHANNELMODE	Change of state of Professional Channel Status channel mode bits.
EM_EVENT_CSCRCERROR	Change of state of Professional Channel Status CRC error state.
EM_EVENT_CSANOTEQUALTOB	Change of state of Channel Status being different for each channel.
EM_EVENT_FFTTRIGGER	FFT trigger going off.
EM_EVENT_FFTBUFFERPROCESSED	FFT buffer has been processed.
EM_EVENT_READINGMINLIMIT	Lower limit of Reading being breached (change of state).
EM_EVENT_READINGMAXLIMIT	Upper limit of Reading being breached (change of state).
EM_EVENT_TRACEMINLIMIT	Lower limit of Trace being breached (change of state).
EM_EVENT_TRACEMAXLIMIT	Upper limit of Trace being breached (change of state).
EM_EVENT_SWEEPSTARTED	A Sweep has started.
EM_EVENT_SWEEPSTEPDONE	A Sweep step has completed.
EM_EVENT_SWEEPFINISHED	A Sweep has finished.
EM_EVENT_SWEEPSENSE	A sense Sweep has sensed a new source point.
EM_EVENT_KEYPRESS	Event key (F2) has been pressed.

5.6.3.2.3 EM_EventOn

EM_EventOn (sEventID, bOn)

This method allows an individual event to be turned on or off, i.e. determines whether the effects set up for this event (log to file, beep etc) happen when the event occurs.

You can alter full details of any events by using EM_SetEvent.

Parameters

sEventID	The event ID of the event whose details you wish to set. See Event IDs below for a list of valid IDs.
bOn	**True** to turn the event On; **False** to turn it off.

Return value

This method has no return value.

Event IDs

EM_EVENT_CHAVALIDBIT	Change of state of channel A Valid bit.
EM_EVENT_CHBVALIDBIT	Change of state of channel B Valid bit.
EM_EVENT_CARRIERINPUTLOCKING	Change of Digital Input Carrier locked state
EM_EVENT_CARRIERBIPHASE	Change of state of Digital Input Carrier biphase error indication.
EM_EVENT_CARRIERBLOCKLENGTH	Change of state of Digital Input Carrier block length error indication.
EM_EVENT_CARRIEREYENARROWING	Change of state of Digital Input Carrier eye-narrowing error indication.
EM_EVENT_CARRIERASYNC	Change of state of Digital Input Carrier asynchronous w.r.t. generator indication.
EM_EVENT_CHANNELCHECKFAILED_CHA	Change of state of channel A Channel Check failed indication.
EM_EVENT_CHANNELCHECKFAILED_CHB	Change of state of channel B Channel Check failed indication.
EM_EVENT_CSPROFBIT	Change of state of Channel Status Professional/ Consumer bit.
EM_EVENT_CSCOPYRIGHTBIT	Change of state of Consumer Channel Status copyright bit.
EM_EVENT_CSEMPHASIS	Change of state of Channel Status emphasis bits.
EM_EVENT_CSCHANNELMODE	Change of state of Professional Channel Status channel mode bits.
EM_EVENT_CSCRCERROR	Change of state of Professional Channel Status CRC error state.
EM_EVENT_CSANOTEQUALTOB	Change of state of Channel Status being different for each channel.
EM_EVENT_FFTTRIGGER	FFT trigger going off.
EM_EVENT_FFTBUFFERPROCESSED	FFT buffer has been processed.
EM_EVENT_READINGMINLIMIT	Lower limit of Reading being breached (change of state).
EM_EVENT_READINGMAXLIMIT	Upper limit of Reading being breached (change of state).

EM_EVENT_TRACEMINLIMIT	Lower limit of Trace being breached (change of state).
EM_EVENT_TRACEMAXLIMIT	Upper limit of Trace being breached (change of state).
EM_EVENT_SWEEPSTARTED	A Sweep has started.
EM_EVENT_SWEEPSTEPDONE	A Sweep step has completed.
EM_EVENT_SWEEPFINISHED	A Sweep has finished.
EM_EVENT_SWEEPSENSE	A sense Sweep has sensed a new source point.
EM_EVENT_KEYPRESS	Event key (F2) has been pressed.

5.7 Sweeps/Regulation

The Sweeps section of this reference contains details of all the properties and methods of the following areas of the dScope:

Sweep Setup
Sweep Settling
Regulation

5.7.1 Sweep Setup

The Sweep Setup section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"Sweep."**

Properties

SW_Result1
SW_Result1FFTDetector
SW_Result2
SW_Result2FFTDetector
SW_Result3
SW_Result3FFTDetector
SW_Result4
SW_Result4FFTDetector
SW_AlarmOn
SW_OptimizeForSpeed
SW_Append
SW_YAxisAutoZoom
SW_SourceTab

The following properties act on the Inner or Outer Sweep source, depending on the current value of the SW_SourceTab property.

SW_SweepSource
SW_StartValue
SW_StopValue
SW_Unit
SW_Interval
SW_Offset
SW_Factor
SW_SenseType
SW_SenseInterval
SW_SenseUnit

SW_SenseEndValue
SW_SenseThreshold
SW_SenseThresholdUnit
SW_TimeInterval
SW_DataTable
SW_ChannelArray
SW_StartChannel
SW_EndChannel
SW_ChannelArrayMode
SW_RunScript
SW_Script
SW_RunScriptWhen
SW_CurrentStep
SW_NumSteps
SW_Regulate
SW_XAxisAutoZoom

Methods

SW_Go
SW_Stop
SW_Pause
SW_SingleStep
SW_IsSweepFinished
SW_MinLimitBreached
SW_MaxLimitBreached
SW_SetYUnit
SW_SetYRange
SW_SetYIntervals
SW_SetMaxLimit
SW_SetMinLimit
SW_ResetYDefaults

5.7.1.1 Properties

5.7.1.1.1 SW_Append

Description

This property is used to select whether to append the next Sweep to existing Sweep Traces on the Trace Window, or to remove existing Sweep Traces before performing a new Sweep.

Values

True	Append Sweeps to existing Sweeps on the Trace window.
False	Remove existing Sweeps from the Trace window before running this sweep.

NB: **If this property is set to False, then the only Sweep Traces that will be removed before performing the next Sweep are those that were appended automatically using this property. Any Sweep Traces that have been manually copied by using the "Copy Trace" option on the Trace window will not be removed.**

5.7.1.1.2 SW_AlarmOn

Description

This property is used to select whether to set off an audible alarm to go off when the Sweep has finished.

Values

True	A beep will sound when the Sweep has finished.
False	No alarm will sound when the Sweep has finished.

5.7.1.1.3 SW_OptimizeForSpeed

Description

This property is used to select whether to optimize the Sweep for speed.

This will turn off the FFT trigger (if not required for the Sweep) and, if the input is Analogue, attempt to alter the Analogue Inputs settings to ensure that minimal auto-ranging occurs during the sweep.

Values

True	Optimize the Sweep for speed.
False	Do not optimize the Sweep for speed.

5.7.1.1.4 SW_Result[N]

Description

This property allows selection of one of the Results to be swept. Up to 4 different Results can be swept at the same time, using values of [N] from 1 to 4, i.e. SW_Result1, SW_Result2, SW_Result3, or SW_Result4.

Values

RESULT_NONE	Resets this Sweep Result so that no Result will be swept.
RESULT_DO_REFSYNCFRAMERATE	Selects the data to be swept to be the frame rate of the Ref Sync source.
RESULT_DO_REFSYNCFRAMERATEDEVIATION	Selects the data to be swept to be the deviation of the Ref Sync source's frame rate from the nearest standard rate.
RESULT_DI_FRAMERATE	Selects the data to be swept to be the frame rate of the Digital Input.
RESULT_DI_FRAMERATEDEVIATION	Selects the data to be swept to be the deviation of the Digital Input frame rate from the nearest standard rate.
RESULT_DIC_AMPL	Selects the data to be swept to be the amplitude of the Digital Input Carrier.
RESULT_DIC_JITTERAMPL	Selects the data to be swept to be the amplitude of the jitter on the Digital Input

	Carrier.
RESULT_DIC_PHASE	Selects the data to be swept to be the phase of the Digital Input Carrier, w.r.t. the Reference Sync.
RESULT_SA_RMSAMPL_CHA	Selects the data to be swept to be the RMS amplitude of channel A of the Signal Analyzer.
RESULT_SA_RMSAMPL_CHB	Selects the data to be swept to be the RMS amplitude of channel B of the Signal Analyzer.
RESULT_SA_RMSAMPL_SEL	Selects the data to be swept to be the RMS amplitude of the selected channel of the Signal Analyzer.
RESULT_SA_RMSAMPL_NONSEL	Selects the data to be swept to be the RMS amplitude of the non-selected channel of the Signal Analyzer.
RESULT_SA_FREQ_CHA	Selects the data to be swept to be the frequency of channel A of the Signal Analyzer.
RESULT_SA_FREQ_CHB	Selects the data to be swept to be the frequency of channel B of the Signal Analyzer.
RESULT_SA_FREQ_SEL	Selects the data to be swept to be the frequency of the selected channel of the Signal Analyzer.
RESULT_SA_FREQ_NONSEL	Selects the data to be swept to be the frequency of the non-selected channel of the Signal Analyzer.
RESULT_SA_PHASE	Selects the data to be swept to be the inter-channel phase of the Signal Analyzer.
RESULT_CTD_CHA	Selects the data to be swept to be the value of channel A of the Continuous-Time Detector.
RESULT_CTD_CHB	Selects the data to be swept to be the value of channel B of the Continuous-Time Detector.
RESULT_CTD_SEL	Selects the data to be swept to be the value of the selected channel of the Continuous-Time Detector.
RESULT_CTD_NONSEL	Selects the data to be swept to be the value of the non-selected channel of the Continuous-Time Detector.
RESULT_FFTD_CHA	Selects the data to be swept to be the value of channel A of an FFT Detector.
RESULT_FFTD_CHB	Selects the data to be swept to be the value of channel B of an FFT Detector.
RESULT_FFTD_SEL	Selects the data to be swept to be the value of the selected channel of an FFT Detector.
RESULT_FFTD_NONSEL	Selects the data to be swept to be the value of the non-selected channel of an FFT Detector.

NB:

If the Result to be swept is an FFT Detector Result, you must *firstly* select the FFT Detector using the SW_Result1FFTDetector property.

5.7.1.1.5 SW_Result[N]FFTDetector

Description

When the Sweep Result (see SW_Result[N]) is set up to be an FFT Detector Result, this property allows selection of the FFT Detector to use. For example, use SW_Result1FFTDetector to set the FFT Detector for SW_Result1.

Values

This property is a short integer and can be any number between 1 and 40. It must be the ID of an FFT Detector that is currently in use.

NB: **If you are setting up a Sweep of an FFT Detector Result, this property must be set *before* using the SW_Result[N] call to set the Result to sweep.**

5.7.1.1.6 SW_YAxisAutoZoom

Description

This property allows you to specify whether you wish to auto-zoom the Y axis of each Sweep, and if so, whether to perform this auto-zooming after each step of the Sweep, or at the end of the Sweep.

Values

SW_AUTOZOOM_NONE	Do not auto-zoom the Y axis during or after the Sweep.
SW_AUTOZOOM_STEP	Auto-zoom the Y axis of each Sweep after each step of the Sweep.
SW_AUTOZOOM_END	Auto-zoom the Y axis of each Sweep at the end of the Sweep.

5.7.1.1.7 SW_SourceTab

Description

This property allows you to select whether you are specifying details for the inner or outer Sweep source.

If this property is set to **SW_SWEEPTAB_INNER**, then all subsequent Sweep source properties (for example, SW_SweepSource, SW_StartValue, or SW_StopValue) will be set for the inner Sweep source. If this is set to **SW_SWEEPTAB_OUTER**, then all subsequent Sweep source properties will be set for the outer source for nested sweeps.

This property is set to **SW_SWEEPTAB_INNER** by default; if you are not interested in nested sweeps, you will not need to change the value of this property.

NB: **This property has no effect on the currently selected tab on the dScope's user interface; it simply represents which details will get changed from the script.**

Values

SW_SWEEPTAB_INNER	Subsequent Sweep source properties will be set for the inner Sweep source.
SW_SWEEPTAB_OUTER	Subsequent Sweep source properties will be set for the outer Sweep source.

5.7.1.1.8 SW_SweepSource

Description

This property allows selection of the source for the Sweep. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

SW_SOURCE_GENFREQ_CHA	Sets the Sweep source to be the frequency of channel A of the Signal Generator.
SW_SOURCE_GENFREQ_CHB	Sets the Sweep source to be the frequency of channel B of the Signal Generator.
SW_SOURCE_GENFREQ_BOTH	Sets the Sweep source to be the frequency of both channels of the Signal Generator.
SW_SOURCE_GENAMPL_CHA	Sets the Sweep source to be the amplitude of channel A of the Signal Generator.
SW_SOURCE_GENAMPL_CHB	Sets the Sweep source to be the amplitude of channel B of the Signal Generator.
SW_SOURCE_GENAMPL_BOTH	Sets the Sweep source to be the amplitude of both channels of the Signal Generator.
SW_SOURCE_DCOFFSET	Sets the Sweep source to be the Digital Outputs DC offset.
SW_SOURCE_JITTERFREQ	Sets the Sweep source to be the frequency of the jitter on the Digital Output Carrier.
SW_SOURCE_JITTERAMPL	Sets the Sweep source to be the amplitude of the jitter on the Digital Output Carrier.
SW_SOURCE_CTD_BPBRFREQ	Sets the Sweep source to be the frequency of the band pass/ band reject filter on the Continuous-Time Detector.
SW_SOURCE_FFTD_BPBRFREQ	Sets the Sweep source to be the frequency of the band pass/ band reject filter on all FFT Detectors.
SW_SOURCE_SENSEFREQ_CHA	Sets the Sweep source to be sensed frequency changes, on channel A of the Signal Analyzer.
SW_SOURCE_SENSEFREQ_CHB	Sets the Sweep source to be sensed frequency changes, on channel B of the Signal Analyzer.
SW_SOURCE_SENSEAMPL_CHA	Sets the Sweep source to be sensed amplitude changes, on channel A of the Signal Analyzer.
SW_SOURCE_SENSEAMPL_CHB	Sets the Sweep source to be sensed amplitude changes, on channel B of the Signal Analyzer.
SW_SOURCE_TIME	Sets the Sweep source to be time intervals.
SW_SOURCE_DATATABLE	Sets the Sweep source to be a user-defined data table.
SW_SOURCE_MANUAL	Sets the Sweep source to be a manual key press.
SW_SOURCE_CHANNELARRAY	Sets the Sweep source to be a dS-NET Switcher Channel array.

5.7.1.1.9 SW_StartValue

Description

This property allows specification of the start value for the Sweep. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

The value must be specified in the unit selected by SW_Unit.

NB: **The start value of the Sweep can be less than the stop value. This may be useful in some cases, for example frequency Sweeps, which may settle faster going from higher to lower frequencies.**

Values

The Sweep start value is represented as a double-precision floating point value.

NB: **If the generator mode (SG_GenMode) has been set to SG_GENMODE_TIED, then channel B's amplitude will be set to the same as channel A.**

5.7.1.1.10 SW_StopValue

Description

This property allows specification of the stop value for the Sweep. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property. The value must be specified in the unit selected by SW_Unit.

NB: **The start value of the Sweep can be less than the stop value. This may be useful in some cases, for example frequency Sweeps, which may settle faster going from higher to lower frequencies.**

Values

The Sweep stop value is represented as a double-precision floating point value.

5.7.1.1.11 SW_Unit

Description

This property allows selection of the unit for the Sweep source (specified using SW_SweepSource). It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

The allowed values for the Sweep source unit depend on the Sweep source selected using SW_SweepSource.

If the Sweep source is set up to be a frequency Sweep (**SW_SOURCE_GENFREQ_CHA**, **SW_SOURCE_GENFREQ_CHB**, **SW_SOURCE_GENFREQ_BOTH**, **SW_SOURCE_JITTERFREQ**, **SW_SOURCE_CTD_BPBRFREQ** or **SW_SOURCE_FFTD_BPBRFREQ)** then the only allowed unit is **UNIT_FREQ_HZ**. (This unit is selected by default for these Sweep sources and does not need to be set explicitly).

If the Sweep source is set up to be the generated amplitude (**SW_SOURCE_GENAMPL_CHA**, **SW_SOURCE_GENAMPL_CHB** or **SW_SOURCE_GENAMPL_BOTH**) then the following units are allowed:

UNIT_DBFS	Sets the Sweep source unit to dBFS.
UNIT_PERCENTFS	Sets the Sweep source unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Sweep source unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Sweep source unit to Hex.
UNIT_VRMS	Sets the Sweep source unit to an RMS voltage.
UNIT_VP	Sets the Sweep source unit to a peak voltage.
UNIT_VPP	Sets the Sweep source unit to a peak-to-peak voltage.
UNIT_DBU	Sets the Sweep source unit to dBu.
UNIT_DBV	Sets the Sweep source unit to dBV.
UNIT_DBM	Sets the Sweep source unit to dBm.
UNIT_W	Sets the Sweep source unit to W.
UNIT_DBSPL	Sets the Sweep source unit to dBSPL.

If the Sweep source is set to **SW_SOURCE_GENDCOFFSET**, then the following units are allowed:

UNIT_DBFS	Sets the Sweep source unit to dBFS.
UNIT_PERCENTFS	Sets the Sweep source unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Sweep source unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Sweep source unit to Hex.

If the Sweep source is set to **SW_SOURCE_JITTERAMPL**, then the following units are allowed:

UNIT_JITTER_NS	Sets the Sweep source unit to ns.
UNIT_JITTER_UI	Sets the Sweep source unit to UI.

5.7.1.1.12 SW_Interval

Description

This property allows selection of whether to use a linear or logarithmic step through the Sweep source values. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

SW_INTERVAL_LINEAR	Steps through the Sweep source values linearly.
SW_INTERVAL_LOG	Steps through the Sweep source values logarithmically.

NB:

This property is ignored if the Sweep source (SW_SweepSource) does not have user-selectable steps, for example manual, data table or time interval Sweep sources.

5.7.1.1.13 SW_Offset

Description

This property specifies the offset to use when stepping linearly through the Sweep source values from SW_StartValue to SW_StopValue. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property. It must be entered in the unit specified by SW_Unit.

The number of Sweep steps (SW_NumSteps) will be adjusted accordingly if this property is changed.

Values

The Sweep start value is represented as a double-precision floating point value.

NB: **This property is ignored unless the Sweep interval (SW_Interval) is set to SW_INTERVAL_LINEAR.**

5.7.1.1.14 SW_Factor

Description

This property allows selection of the factor to use when stepping logarithmically through the Sweep source values from SW_StartValue to SW_StopValue. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

The Sweep start value is represented as a double-precision floating point value. It must be a value between 0.01 and 100.0.

NB: **This property is ignored unless the Sweep interval (SW_Interval) is set to SW_INTERVAL_LOG.**

5.7.1.1.15 SW_SenseType

Description

This property allows selection of the type of interval to sense when sensing frequency or amplitude as the Sweep source. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

SW_SENSETYPE_OFFSET	Selects that the sense source must change by a certain linear offset before it is treated as a new point.
SW_SENSETYPE_FACTOR	Selects that the sense source must change by a certain factor before it is treated as a new point.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be a sense source (SW_SOURCE_SENSE_...).**

5.7.1.1.16 SW_SenseInterval

Description

This property allows specification of the offset or factor to detect when sensing a Sweep source. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

The Sweep sense interval is entered as a double-precision floating point value.

If the Sweep sense type (SW_SenseType) is set to **SW_SENSETYPE_OFFSET**, it must be entered in the unit specified by SW_Unit.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be a sense source (SW_SOURCE_SENSE_...).**

5.7.1.1.17 SW_SenseUnit

Description

This property allows selection of the unit for entry of the Sweep sense interval (SW_SenseInterval) and end value (SW_SenseEndValue). It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

If the analyzer is currently set up to analyze the demodulated jitter signal through the Analogue Inputs (See AI_Source for further details), then the following values are allowed:

UNIT_JITTER_NS	Sets Sweep sense unit to ns.
UNIT_JITTER_UI	Sets Sweep sense unit to UI.

Under normal analysis, the following values are allowed:

UNIT_DBFS	Sets Sweep sense unit to dBFS.
UNIT_PERCENTFS	Sets Sweep sense unit to %FS (percentage of full scale).
UNIT_FFS	Sets Sweep sense unit to FFS (fraction of full scale).
UNIT_HEX	Sets Sweep sense unit to Hex.
UNIT_V	Sets Sweep sense unit to V.
UNIT_DBU	Sets Sweep sense unit to dBu.
UNIT_DBV	Sets Sweep sense unit to dBV.
UNIT_DBM	Sets Sweep sense unit to dBm.
UNIT_W	Sets Sweep sense unit to W.
UNIT_DBSPL	Sets Sweep sense unit to dBSPL.
UNIT_DBR	Sets Sweep sense unit to dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Sets Sweep sense unit to percentage of the reference amplitude (SA_RefAmpl).

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be a sense source (SW_SOURCE_SENSE_...).**

5.7.1.1.18 SW_SenseEndValue

Description

This property allows specification of the end value for sensed Sweeps. Once this value has been sensed, the Sweep will stop. This property will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

The end value must be specified in the unit selected by SW_SenseUnit.

NB: **The Sweep will actually stop at a detected value within SW_SenseInterval of the value specified.**

Values

The sense end value is represented as a double-precision floating point value.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be a sense source (SW_SOURCE_SENSE_...).**

5.7.1.1.19 SW_SenseThreshold

Description

This property allows specification of the threshold value for sensed Sweeps. While the Signal Analyzer's RMS amplitude is below this value, no points will register as new points for the Sweep. This property will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

The threshold value must be specified in the unit selected by SW_SenseThresholdUnit.

Values

The sense threshold value is represented as a double-precision floating point value.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be a sense source (SW_SOURCE_SENSE_...).**

5.7.1.1.20 SW_SenseThresholdUnit

Description

This property allows selection of the unit for entry of the threshold value for sensed Sweeps (see SW_SenseThreshold). It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

If the analyzer is currently set up to analyze the demodulated jitter signal through the Analogue Inputs (See AI_Source for further details), then the following values are allowed:

UNIT_JITTER_NS	Sets sense threshold unit to ns.
UNIT_JITTER_UI	Sets sense threshold unit to UI.

Under normal analysis, the following values are allowed:

UNIT_DBFS	Sets sense threshold unit to dBFS.
UNIT_PERCENTFS	Sets sense threshold unit to %FS (percentage of full scale).
UNIT_FFS	Sets sense threshold unit to FFS (fraction of full scale).
UNIT_HEX	Sets sense threshold unit to Hex.
UNIT_V	Sets sense threshold unit to V.
UNIT_DBU	Sets sense threshold unit to dBu.
UNIT_DBV	Sets sense threshold unit to dBV.
UNIT_DBM	Sets sense threshold unit to dBm.
UNIT_W	Sets sense threshold unit to W.
UNIT_DBSPL	Sets sense threshold unit to dBSPL.
UNIT_DBR	Sets sense threshold unit to dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Sets sense threshold unit to percentage of the reference amplitude (SA_RefAmpl).

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be a sense source (SW_SOURCE_SENSE_...).**

5.7.1.1.21 SW_TimeInterval

Description

This property allows specification of the time interval, in seconds, for Sweeps whose source is a timer. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

The time interval is entered as a double-precision value. It has a resolution of 0.1 seconds.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be SW_SOURCE_TIME.**

5.7.1.1.22 SW_DataTable

Description

This property allows specification of the data table for a Sweep. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

Any valid file name can be used, enclosed in double quotation marks ("..."). This should be the file name of a Sweep data table (*.tbl) or a dScope script that is used to create a data table (*.dss).

If a full path name is specified, the system will look for this exact file.

If a file name only is specified, then the system will look in the "Sweep data tables" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default).

If necessary, the system will automatically append the correct filename extension (".tbl" for Sweep data tables).

5.7.1.1.23 SW_ChannelArray

Description

This property allows specification of the dS-NET Switcher Channel array to use for a Sweep. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

The name of any dS-NET Switcher Channel Array already set up on the system can be used.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be SW_SOURCE_CHANNELARRAY.**

5.7.1.1.24 SW_StartChannel

Description

This property allows specification of the start channel for Sweeps whose source is a dS-NET Switcher Channel array. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

The start channel is entered as a short integer value.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be SW_SOURCE_CHANNELARRAY.**

5.7.1.1.25 SW_EndChannel

Description

This property allows specification of the end channel for Sweeps whose source is a dS-NET Switcher Channel array. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

The end channel is entered as a short integer value.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be SW_SOURCE_CHANNELARRAY.**

5.7.1.1.26 SW_ChannelArrayMode

Description

This property allows specification of the mode of operation for Sweeps whose source is a dS-NET Switcher Channel array. It determines whether channels will be turned on or off at each sweep step. This property will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

SW_CHANNELARRAYMODE_ON	Each channel is turned on, with all other channels turned off.
SW_CHANNELARRAYMODE_OFF	Each channel is turned off, with all other channels turned on.

NB: **This property is ignored unless the Sweep source (SW_SweepSource) is set to be SW_SOURCE_CHANNELARRAY.**

5.7.1.1.27 SW_RunScript

Description

This property is used to select whether to run a script at a specified point in each Sweep step. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

Values

True	Run a script during each step of the Sweep.
False	Do not run a script during each step of the Sweep (default).

5.7.1.1.28 SW_Script

Description

This property allows specification of the script to run during each step of a Sweep. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property, and will be ignored unless the value of the SW_RunScript property is set to **True**.

Values

Any valid file name can be used, enclosed in double quotation marks ("..."). This should be the file name of an automation script (*.dss).

If a full path name is specified, the system will look for this exact file.

If a full path name is not specified, then the system will look in the "Automation" subfolder of the folder specified in the Options dialogue box for scripts (See OPT_ScriptsFolder).

If necessary, the system will automatically append the correct filename extension (".dss" for Automation scripts).

5.7.1.1.29 SW_RunScriptWhen

Description

This property allows specification of when to run a script during each Sweep step. This property will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property, and will be ignored unless the value of the SW_RunScript property is set to **True**.

Values

SW_RUNSCRIPTWHEN_ATSTART	The specified script is run at the start of the Sweep step.
SW_RUNSCRIPTWHEN_AFTERSOURCE	The specified script is run after the Sweep source details have been set.
SW_RUNSCRIPTWHEN_AFTERREGULATION	The specified script is run after Regulation has been performed, if applicable.
SW_RUNSCRIPTWHEN_ATEND	The specified script is run at the end of the Sweep step.

5.7.1.1.30 SW_CurrentStep

Description

This **read-only** property returns the index of the current Sweep step. It can be used from a script running at each Sweep step (See SW_Script and SW_RunScript for further information).

Values

The current Sweep step is represented as a short integer value. It can be any number from 0 to the number of steps (SW_NumSteps).

5.7.1.1.31 SW_NumSteps

Description

This property allows specification of the number of steps to perform in the Sweep. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property.

The offset or factor (SW_Offset or SW_Factor) will be adjusted accordingly if this property is changed.

Values

The number of steps is entered as a short integer value.

NB: **This property is ignored unless the Sweep source ([SW_SweepSource]) is set to be one with a user-definable number of steps.**

The number of points in the Sweep will be one more than the number of steps, as it includes the start point.

5.7.1.1.32 SW_Regulate

Description

This property is used to select whether to perform Regulation at each step of the Sweep. It will be set for the inner or outer Sweep source, dependent on the value of the SW_SourceTab property. If this option is selected, Regulation is performed after setting the Sweep source for the current step, but before reading the Results. The currently selected Regulation parameters are used for Regulation.

NB: **Care should be taken to ensure that the Regulation Source is not the same as the Source to be used for the Sweep, and that the Result to be regulated is not in use on the Sweep.**

Values

True	Perform Regulation during each step of the Sweep.
False	Do not perform Regulation during the Sweep.

5.7.1.1.33 SW_XAxisAutoZoom

Description

This property allows you to specify whether you wish to auto-zoom the X axis of each Sweep, and if so, whether to perform this auto-zooming after each step of the Sweep, or at the end of the Sweep.

Values

SW_AUTOZOOM_NONE	Do not auto-zoom the X axis during or after the Sweep.
SW_AUTOZOOM_STEP	Auto-zoom the X axis of each Sweep after each step of the Sweep.
SW_AUTOZOOM_END	Auto-zoom the X axis of each Sweep at the end of the Sweep.

5.7.1.2 Methods

5.7.1.2.1 SW_Go

SW_Go ()

This method starts the Sweep according to the currently selected Sweep source details.

NB: **After starting the Sweep, control will be returned *immediately* to the script, before the Sweep has finished. To wait until the Sweep has finished, use the SW_IsSweepFinished method.**

Parameters

This method has no parameters.

Return value

This method has no return value.

5.7.1.2.2 SW_Stop

SW_Stop ()

This method stops the currently running Sweep.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.7.1.2.3 SW_Pause

SW_Pause ()

This method pauses the currently running Sweep. It can then be restarted again using SW_Go or SW_SingleStep.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.7.1.2.4 SW_SingleStep

SW_SingleStep ()

This method performs a single step of the current Sweep. No action will be performed unless the current Sweep is paused, or has been started using this method.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.7.1.2.5 SW_IsSweepFinished

bFinished = SW_IsSweepFinished ()

This method can be called to determine whether the currently running Sweep has finished executing.

For example:

```
While Not Sweep.SW_IsSweepFinished()
    ' Wait a bit longer...
    Sleep(0)
Wend
```

NB: **There is a reason that this method is implemented the way it is, rather than just as a "WaitUntilFinished" function. This is because if a "Wait" method were called from an external program, it would go into a loop and stop the dScope running, so the function would never return.**

Parameters

This method has no parameters.

Return value

This method returns **True** if the Sweep has finished, or **False** if it is currently running.

5.7.1.2.6 SW_MinLimitBreached

bBreached = SW_MinLimitBreached (sTraceType, sChannel)

This method can be called to determine whether the last Sweep run breached its minimum limit.

NB: **It is also possible to detect limit breaches using Events.**

Parameters

sTraceType	The Trace type of the Sweep to check on. It can have any of the values listed under Trace Types below.
sChannel	The channel that the Sweep was done on. It can be one of the values listed under Channels, below.

Return value

This method returns **True** if the Sweep's minimum limit was breached, or **False** if either the Sweep had no minimum limit applied, or the limit was not breached.

Trace Types

TRACETYPE_SWEEP1	Checks the minimum limit of a Sweep done for the first Result specified (using **SW_Result1**; See SW_Result[N] for details).
TRACETYPE_SWEEP2	Checks the minimum limit of a Sweep done for the second Result specified (using **SW_Result2**; See SW_Result[N] for details).
TRACETYPE_SWEEP3	Checks the minimum limit of a Sweep done for the third Result specified (using **SW_Result3**; See SW_Result[N] for details).
TRACETYPE_SWEEP4	Checks the minimum limit of a Sweep done for the fourth Result specified (using **SW_Result4**; See SW_Result[N] for details).

Channels

CHANNEL_A	Checks the minimum limit of a Sweep done for a channel A Result (or a Result where the channel is not specified).
CHANNEL_B	Checks the minimum limit of a Sweep done for a channel B Result.

5.7.1.2.7 SW_MaxLimitBreached

bBreached = SW_MaxLimitBreached (sTraceType, sChannel)

This method can be called to determine whether the last Sweep run breached its maximum limit.

NB: **It is also possible to detect limit breaches using Events.**

Parameters

sTraceType	The Trace type of the Sweep to check on. It can have any of the values listed under Trace Types below.
sChannel	The channel that the Sweep was done on. It can be one of the values listed under Channels, below.

Return value

This method returns **True** if the Sweep's maximum limit was breached, or **False** if either the Sweep had no maximum limit applied, or the limit was not breached.

Trace Types

TRACETYPE_SWEEP1	Checks the maximum limit of a Sweep done for the first Result specified (using **SW_Result1**; See SW_Result[N] for details).
TRACETYPE_SWEEP2	Checks the maximum limit of a Sweep done for the second Result specified (using **SW_Result2**; See SW_Result[N] for details).
TRACETYPE_SWEEP3	Checks the maximum limit of a Sweep done for the third Result specified (using **SW_Result3**; See SW_Result[N] for details).
TRACETYPE_SWEEP4	Checks the maximum limit of a Sweep done for the fourth Result specified (using **SW_Result4**; See SW_Result[N] for details).

Channels

CHANNEL_A	Checks the maximum limit of a Sweep done for a channel A Result (or a Result where the channel is not specified).
CHANNEL_B	Checks the maximum limit of a Sweep done for a channel B Result.

5.7.1.2.8 SW_SetYUnit

bRet = SW_SetYUnit (sResult, sUnit)

This method sets the Y unit for one of the Sweep Results.

Parameters

sResult	The Sweep Result to set the Y unit for. This must be a number between 1 and 4.
sUnit	The Y unit to set for this Sweep Result. It can be one of the values listed under Units, below.

Return value

This method returns **True** if the function is successful, or **False** if it fails.

Units

The unit must be a valid unit for the Result selected. If the Result is from the Continuous-Time Detector or FFT Detector, then the unit must also be valid for the current relativity and response of the Detector. The following list details all the units available:

UNIT_FREQ_HZ	Sets the Sweep's Y unit to Hz.
UNIT_DBFS	Sets the Sweep's Y unit to dBFS.
UNIT_PERCENTFS	Sets the Sweep's Y unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Sweep's Y unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Sweep's Y unit to Hex.
UNIT_V	Sets the Sweep's Y unit to Volts.
UNIT_VRMS	Sets the Sweep's Y unit to Volts, RMS.
UNIT_VP	Sets the Sweep's Y unit to Volts, peak.
UNIT_VPP	Sets the Sweep's Y unit to Volts, peak-to-peak.
UNIT_DBU	Sets the Sweep's Y unit to dBu.
UNIT_DBV	Sets the Sweep's Y unit to dBV.
UNIT_DBM	Sets the Sweep's Y unit to dBm.
UNIT_W	Sets the Sweep's Y unit to W.
UNIT_DBSPL	Sets the Sweep's Y unit to dBSPL.
UNIT_DBR	Sets the Sweep's Y unit to dBr (dB, relative to the reference amplitude specified using SA_RefAmpl).
UNIT_PERCENTREF	Sets the Sweep's Y unit to a percentage of the reference amplitude, specified using SA_RefAmpl.
UNIT_RELATIVE_DB	Sets the Sweep's Y unit to dB.
UNIT_RELATIVE_PERCENT	Sets the Sweep's Y unit to percent.
UNIT_JITTER_UI	Sets the Sweep's Y unit to UI.
UNIT_JITTER_NS	Sets the Sweep's Y unit to ns.
UNIT_PHASE_SAMPLES	Sets the Sweep's Y unit to be samples.

UNIT_PHASE_DEGREES	Sets the Sweep's Y unit to be degrees.
UNIT_PHASE_RADIANS	Sets the Sweep's Y unit to be radians.
UNIT_PHASE_US	Sets the Sweep's Y unit to be microseconds.
UNIT_PPM	Sets the Sweep's Y unit to be ppm (parts per million).

5.7.1.2.9 SW_SetYRange

bRet = SW_SetYRange (sResult, dMinValue, dMaxValue)

This method sets the Y range for one of the Sweep Results. The minimum and maximum values must be entered in the unit specified using the SW_SetYUnit method.

Parameters

sResult	The Sweep Result to set the Y range for. This must be a number between 1 and 4.
dMinValue	The minimum value for the Sweep's Y axis.
dMaxValue	The maximum value for the Sweep's Y axis.

Return value

This method returns **True** if the function is successful, or **False** if it fails.

5.7.1.2.10 SW_SetYIntervals

bRet = SW_SetYIntervals (sResult, sNumIntervals, bLog)

This method sets the intervals for the Y axis for one of the Sweep Results.

Parameters

sResult	The Sweep Result to set the Y axis intervals for. This must be a number between 1 and 4.
sNumIntervals	The number of intervals to display on the Sweep's Y axis. Use **TRACE_INTERVALS_AUTO** to specify that the dScope should automatically calculate the intervals.
bLog	**True** to display the axis logarithmically, **False** to display it linearly.

NB: **If the scale is set to logarithmic, and the Y scale starts at 0, the minimum Y value will be adjusted when displayed to allow it to be shown logarithmically.**

Return value

This method returns **True** if the function is successful, or **False** if it fails. This may be because the number of intervals is invalid, or because the selected Y unit does not allow a logarithmic scale.

5.7.1.2.11 SW_SetMaxLimit

bRet = SW_SetMaxLimit (sResult, strLimitFile)

This method sets the Upper Limit Line for the selected Sweep Result.

Parameters

sResult	The Sweep Result to set the limit file for. This must be a number between 1 and 4.
strLimitFile	The file name of a Limit Line to use as this Sweep's upper limit. Any valid file name can be used, enclosed in double quotation marks ("..."). The file name passed can be the name of a limit file (*.lmt), or the name of a script file used to create a Limit Table (*.dss). If a full path name is not specified, then the system will look in the folder specified in the Options dialogue box for Limit Files (See OPT_LimitFilesFolder). If necessary, the system will automatically append the file extension for Limit Table files (".lmt").

Return value

This method returns **True** if the limit was set successfully, or **False** if it failed. This may be because the file passed was invalid, or the limit file's units are incompatible with the Sweep's units.

5.7.1.2.12 SW_SetMinLimit

bRet = SW_SetMinLimit (sResult, strLimitFile)

This method sets the Lower Limit Line for the selected Sweep Result.

Parameters

sResult	The Sweep Result to set the limit file for. This must be a number between 1 and 4.
strLimitFile	The file name of a Limit Line to use as this Sweep's lower limit. Any valid file name can be used, enclosed in double quotation marks ("..."). The file name passed can be the name of a limit file (*.lmt), or the name of a script file used to create a Limit Table (*.dss). If a full path name is not specified, then the system will look in the folder specified in the Options dialogue box for Limit Files (See OPT_LimitFilesFolder). If necessary, the system will automatically append the file extension for Limit Table files (".lmt").

Return value

This method returns **True** if the limit was set successfully, or **False** if it failed. This may be because the file passed was invalid, or the limit file's units are incompatible with the Sweep's units.

5.7.1.2.13 SW_ResetYDefaults

bRet = SW_ResetYDefaults (sResult)

This method resets the Y settings to their defaults for one of the Sweep Results.

Parameters

sResult	The Sweep Result to reset the Y settings for. This must be a number between 1 and 4.

Return value

This method returns **True** if the function is successful, or **False** if it fails.

5.7.1.2.14 SW_SetXAxisResult

bRet = SW_SetXAxisResult (sResult)

This method allows selection of one of the dScope's input parameters (Results) to be plotted on the X axis of a Sweep, rather than the Sweep Source details.

Parameters

sResult	The Result to be plotted on the X axis. It can be one of the values listed under Values, below.

Return value

This method returns **True** if the method is successful, or **False** if it fails.

Values

RESULT_NONE	Resets the X axis Result, so that the *Source* details will be plotted on the X axis.
RESULT_DO_REFSYNCFRAMERATE	Specifies that the frame rate of the Ref Sync source should be plotted on the Sweep's X axis.
RESULT_DO_REFSYNCFRAMERATEDEVIATION	Specifies that the deviation of the Ref Sync source's frame rate from the nearest standard rate should be plotted on the Sweep's X axis.
RESULT_DI_FRAMERATE	Specifies that the frame rate of the Digital Input should be plotted on the Sweep's X axis.

RESULT_DI_FRAMERATEDEVIATION	Specifies that the deviation of the Digital Input frame rate from the nearest standard rate should be plotted on the Sweep's X axis.
RESULT_DIC_AMPL	Specifies that the amplitude of the Digital Input Carrier should be plotted on the Sweep's X axis.
RESULT_DIC_JITTERAMPL	Specifies that the amplitude of the jitter on the Digital Input Carrier should be plotted on the Sweep's X axis.
RESULT_DIC_PHASE	Specifies that the phase of the Digital Input Carrier, w.r.t. the Reference Sync should be plotted on the Sweep's X axis.
RESULT_SA_RMSAMPL_SEL	Specifies that the RMS amplitude of the Signal Analyzer (on the same channel as the Sweep Result) should be plotted on the Sweep's X axis.
RESULT_SA_FREQ_SEL	Specifies that the frequency of the Signal Analyzer (on the same channel as the Sweep Result) should be plotted on the Sweep's X axis.
RESULT_SA_PHASE	Specifies that the inter-channel phase of the Signal Analyzer should be plotted on the Sweep's X axis.
RESULT_CTD_SEL	Specifies that the value of the Continuous-Time Detector (on the same channel as the Sweep Result) should be plotted on the Sweep's X axis.
RESULT_FFTD_SEL	Specifies that the value of the FFT Detector (on the same channel as the Sweep Result) should be plotted on the Sweep's X axis.

NB: **If the Result to be used on the X axis is an FFT Detector Result (RESULT_FFTD_SEL), you must *firstly* select the FFT Detector using the SW_SetXAxisFFTDetector method.**

5.7.1.2.15 SW_SetXAxisFFTDetector

bRet = SW_SetXAxisFFTDetector (sDetectorID)

When the Result to be plotted on the Sweep's X axis is an FFT Detector Result (i.e. SW_SetXAxisResult has been called with a parameter of **RESULT_FFTD_SEL**), this property allows selection of the FFT Detector to use.

NB: **If an FFT Detector is required, this property must be set *before* using the SW_SetXAxisResult method.**

Parameters

sDetectorID	The ID of the FFT Detector whose Result values are to be plotted on the X axis. It can be any number between 1 and 40 and must be the ID of an FFT Detector that is currently in use.

Return value

This method returns **True** if the method is successful, or **False** if it fails.

5.7.1.2.16 SW_SetXUnit

bRet = SW_SetXUnit (sUnit)

This method sets the unit for the X axis of the sweep.

NB: A Result to plot on the X axis must have been selected using SW_SetXAxisResult before this method can be used.

Parameters

sUnit	The X unit to set for the Result to be plotted on the X axis. It can be one of the values listed under Units, below.

Return value

This method returns **True** if the function is successful, or **False** if it fails.

Units

The unit must be a valid unit for the Result selected using SW_SetXAxisResult. If the Result is from the Continuous-Time Detector or FFT Detector, then the unit must also be valid for the current relativity and response of the Detector. The following list details all the units available:

UNIT_FREQ_HZ	Sets the Sweep's X unit to Hz.
UNIT_DBFS	Sets the Sweep's X unit to dBFS.
UNIT_PERCENTFS	Sets the Sweep's X unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Sweep's X unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Sweep's X unit to Hex.
UNIT_V	Sets the Sweep's X unit to Volts.
UNIT_VRMS	Sets the Sweep's X unit to Volts, RMS.
UNIT_VP	Sets the Sweep's X unit to Volts, peak.
UNIT_VPP	Sets the Sweep's X unit to Volts, peak-to-peak.
UNIT_DBU	Sets the Sweep's X unit to dBu.
UNIT_DBV	Sets the Sweep's X unit to dBV.
UNIT_DBM	Sets the Sweep's X unit to dBm.
UNIT_W	Sets the Sweep's X unit to W.
UNIT_DBSPL	Sets the Sweep's X unit to dBSPL.
UNIT_DBR	Sets the Sweep's X unit to dBr (dB, relative to the reference amplitude specified using SA_RefAmpl).
UNIT_PERCENTREF	Sets the Sweep's X unit to a percentage of the reference amplitude, specified using SA_RefAmpl.
UNIT_RELATIVE_DB	Sets the Sweep's X unit to dB.
UNIT_RELATIVE_PERCENT	Sets the Sweep's X unit to percent.
UNIT_JITTER_UI	Sets the Sweep's X unit to UI.
UNIT_JITTER_NS	Sets the Sweep's X unit to ns.
UNIT_PHASE_SAMPLES	Sets the Sweep's X unit to be samples.
UNIT_PHASE_DEGREES	Sets the Sweep's X unit to be degrees.
UNIT_PHASE_RADIANS	Sets the Sweep's X unit to be radians.
UNIT_PHASE_US	Sets the Sweep's X unit to be microseconds.

UNIT_PPM	Sets the Sweep's X unit to be ppm (parts per million).

5.7.1.2.17 SW_SetXRange

bRet = SW_SetXRange (dMinValue, dMaxValue)

This method sets the range for the X axis of the Sweep. The minimum and maximum values must be entered in the unit specified using the SW_SetXUnit method.

NB: A Result to plot on the X axis must have been selected using SW_SetXAxisResult before this method can be used.

Parameters

dMinValue	The minimum value for the Sweep's X axis.
dMaxValue	The maximum value for the Sweep's X axis.

Return value

This method returns **True** if the function is successful, or **False** if it fails.

5.7.1.2.18 SW_SetXIntervals

bRet = SW_SetXIntervals (sNumIntervals, bLog)

This method sets the intervals for the X axis of the Sweep.

NB: A Result to plot on the X axis must have been selected using SW_SetXAxisResult before this method can be used.

Parameters

sNumIntervals	The number of intervals to display on the Sweep's X axis. Use **TRACE_INTERVALS_AUTO** to specify that the dScope should automatically calculate the intervals.
bLog	**True** to display the axis logarithmically, **False** to display it linearly.

NB: If the scale is set to logarithmic, and the X scale starts at 0, the minimum X value will be adjusted when displayed to allow it to be shown logarithmically.

Return value

This method returns **True** if the function is successful, or **False** if it fails. This may be because the number of intervals is invalid, or because the selected X unit does not allow a logarithmic scale.

5.7.2 Settling Parameters

The Sweep Settling section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"Settling."**

Note that there are seven different sets of settling details, and a set of settling details will cover all Results from that area of the application. The Results that are covered by each set of details are as follows:

1) Signal Analyzer RMS amplitude Results:
RESULT_SA_RMSAMPL_CHA, **RESULT_SA_RMSAMPL_CHB**, **RESULT_SA_RMSAMPL_SEL**, **RESULT_SA_RMSAMPL_NONSEL**

2) Signal Analyzer frequency Results:
RESULT_SA_FREQ_CHA, **RESULT_SA_FREQ_CHB**, **RESULT_SA_FREQ_SEL**, **RESULT_SA_FREQ_NONSEL**

3) Signal Analyzer phase Result:
RESULT_SA_PHASE

4) Continuous-Time Detector Results:
RESULT_CTD_CHA, **RESULT_SA_FREQ_CHB**, **RESULT_SA_FREQ_SEL**, **RESULT_SA_FREQ_NONSEL**

2) FFT Detector Results:
RESULT_FFTD_CHA, **RESULT_FFTD_CHB**, **RESULT_FFTD_SEL**, **RESULT_FFTD_NONSEL**

3) Digital Input Carrier amplitude Result:
RESULT_DIC_AMPL

4) Digital Input Carrier jitter Result:
RESULT_DIC_JITTERAMPL

5) Digital Input Carrier phase Result:
RESULT_DIC_PHASE

6) Digital Input frame rate Results:
RESULT_DI_FRAMERATE, **RESULT_DI_FRAMERATEDEVIATION**

7) Reference Sync source Result:
RESULT_DO_REFSYNCSOURCE, **RESULT_DO_REFSYNCSOURCEDEVIATION**

Properties

SETT_SAAmplConvergence
SETT_SAAmplTolerance
SETT_SAAmplSettlingTime
SETT_SAAmplNumResults
SETT_SAAmplAverage
SETT_SAFreqConvergence
SETT_SAFreqTolerance
SETT_SAFreqSettlingTime
SETT_SAFreqNumResults
SETT_SAFreqAverage
SETT_SAPhaseConvergence
SETT_SAPhaseTolerance
SETT_SAPhaseSettlingTime
SETT_SAPhaseNumResults

Methods

There are no methods available to control the Sweep Settling

5.7.2.1 Properties

5.7.2.1.1 SETT_SAAmplConvergence

Description

This property allows selection of the convergence to use when detecting whether Signal Analyzer RMS amplitude Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.2 SETT_SAAmplTolerance

Description

This property allows specification of the tolerance to use when detecting whether Signal Analyzer RMS amplitude Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.3 SETT_SAAmplSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether Signal Analyzer RMS amplitude Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.4 SETT_SAAmplNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether Signal Analyzer RMS amplitude Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.5 SETT_SAAmplAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether Signal Analyzer RMS amplitude Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.6 SETT_SAFreqConvergence

Description

This property allows selection of the convergence to use when detecting whether Signal Analyzer frequency Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.7 SETT_SAFreqTolerance

Description

This property allows specification of the tolerance to use when detecting whether Signal Analyzer frequency Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.8 SETT_SAFreqSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether Signal Analyzer frequency Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.9 SETT_SAFreqNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether Signal Analyzer frequency Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.10 SETT_SAFreqAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether Signal Analyzer frequency Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.11 SETT_SAPhaseConvergence

Description

This property allows selection of the convergence to use when detecting whether the Signal Analyzer inter-channel phase has settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.12 SETT_SAPhaseTolerance

Description

This property allows specification of the tolerance to use when detecting whether the Signal Analyzer inter-channel phase has settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.13 SETT_SAPhaseSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether the Signal Analyzer inter-channel phase has settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.14 SETT_SAPhaseNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether the Signal Analyzer inter-channel phase has settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.15 SETT_SAPhaseAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether the Signal Analyzer inter-channel phase has settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.16 SETT_CTDConvergence

Description

This property allows selection of the convergence to use when detecting whether Continuous-Time Detector Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.17 SETT_CTDTolerance

Description

This property allows specification of the tolerance to use when detecting whether Continuous-Time Detector Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.18 SETT_CTDSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether Continuous-Time Detector Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.19 SETT_CTDNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether Continuous-Time Detector Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.20 SETT_CTDAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether Continuous-Time Detector Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.21 SETT_FFTDConvergence

Description

This property allows selection of the convergence to use when detecting whether FFT Detector Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.22 SETT_FFTDTolerance

Description

This property allows specification of the tolerance to use when detecting whether FFT Detector Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.23 SETT_FFTDSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether FFT Detector Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.24 SETT_FFTDNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether FFT Detector Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.25 SETT_FFTDAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether FFT Detector Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.26 SETT_DICAmplConvergence

Description

This property allows selection of the convergence to use when detecting whether Digital Input Carrier amplitude Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.27 SETT_DICAmplTolerance

Description

This property allows specification of the tolerance to use when detecting whether Digital Input Carrier amplitude Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.28 SETT_DICAmplSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether Digital Input Carrier amplitude Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.29 SETT_DICAmplNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether Digital Input Carrier amplitude Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.30 SETT_DICAmplAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether Digital Input Carrier amplitude Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.31 SETT_DICJitterConvergence

Description

This property allows selection of the convergence to use when detecting whether Digital Input Carrier jitter Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.32 SETT_DICJitterTolerance

Description

This property allows specification of the tolerance to use when detecting whether Digital Input Carrier jitter Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.33 SETT_DICJitterSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether Digital Input Carrier jitter Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.34 SETT_DICJitterNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether Digital Input Carrier jitter Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.35 SETT_DICJitterAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether Digital Input Carrier jitter Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.36 SETT_DICPhaseConvergence

Description

This property allows selection of the convergence to use when detecting whether Digital Input Carrier phase Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the result before that, etc.

5.7.2.1.37 SETT_DICPhaseTolerance

Description

This property allows specification of the tolerance to use when detecting whether Digital Input Carrier phase Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.38 SETT_DICPhaseSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether Digital Input Carrier phase Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.39 SETT_DICPhaseNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether Digital Input Carrier phase Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.40 SETT_DICPhaseAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether Digital Input Carrier phase Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.41 SETT_DIFrameRateConvergence

Description

This property allows selection of the convergence to use when detecting whether Digital Input frame rate Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.42 SETT_DIFrameRateTolerance

Description

This property allows specification of the tolerance to use when detecting whether Digital Input frame rate Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.43 SETT_DIFrameRateSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether Digital Input frame rate Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.44 SETT_DIFrameRateNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether Digital Input frame rate Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.45 SETT_DIFrameRateAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether Digital Input frame rate Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.2.1.46 SETT_RefSyncSourceConvergence

Description

This property allows selection of the convergence to use when detecting whether Reference Sync Results have settled.

Values

SW_CONVERGENCE_NONE	Selects that no convergence should be used. Results are not checked to see if they are within the given tolerance of the previous Result.
SW_CONVERGENCE_NORMAL	Selects that normal convergence should be used. This indicates that each Result should be within the given tolerance of the last Result.
SW_CONVERGENCE_EXPONENTIAL	Selects that exponential convergence should be used. This indicates that each Result should be within the given tolerance of the last Result, which should be within twice the tolerance of the Result before that, which should be within four times the tolerance of the Result before that, etc.

5.7.2.1.47 SETT_RefSyncSourceTolerance

Description

This property allows specification of the tolerance to use when detecting whether Reference Sync source Results have settled.

Values

The tolerance value is entered in percent, as a double-precision floating point value.

5.7.2.1.48 SETT_RefSyncSourceSettlingTime

Description

This property allows specification of the settling time to wait before reading Results when detecting whether Reference Sync source Results have settled.

Values

The settling time is entered in milliseconds between 0 and 5000, as a short integer value.

5.7.2.1.49 SETT_RefSyncSourceNumResults

Description

This property allows specification of the number of Results to read that meet the tolerance requirements, when detecting whether Reference Sync source Results have settled.

Values

The number of Results is entered as a short integer value, between 1 and 10.

5.7.2.1.50 SETT_RefSyncSourceAverage

Description

This property allows specification of whether to average the number of Results read, when detecting whether Reference Sync source Results have settled.

Values

True	Average Results and use the average as the Sweep Result for this step.
False	Do not average Results; use the last value read as the Sweep Result for this step.

5.7.3 Regulation

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The Regulation section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"Regulation."**

Properties

REG_Result
REG_ResultFFTDetector
REG_Channel

Methods

5.7.3.1 Properties

5.7.3.1.1 REG_Result

Description

This property allows selection of the Result to be regulated. If relevant, the channel(s) to regulate the Result for can be selected using REG_Channel.

Values

REG_RESULT_RMSAMPLITUDE	Selects the Signal Analyzer RMS amplitude to be regulated.
REG_RESULT_CTDETECTOR	Selects the value of the Continuous-Time Detector to be regulated.
REG_RESULT_FFTDETECTOR	Selects the value of an FFT Detector to be regulated. NB: The FFT Detector to be used can be selected using the REG_ResultFFTDetector property.

NB: **If the Result to be swept is REG_RESULT_FFTDETECTOR, you must *firstly* select the FFT Detector using the REG_ResultFFTDetector property.**

5.7.3.1.2 REG_ResultFFTDetector

Description

When the Result to be regulated (see REG_Result) is set up to be an FFT Detector Result (**REG_RESULT_FFTDETECTOR**), this property allows selection of the FFT Detector to use.

Values

This property is a short integer and can be any number between 1 and 40. It must be the ID of an FFT Detector that is currently in use.

NB: **If you are setting up regulation of an FFT Detector Result, this property must be set *before* using the REG_Result call to set the Result to regulate.**

5.7.3.1.3 REG_Channel

Description

For Results to be regulated that can be on more than one channel, this property allows selection of the channel. For example, if the Result to be regulated (REG_Result) is **REG_RESULT_CTDETECTOR**, this property allows you to select which channel of the Continuous-Time Detector to regulate.

NB: **This channel also affects which channel(s) of the source will be varied; see REG_Source and REG_SourceOppChannel for details.**

Values

REG_CHANNEL_A	Regulates channel A of the selected Result.
REG_CHANNEL_B	Regulates channel B of the selected Result.
REG_CHANNEL_SEL	Regulates the channel of the selected Result that matches the currently selected channel of the Signal Analyzer. If "Both" channels are selected in the Signal Analyzer, then channel A will be regulated.
REG_CHANNEL_NONSEL	Regulates the channel of the selected Result that matches the channel of the Signal Analyzer that is *not* currently selected. If "Both" channels are selected in the Signal Analyzer, then channel B will be regulated.
REG_CHANNEL_BOTH	Regulates the selected Result until both channel A *and* channel B meet the end criteria. Note that this will result in two passes of the regulation occurring.
REG_CHANNEL_BOTHTIED	Regulates the selected Result until both channel A *and* channel B meet the end criteria, but with the source for both channels tied together. This method is quicker than **REG_CHANNEL_BOTH**, providing that both channels have sufficiently similar characteristics to regulate with the same source value.
REG_CHANNEL_EITHER	Regulates the selected Result until *either* channel A *or* channel B meet the end criteria.

NB: **REG_CHANNEL_BOTHTIED and REG_CHANNEL_EITHER are not available unless the regulation type (REG_RegulationType) is set to "absolute" (REG_REGULATIONTYPE_ABSOLUTE).**

5.7.3.1.4 REG_RegulationType

Description

This property allows selection of the type of regulation to perform.

Values

REG_REGULATIONTYPE_ABSOLUTE	Regulates the selected Result until it reaches the value specified by REG_RegulateTo (within the specified tolerance).
REG_REGULATIONTYPE_MAXIMUM	Regulates the selected Result until its maximum value is found.
REG_REGULATIONTYPE_MINIMUM	Regulates the selected Result until its minimum value is found.

5.7.3.1.5 REG_RegulateTo

Description

This property represents the absolute value to regulate to. It must be specified in the unit selected by REG_ResultUnit.

Values

The value to regulate to is represented as a double-precision floating point value.

NB: **This property is ignored unless the regulation type (REG_RegulationType) is set to be "absolute" (REG_REGULATIONTYPE_ABSOLUTE).**

5.7.3.1.6 REG_ResultUnit

Description

This property allows selection of the unit for the value to regulate to (REG_RegulateTo).

Values

The allowed values for the Result unit depend on the currently selected Result.

If the Result is a relative amplitude (i.e. for the Continuous-Time Detector or an FFT Detector), then the following values are valid:

UNIT_RELATIVE_DB	Sets Result unit to dB.
UNIT_RELATIVE_PERCENT	Sets Result unit to %.

If the Result is an absolute amplitude, and the analyzer is currently set up to analyze the demodulated jitter signal through the Analogue Inputs (See AI_Source for further details), then the following values are allowed (this applies to the RMS Amplitude Result, and the Detector Results when the Detector's relativity is set to "absolute"):

UNIT_JITTER_NS	Sets Result unit to ns.
UNIT_JITTER_UI	Sets Result unit to UI.

If the Result is an absolute amplitude, then the following values are valid:

UNIT_DBFS	Sets Result unit to dBFS.
UNIT_PERCENTFS	Sets Result unit to %FS (percentage of full scale).
UNIT_FFS	Sets Result unit to FFS (fraction of full scale).
UNIT_HEX	Sets Result unit to Hex.

UNIT_V	Sets Result unit to V.
UNIT_DBU	Sets Result unit to dBu.
UNIT_DBV	Sets Result unit to dBV.
UNIT_DBM	Sets Result unit to dBm.
UNIT_W	Sets Result unit to W.
UNIT_DBSPL	Sets Result unit to dBSPL.
UNIT_DBR	Sets Result unit to dBr (dB with respect to the reference amplitude, SA_RefAmpl).
UNIT_PERCENTREF	Sets Result unit to percentage of the reference amplitude (SA_RefAmpl).

If the Result is the Signal Analyzer frequency, then the following units are valid:

UNIT_FREQ_HZ	Sets Result unit to Hz.
UNIT_FREQ_OFFSET	Sets Result unit to offset from the reference frequency (SA_RefFreq), in Hz.
UNIT_FREQ_RATIO	Sets Result unit to a ratio of the reference frequency (SA_RefFreq).

If the Result is the Signal Analyzer phase, then the following units are valid:

UNIT_PHASE_DEGREES	Sets Result unit to degrees.
UNIT_PHASE_RADIANS	Sets Result unit to radians.
UNIT_PHASE_US	Sets Result unit to microseconds.
UNIT_PHASE_SAMPLES	Sets Result unit to samples.

NB: **This property is ignored unless the regulation type (REG_RegulationType) is set to be "absolute" (REG_REGULATIONTYPE_ABSOLUTE).**

5.7.3.1.7 REG_ToleranceType

Description

This property allows selection of whether the tolerance (REG_Tolerance) of the value to regulate to is specified as an offset in the specified Result unit (REG_ResultUnit), or as a ratio in dB or % (as specified by REG_ToleranceUnit).

Values

REG_TOLERANCETYPE_OFFSET	Specifies that the tolerance (REG_Tolerance) is an offset from the value to regulate to.
REG_TOLERANCETYPE_RATIO	Specifies that the tolerance (REG_Tolerance) is a ratio of the value to regulate to.

5.7.3.1.8 REG_Tolerance

Description

This property represents the tolerance within which the regulation algorithm attempts to find its end value.

If the regulation type (REG_RegulationType) is "absolute", then the tolerance specifies how close the selected Result must be to the requested value (REG_RegulateTo) in order for regulation to succeed.

If the regulation type is "minimum" or "maximum", then the tolerance specifies how close together three results must be when a peak or trough is found, in order to stop regulating.

Values

The tolerance is represented as a double-precision floating point value.

5.7.3.1.9 REG_ToleranceUnit

Description

This property allows selection of the unit in which tolerance (REG_Tolerance) is specified.

If the tolerance type (REG_ToleranceType) is set to "offset" (**REG_TOLERANCETYPE_OFFSET**), then this property is ignored; the tolerance is *always* an offset in the same unit as the Result unit (REG_ResultUnit).

If the tolerance type is "ratio" (**REG_TOLERANCETYPE_RATIO**), then this property specifies the unit in which the tolerance is entered.

Values

The following values are allowed for the tolerance unit:

UNIT_RELATIVE_DB	Sets the tolerance unit to dB
UNIT_RELATIVE_PERCENT	Sets the tolerance unit to %

NB: **UNIT_RELATIVE_DB is only a valid unit when the Result to be regulated is an amplitude Result (e.g. Signal Analyzer RMS amplitude, or Continuous-Time or FFT Detector value).**

5.7.3.1.10 REG_Trend

Description

This property allows selection of the way the Result value varies with the Source. If the relationship between the Source and Result is known, it can help to speed up the regulation algorithm.

NB: **The trend of the Result within the selected Source range (REG_SourceMinLimit to REG_SourceMaxLimit) is important; the relationship between them outside this range is not important.**

Values

REG_TREND_INCREASING	Specifies that the Result increases monotonically as the Source is increased.
REG_TREND_DECREASING	Specifies that the Result decreases monotonically as the Source is increased.
REG_TREND_UNKNOWN	Specifies that the trend of the Result with respect to the Source is not known.
REG_TREND_NONMONOTONIC	Specifies that the Result is non-monotonic with respect to the Source.

5.7.3.1.11 REG_Sensitivity

Description

This property represents the sensitivity of the minimum and maximum regulation algorithms (REG_RegulationType) to local minima or maxima, in %.

When the algorithm passes a peak or trough, it will ignore it unless the difference between the peak/ trough and the last point read is at least as much as this sensitivity value.

A value of 100% will find the very first minimum/maximum of any size; a value of 0% will ensure that *every* peak or trough is considered, at the expense of taking longer to reach a conclusion.

Values

The sensitivity is represented as a double-precision floating point value.

NB: **This property is ignored unless the regulation type (REG_RegulationType) is set to be "minimum" or "maximum" (REG_REGULATIONTYPE_MINIMUM or REG_REGULATIONTYPE_MAXIMUM).**

5.7.3.1.12 REG_Timeout

Description

This property represents the timeout for the regulation algorithm, in seconds. If regulation does not complete within this time, regulation will stop. The Source and Result fields will be left at the points set by the regulation algorithm just before the regulation timed out.

NB: **If regulation is set up to regulate two channels separately (See REG_CHANNEL_BOTH in REG_Channel), then this timeout will apply to *each* stage of the regulation; i.e. the total timeout will be twice the value entered here.**

Values

The timeout is represented as a short integer value, and can be between 1 and 3600 seconds.

5.7.3.1.13 REG_Source

Description

This property allows selection of the source parameter to be varied. If relevant, the channel(s) to vary are determined by the channel set for the Result (REG_Channel) and whether to vary the same or the opposite channel of the source (REG_SourceOppChannel).

Values

REG_SOURCE_GENFREQ	Varies the Signal Generator frequency.
REG_SOURCE_GENAMPL	Varies the Signal Generator amplitude.
REG_SOURCE_DCOFFSET	Varies the Digital Outputs DC offset.
REG_SOURCE_JITTERFREQ	Varies the Digital Output Carrier jitter frequency.
REG_SOURCE_JITTERAMPL	Varies the Digital Output Carrier jitter amplitude.

5.7.3.1.14 REG_SourceOppChannel

Description

This property is used to select whether to vary the same channel of the source as selected for the Result (See REG_Channel), or to vary the opposite channel.

NB: **This property is ignored unless the selected source (REG_Source) can be varied on more than one channel - i.e. REG_SOURCE_GENFREQ or REG_SOURCE_GENAMPL.**

Values

True	Vary the opposite channel of the source to that selected by REG_Channel.
False	Vary the same channel of the source as selected by REG_Channel.

5.7.3.1.15 REG_SourceMinLimit

Description

This property specifies the minimum limit for variation of the source. The regulation algorithm will not set the source value lower than this value, and in some cases regulation will fail if this limit is reached before an end value is found.

The minimum limit must be specified in the unit selected by REG_SourceUnit.

Values

The minimum source limit is represented as a double-precision floating point value.

5.7.3.1.16 REG_SourceMaxLimit

Description

This property specifies the maximum limit for variation of the source. The regulation algorithm will not set the source value higher than this value, and in some cases regulation will fail if this limit is reached before an end value is found.

The maximum limit must be specified in the unit selected by REG_SourceUnit.

Values

The maximum source limit is represented as a double-precision floating point value.

5.7.3.1.17 REG_SourceUnit

Description

This property allows selection of the unit for the source value to vary (as specified by REG_Source). This unit affects the minimum and maximum limits of the source (REG_SourceMinLimit, REG_SourceMaxLimit) and the initial step size (REG_StepSize)

Values

The allowed values for the source unit depend on the currently selected source (REG_Source).

If the source is the Signal Generator frequency (**REG_SOURCE_GENFREQ**) then the following units are valid:

UNIT_FREQ_HZ	Sets source unit to Hz.
UNIT_FREQ_OFFSET	Sets source unit to offset from the reference frequency (SG_RefFreq), in Hz.
UNIT_FREQ_RATIO	Sets source unit to a ratio of the reference frequency (SG_RefFreq).

If the source is the Signal Generator amplitude (**REG_SOURCE_GENAMPL**) then the following units are valid:

UNIT_DBFS	Sets source unit to dBFS.
UNIT_PERCENTFS	Sets source unit to %FS (percentage of full scale).
UNIT_FFS	Sets source unit to FFS (fraction of full scale).
UNIT_HEX	Sets source unit to Hex.
UNIT_V	Sets source unit to V.
UNIT_DBU	Sets source unit to dBu.
UNIT_DBV	Sets source unit to dBV.
UNIT_DBM	Sets source unit to dBm.
UNIT_W	Sets source unit to W.
UNIT_DBSPL	Sets source unit to dBSPL.
UNIT_DBR	Sets source unit to dBr (dB with respect to the reference amplitude, SG_RefAmpl).
UNIT_PERCENTREF	Sets source unit to percentage of the reference amplitude SG_RefAmpl

If the source is the Digital Outputs DC Offset, then the following units are valid:

UNIT_DBFS	Sets source unit to dBFS.
UNIT_PERCENTFS	Sets source unit to %FS (percentage of full scale).
UNIT_FFS	Sets source unit to FFS (fraction of full scale).
UNIT_HEX	Sets source unit to Hex.

If the source is the Digital Output Carrier jitter frequency, then the source unit is **UNIT_FREQ_HZ** and cannot be changed.

If the source is the Digital Output Carrier jitter amplitude, then the following units are valid:

UNIT_JITTER_NS	Sets source unit to ns.
UNIT_JITTER_UI	Sets source unit to UI.

5.7.3.1.18 REG_StepSize

Description

This property specifies the *initial* step size to use when the regulation uses a stepping algorithm to find its end value. The stepping algorithm is used when the trend (REG_Trend) is non-monotonic, and will therefore apply when a minimum or maximum regulation type (REG_RegulationType) are selected. The step size is entered in the unit specified by REG_SourceUnit.

NB: **This property is ignored if the step size has been set to automatic (see REG_StepSizeAuto).**

Values

The initial step size is represented as a double-precision floating point value.

5.7.3.1.19 REG_StepSizeAuto

Description

This property is used to specify an automatic initial step size to use when the regulation uses a stepping algorithm to find its end value. The stepping algorithm is used when the trend (REG_Trend) is non-monotonic, and will therefore apply when a minimum or maximum regulation type (REG_RegulationType) are selected.

Values

True	Allow the regulation algorithm to automatically calculate an initial step size.
False	Do not calculate an initial step size automatically; use the step size specified by REG_StepSize.

5.7.3.1.20 REG_Direction

Description

This property allows selection of the direction to vary the source by when regulating. It is only relevant when a stepping algorithm is used (i.e. when the trend specified by REG_Trend is non-monotonic).

Values

REG_DIRECTION_UP	Starts by increasing the source from its current value.
REG_DIRECTION_DOWN	Starts by decreasing the source from its current value.
REG_DIRECTION_UNKNOWN	Specified that the direction to vary the source by is unknown. In this case, regulation will start the source value at its minimum value (as specified by REG_SourceMinLimit), and use a direction of **REG_DIRECTION_UP**.

5.7.3.2 Methods

5.7.3.2.1 REG_Start

bRet = REG_Start ()

This method starts the regulation process with the specified parameters.

Parameters

This method has no parameters.

Return value

This method returns **True** if the regulation started successfully, or **False** if it failed. This may happen if regulation is already running, or if any of the parameters entered are invalid with respect to other parameters (for example, if the step size set using REG_StepSize is greater than the range specified using REG_SourceMinLimit and REG_SourceMaxLimit).

5.7.3.2.2 REG_Stop

REG_Stop()

This method stops regulation, if it is currently running.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.7.3.2.3 REG_GetStatus

Description

This property returns the current status of the regulation process.

Values

REG_STATUS_NONE	Regulation has not yet been started.
REG_STATUS_INPROGRESS	Regulation is currently in progress
REG_STATUS_OK	The last regulation succeeded.
REG_STATUS_FAILED	The last regulation failed.

5.8 Trace window

The Trace Window section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"TraceWindow."**

For further information on creating and accessing Traces, see Creating and accessing Traces below.

Properties

TW_GraphTitle
TW_GraphComment
TW_EditImpulseWindow

Methods

TW_CreateTrace
TW_CreateTraceFromSweepTrace
TW_SetCurrentTrace
TW_SetCurrentTraceFromEventParam
TW_GetCurrentTrace
TW_RemoveTrace
TW_GetFirstTraceOfType
TW_GetNextTraceOfType
TW_Export
TW_Print
TW_Copy
TW_AutoZoomAll
TW_DefaultZoomAll

NB: **This list currently only gives basic automation functionality to the Trace Window; it is anticipated that more properties and methods will be added in due course.**

Creating and accessing Traces

Trace Window automation works by holding the notion of a current Trace. You can either create a user-defined Trace, or access an existing Trace, and this will result in a Trace ID being returned to you. You can then pass this Trace ID to the TW_SetCurrentTrace routine, and then any action performed on the Trace object will be performed on this Trace.

Creating a Trace is a simple matter of defining the X and Y scale ranges, and the number of points that the Trace will contain. The X and Y units, and Trace name and colour can also be set.

Accessing an existing Trace is even simpler. You simply decide what type of Trace it is you're trying to access, and which channel it's on, and the Trace ID will be returned to you.

For example : the following code example sets up a simple 5-point Trace that has an X axis in Hz and a Y axis in dBu.

```
sTraceID = TraceWindow.TW_CreateTrace (CHANNEL_A,
      5, 20, 20000, -30.0, 28.0)
If sTraceID <> TRACE_NULL_ID Then
   TraceWindow.TW_SetCurrentTrace sTraceID, True
   Trace.TRACE_XUnit = UNIT_FREQ_HZ
   Trace.TRACE_YUnit = UNIT_DBU
   Trace.TRACE_Name = "My new Trace"
```

```
    ' Trace is now ready to add points to...
End If
```

The following code will access the current live Scope Trace on channel B, and set it as the current Trace for further analysis.

```
sTraceID = TraceWindow.TW_GetFirstTraceOfType(TRACETYPE_SCOPE,
        CHANNEL_B)
TraceWindow.TW_SetCurrentTrace sTraceID, False
```

5.8.1 Properties

5.8.1.1 TW_GraphTitle

Description

This property can be used to set or read the title of the Graph. This title can be included when printing and exporting the Trace window (see TW_Print, TW_Export and TW_CopyToClipboard).

Values

Any valid string can be entered.

5.8.1.2 TW_GraphComment

Description

This property can be used to set or read the comment for the Graph. This comment can be included when printing and exporting the Trace window (see TW_Print, TW_Export and TW_CopyToClipboard).

Values

Any valid string can be entered.

5.8.1.3 TW_EditImpulseWindow

Description

This property can be used to toggle the Trace Window into and out of a mode where the Window Function for an Impulse Response can be edited by dragging. If the Impulse Response Window Function is not currently displayed on the Trace Window, it will be added.

NB: **Unlike entering this mode from the user interface, when called from a script, the "Edit Impulse Window" toolbar will *not* be displayed.**

Values

True	Enter Impulse Response Window editing mode.
False	Exit Impulse Response Window editing mode.

5.8.2 Methods

5.8.2.1 TW_CreateTrace

sTraceID = TW_CreateTrace (sChannel, lNumPoints, dXMin, dXMax, dYMin, dYMax)

This method creates a user-defined Trace on the Trace window.

For further details on how to create user-defined Traces, see Creating and accessing Traces.

Parameters

sChannel	The channel to create the Trace on (**CHANNEL_A** or **CHANNEL_B**).
lNumPoints	The number of points there will be in the Trace. NB: The more points there are in the Trace, the slower that access to and drawing of this Trace will be.
dXMin	The start X axis value for this Trace.
dXMax	The end X axis value for this Trace.
dYMin	The start Y axis value for this Trace.
dYMax	The end Y axis value for this Trace.

Return value

This method returns the Trace ID of the Trace just created, or **TRACE_NULL_ID** if it failed.

NB: **If this method succeeds, it automatically sets the current Trace to the Trace just created, so you don't need to call TW_SetCurrentTrace after using this method.**

5.8.2.2 TW_CreateTraceFromSweepTrace

sTraceID = TW_CreateTraceFromSweepTrace (sTraceType, sChannel)

This method creates a user-defined Trace on the Trace window, taking an existing Live Sweep Trace as the basis. This may be useful if you want to take a Sweep Trace and then manipulate the results in some way without altering the Sweep just performed.

For further details on how to create user-defined Traces, see Creating and accessing Traces.

Parameters

sTraceType	The Trace type of the live Sweep to copy. See the Trace types section below for the values to use here.
sChannel	The channel to create the Trace on.

Return value

This method returns the Trace ID of the Trace just created, or **TRACE_NULL_ID** if it failed.

NB: **If this method succeeds, it automatically sets the current Trace to the Trace just created, so you don't need to call TW_SetCurrentTrace after using this method.**

Trace types

The ***sTraceType*** parameter can have any of the following values:

TRACETYPE_SWEEP1	Selects the Sweep Trace to copy to be the current Live Sweep of the first Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP2	Selects the Sweep Trace to copy to be the current Live Sweep of the second Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP3	Selects the Sweep Trace to copy to be the current Live Sweep of the third Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP4	Selects the Sweep Trace to copy to be the current Live Sweep of the fourth Sweep Result from the Sweep Setup panel.

5.8.2.3 TW_SetCurrentTrace

bRet = TW_SetCurrentTrace (sTraceID, bUpdateDisplay)

This method sets the current Trace on the Trace window. All further actions carried out on a Trace will affect the Trace whose ID we pass to this method.

Parameters

sTraceID	The ID of the Trace to set as the current Trace. This will be the value returned from a routine such as TW_CreateTrace or TW_GetFirstTraceOfType.
bUpdateDisplay	Whether to set this as the current Trace for the display. If set to **True**, then the Trace will be set as the current Trace in the user interface and the user will be able to alter the Trace's details using the buttons and options on the Trace Window. If it is set to **False** however, then it will be the current Trace solely for automation purposes and the user interface will keep showing a different Trace as current.

Return value

This method returns **True** if it was successfully set as the current Trace, or **False** if the call failed (for example, if the Trace ID is invalid).

5.8.2.4 TW_GetCurrentTrace

sTraceID = TW_GetCurrentTrace ()

This method returns the unique Trace ID of the current Trace on the Trace window.

Parameters

This method has no parameters.

Return value

This method returns the unique Trace ID of the current Trace, or **TRACE_NULL_ID** if there is no current Trace.

5.8.2.5 TW_SetCurrentTraceFromEventParam

bRet = TW_SetCurrentTraceFromEventParam (lParam, bUpdateDisplay)

The dScope Event Manager can be set up to fire an event to a script when a Trace's upper or lower limit is breached. When it does so, the event subroutine is passed a parameter that represents the Trace whose limit was breached.

This method can be used from within one of these event subroutines to set the current Trace from the parameter passed. All further actions carried out on a Trace will affect the Trace that has been set using this method.

NB: **This method should ONLY be called from within either the Event_TraceMinLimit or the Event_TraceMaxLimit subroutine.**

Parameters

lParam	The parameter passed to the event subroutine, which should be passed unchanged to this method.
bUpdateDisplay	Whether to set this as the current Trace for the display. If set to **True**, then the Trace will be set as the current Trace in the user interface and the user will be able to alter the Trace's details using the buttons and options on the Trace Window. If it is set to **False** however, then it will be the current Trace solely for automation purposes and the user interface will keep showing a different Trace as current.

NB: **The lParam is NOT the same as a Trace ID, and should not be used as such.**

Return value

This method returns **True** if the current Trace was successfully set, or **False** if the call failed (for example, if the lParam is invalid).

5.8.2.6 TW_RemoveTrace

bRet = TW_RemoveTrace (sTraceID)

This method removes a Trace from the Trace window.

If the Trace to be removed is user-defined, or is a copy of a Live Trace, this method will delete the Trace completely. If it is a Live Trace, it will be removed from the Trace window but can be added again later.

Parameters

sTraceID	The ID of the Trace to remove. This will be the value returned from a

routine such as TW_CreateTrace or TW_GetFirstTraceOfType.

Return value

This method returns **True** if the Trace is successfully removed, or **False** if it failed. This may be because the Trace ID passed is invalid.

NB: **If the Trace to be removed is the current Trace, the next Trace in the Trace Window's legend will be selected as the current Trace.**

5.8.2.7 TW_LoadTrace

sTraceID = TW_LoadTrace (strFileName, sChannel)

This method loads a previously saved Trace into the Trace window.

Parameters

strFileName	File name of the Trace file to load, enclosed in double quotation marks ("..."). If a full path name is specified, the system will look for this exact file. If a file name only is specified, then the system will look in the folder specified in the Options dialogue box for Traces (See OPT_TracesFolder). If necessary, dScope will automatically append the file extension for Trace files (".tra").
sChannel	Which channel of the Trace window to load the Trace onto (**CHANNEL_A** or **CHANNEL_B**).

Return value

This method returns the Trace ID of the Trace just loaded, or **TRACE_NULL_ID** if it failed. This may be because the file specified does not exist.

5.8.2.8 TW_GetFirstTraceOfType

sTraceID = TW_GetFirstTraceOfType (sTraceType, sChannel)

This method gets the first Trace of a given type on the Trace Window.

Parameters

sTraceType	The Trace type of the Trace to get. See the Trace types section below for the values to use here.
sChannel	The channel that the Trace to get is on. See the Channels section below for the values to use here.

Return value

This method returns the Trace ID of the first Trace of the given type, **TRACE_NULL_ID** if one is not found.

NB: **If this method succeeds, it automatically sets the current Trace to the Trace just created, so you don't need to call TW_SetCurrentTrace after using this method.**

Trace types

The ***sTraceType*** parameter can have any of the following values:

TRACETYPE_SCOPE	Specifies that the Trace to look for should be a Scope Trace.
TRACETYPE_FFT	Specifies that the Trace to look for should be an FFT Trace.
TRACETYPE_CTA	Specifies that the Trace to look for should be a CT Detector Trace (i.e. Trace of the Continuous-Time Detector output).
TRACETYPE_CTFFT	Specifies that the Trace to look for should be a CT Detector FFT Trace (i.e. an FFT Trace of the Continuous-Time Detector output).
TRACETYPE_FILTER	Specifies that the Trace to look for should be a Filter Trace.
TRACETYPE_WINDOW	Specifies that the Trace to look for should be a Window Function Trace.
TRACETYPE_SWEEP1	Specifies that the Trace to look for should be a Sweep of the first Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP2	Specifies that the Trace to look for should be a Sweep of the second Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP3	Specifies that the Trace to look for should be a Sweep of the third Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP4	Specifies that the Trace to look for should be a Sweep of the fourth Sweep Result from the Sweep Setup panel.
TRACETYPE_USER	Specifies that the Trace to look for should be a user-defined Trace.

Channels

The ***sChannel*** parameter can have any of the following values:

TW_CHA	Specifies that the Trace to look for should contain data from channel A.
TW_CHB	Specifies that the Trace to look for should contain data from channel B.
TW_CHBOTH	Specifies that the Trace to look for can contain data from either channel.

5.8.2.9 TW_GetNextTraceOfType

sTraceID = TW_GetNextTraceOfType (sTraceType)

This method gets the next Trace of a given type on the Trace Window. This allows you to obtain a Trace by cycling through all Traces of a given type using TW_GetFirstTraceOfType and this method (see example, below).

Parameters

sTraceType	The Trace type of the Trace to get. See the Trace types section below for the values to use here.

Return value

This method returns the Trace ID of the first Trace of the given type, **TRACE_NULL_ID** if one is not found.

NB: **If this method succeeds, it automatically sets the current Trace to the Trace just created, so you don't need to call TW_SetCurrentTrace after using this method.**

Trace types

The ***sTraceType*** parameter can have any of the following values:

TRACETYPE_SCOPE	Specifies that the Trace to look for should be a Scope Trace.
TRACETYPE_FFT	Specifies that the Trace to look for should be an FFT Trace.
TRACETYPE_CTA	Specifies that the Trace to look for should be a CT Detector Trace (i.e. Trace of the Continuous-Time Detector output).
TRACETYPE_CTFFT	Specifies that the Trace to look for should be a CT Detector FFT Trace (i.e. an FFT Trace of the Continuous-Time Detector output).
TRACETYPE_FILTER	Specifies that the Trace to look for should be a Filter Trace.
TRACETYPE_WINDOW	Specifies that the Trace to look for should be a Window Function Trace.
TRACETYPE_SWEEP1	Specifies that the Trace to look for should be a Sweep of the first Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP2	Specifies that the Trace to look for should be a Sweep of the second Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP3	Specifies that the Trace to look for should be a Sweep of the third Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP4	Specifies that the Trace to look for should be a Sweep of the fourth Sweep Result from the Sweep Setup panel.
TRACETYPE_USER	Specifies that the Trace to look for should be a user-defined Trace.

Example

The following code snippet will get the last Scope Trace (live, or copied) that was entered on the system:

```
sLastTraceID = TRACE_NULL_ID
sTraceID = TraceWindow.TW_GetFirstTraceOfType(TRACETYPE_SCOPE, CHANNEL_A)

' Loop until Scope Traces run out.
While sTraceID <> TRACE_NULL_ID
   sLastTraceID = sTraceID
   sTraceID = TraceWindow.TW_GetNextTraceOfType(TRACETYPE_SCOPE)
Wend

' Now we've dropped out of the loop, sLastTraceID
' holds the last Trace of the right type.
If sLastTraceID <> TRACE_NULL_ID Then
   TraceWindow.TW_SetCurrentTrace(sLastTraceID)
End If
```

5.8.2.10 TW_CopyTrace

sTraceID = TW_CopyTrace (sTraceID)

This method copies a Trace on the Trace window.

Parameters

sTraceID	The Trace ID of the Trace to copy.

Return value

This method returns the Trace ID of the Trace created by the copy operation, **TRACE_NULL_ID** if the operation failed.

NB: **Unlike TW_GetFirstTraceOfType and TW_GetNextTraceOfType, this method does NOT set the current Trace to the one that was just created by the copy operation. To set this as the current Trace, use the TW_SetCurrentTrace method with the Trace ID returned.**

5.8.2.11 TW_Export

bRet = TW_Export (sChannel, strFileName)

This method exports the graph for the specified channel.

Parameters

sChannel The channel to export the graph for (**CHANNEL_A**, **CHANNEL_B** or **CHANNEL_BOTH**).
NB: **CHANNEL_BOTH** will only work as a parameter if all Traces are currently being shown on the same View.

strFileName The file name to export to.

Any valid file name can be used, enclosed in double quotation marks ("..."). The type of file created will depend on the file extension entered:

emf	Windows Enhanced Metafile.
bmp	24-bit colour Bitmap file.
jpg	JPEG (Joint Photographic Experts Group) file.
gif	GIF (Graphic Interchange Format) file
tif	TIFF (Tagged Image File Format) file.
png	PNG (Portable Network Graphics) file.

e.g. using a file name of "Test.jpg" will cause the file to be saved as a JPEG. If the file extension cannot be determined, the file will be saved as a Windows Enhanced Metafile (emf).

If a full path name is specified, the system will save the file as specified.
If a file name only is specified, then the system will save the file in the folder specified in the Options dialogue box for Graph exports (See OPT_GraphExportsFolder).

If a file extension is not specified, the system will automatically append a file extension of ".emf" (Windows Enhanced Metafile).

Return value

This method returns **True** if the graph was exported successfully, or **False** if it failed. This may be because the file name passed is invalid.

NB: **The Trace window must be open on the current Page for this method to work, and there must NOT be a Print Preview or Export Preview window open.**

5.8.2.12 TW_Print

bRet = TW_Print (sChannel)

This method prints the graph for the specified channel.

Parameters

sChannel	The channel to print the graph for (**CHANNEL_A**, **CHANNEL_B** or **CHANNEL_BOTH**). NB: **CHANNEL_BOTH** will only work as a parameter if all Traces are currently being shown on the same View.

Return value

This method returns **True** if the graph was printed successfully, or **False** if it failed.

NB: **The Trace window must be open on the current Page for this method to work.**

5.8.2.13 TW_CopyToClipboard

bRet = TW_CopyToClipboard (sChannel)

This method copies the graph for the specified channel to the Windows Clipboard.

Parameters

sChannel	The channel to copy the graph for (**CHANNEL_A**, **CHANNEL_B** or **CHANNEL_BOTH**). NB: **CHANNEL_BOTH** will only work as a parameter if all Traces are currently being shown on the same View.

Return value

This method returns **True** if the graph was copied successfully, or **False** if it failed.

NB: **The Trace window must be open on the current Page for this method to work.**

5.8.2.14 TW_AutoZoomAll

TW_AutoZoomAll (sChannel)

This method auto-zooms all Traces on the specified channel.

For further details of how individual Traces are auto-zoomed, see TRACE_AutoZoomX and TRACE_AutoZoomY.

Parameters

sChannel	The channel to auto-zoom the Traces on.

Return value

This method has no return value.

5.8.2.15 TW_DefaultZoomAll

TW_DefaultZoomAll (sChannel)

This method zooms all Traces on the specified channel to their default values.

For further details of default zoom values for individual Traces, see TRACE_DefaultZoomX and TRACE_DefaultZoomY.

Parameters

sChannel	The channel on which to set the Traces to their default zoom values.

Return value

This method has no return value.

5.8.3 Trace

The Trace section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"Trace."**

Before using any of the following properties and methods, the current Trace must be selected. See TW_SetCurrentTrace for further details. Note that where the "current Trace" is mentioned in this section, it is the current Trace as set for automation, and not necessarily the current Trace as selected on the Trace Window.

For further information on how to create or select a Trace to apply these methods and properties to, see Creating and accessing Traces.

Properties

Methods

5.8.3.1 Properties

5.8.3.1.1 TRACE_Name

Description

This property can be used to set or read the name of the current Trace.

Values

Any string can be entered up to 32 characters long.

5.8.3.1.2 TRACE_ID

Description

This **read-only** property returns the unique ID of the Trace. This ID is used when setting and retrieving the current Trace using TW_SetCurrentTrace and TW_GetCurrentTrace.

Values

This property will return an integer value.

5.8.3.1.3 TRACE_Type

Description

This **read-only** property returns the type of the Trace.

Values

This property will return one of the following values:

TRACETYPE_SCOPE	Indicates that the Trace is a Scope Trace.
TRACETYPE_FFT	Indicates that the Trace is an FFT Trace.
TRACETYPE_CTA	Indicates that the Trace is a CT Detector Trace (i.e. Trace of the Continuous-Time Detector output).
TRACETYPE_CTFFT	Indicates that the Trace is a CT Detector FFT Trace (i.e. an FFT Trace of the Continuous-Time Detector output).
TRACETYPE_FILTER	Indicates that the Trace is a Filter Trace.
TRACETYPE_WINDOW	Indicates that the Trace is a Window Function Trace.
TRACETYPE_SWEEP1	Indicates that the Trace is a Sweep of the first Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP2	Indicates that the Trace is a Sweep of the second Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP3	Indicates that the Trace is a Sweep of the third Sweep Result from the Sweep Setup panel.
TRACETYPE_SWEEP4	Indicates that the Trace is a Sweep of the fourth Sweep Result from the Sweep Setup panel.
TRACETYPE_USER	Indicates that the Trace is a user-defined Trace.
TRACETYPE_MINLIMITLINE	Indicates that the Trace is a lower Limit Line for another Trace.

TRACETYPE_MAXLIMITLINE	Indicates that the Trace is an upper Limit Line for another Trace.

5.8.3.1.4 TRACE_Channel

Description

This **read-only** property returns the channel that this Trace contains data from.

Values

This property will return one of the following values:

TW_CHA	Specifies that the Trace contains data from channel A.
TW_CHB	Specifies that the Trace contains data from channel B.

5.8.3.1.5 TRACE_XUnit

Description

This property represents the X unit of the current Trace.

Values

If the Trace is an existing Trace, then the unit must be a valid unit for the type of Trace concerned.

If the Trace is a user-defined Trace created using TW_CreateTrace, then this can be ANY valid unit allowed in the dScope. The following list details all the units available:

UNIT_MS	Sets the Trace's X unit to milliseconds.
UNIT_FREQ_HZ	Sets the Trace's X unit to Hz.
UNIT_DBFS	Sets the Trace's X unit to dBFS.
UNIT_PERCENTFS	Sets the Trace's X unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Trace's X unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Trace's X unit to Hex.
UNIT_V	Sets the Trace's X unit to Volts.
UNIT_VRMS	Sets the Trace's X unit to Volts, RMS.
UNIT_VP	Sets the Trace's X unit to Volts, peak.
UNIT_VPP	Sets the Trace's X unit to Volts, peak-to-peak.
UNIT_DBU	Sets the Trace's X unit to dBu.
UNIT_DBV	Sets the Trace's X unit to dBV.
UNIT_DBM	Sets the Trace's X unit to dBm.
UNIT_W	Sets the Trace's X unit to W.
UNIT_DBSPL	Sets the Trace's X unit to dBSPL.
UNIT_DBR	Sets the Trace's X unit to dBr (dB, relative to the reference amplitude specified using SA_RefAmpl).
UNIT_PERCENTREF	Sets the Trace's X unit to a percentage of the reference amplitude, specified using SA_RefAmpl.
UNIT_RELATIVE_DB	Sets the Trace's X unit to dB.
UNIT_RELATIVE_PERCENT	Sets the Trace's X unit to percent.
UNIT_JITTER_UI	Sets the Trace's X unit to UI.
UNIT_JITTER_NS	Sets the Trace's X unit to ns.

UNIT_PHASE_SAMPLES	Sets the Trace's X unit to be samples.
UNIT_PHASE_DEGREES	Sets the Trace's X unit to be degrees.
UNIT_PHASE_RADIANS	Sets the Trace's X unit to be radians.
UNIT_PHASE_US	Sets the Trace's X unit to be microseconds.
UNIT_PPM	Sets the Trace's X unit to be ppm (parts per million).

NB: **When specified for a user-defined Trace, this unit is solely for display, and has no use for other purposes such as unit conversion.**

5.8.3.1.6 TRACE_YUnit

Description

This property represents the Y unit of the current Trace.

Values

If the Trace is an existing Trace, then the unit must be a valid unit for the type of Trace concerned.

If the Trace is a user-defined Trace created using TW_CreateTrace, then this can be ANY valid unit allowed in the dScope. The following list details all the units available:

UNIT_MS	Sets the Trace's Y unit to milliseconds.
UNIT_FREQ_HZ	Sets the Trace's Y unit to Hz.
UNIT_DBFS	Sets the Trace's Y unit to dBFS.
UNIT_PERCENTFS	Sets the Trace's Y unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Trace's Y unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Trace's Y unit to Hex.
UNIT_V	Sets the Trace's Y unit to Volts.
UNIT_VRMS	Sets the Trace's Y unit to Volts, RMS.
UNIT_VP	Sets the Trace's Y unit to Volts, peak.
UNIT_VPP	Sets the Trace's Y unit to Volts, peak-to-peak.
UNIT_DBU	Sets the Trace's Y unit to dBu.
UNIT_DBV	Sets the Trace's Y unit to dBV.
UNIT_DBM	Sets the Trace's Y unit to dBm.
UNIT_W	Sets the Trace's Y unit to W.
UNIT_DBSPL	Sets the Trace's Y unit to dBSPL.
UNIT_DBR	Sets the Trace's Y unit to dBr (dB, relative to the reference amplitude specified using SA_RefAmpl).
UNIT_PERCENTREF	Sets the Trace's Y unit to a percentage of the reference amplitude, specified using SA_RefAmpl.
UNIT_RELATIVE_DB	Sets the Trace's Y unit to dB.
UNIT_RELATIVE_PERCENT	Sets the Trace's Y unit to percent.
UNIT_RELATIVE_GAIN	Sets the Trace's Y unit to a gain factor, relative to the specified value.
UNIT_RELATIVE_ANAVSGEN	Sets the Trace's Y unit to a special unit relating the input level to a set level on the Signal Generator. For example, if the Signal Analyzer unit (SA_RMSAmplUnit) is set to dB SPL and the Signal Generator unit (SG_ChAAmplUnit) is set to V (RMS), then this unit will show dB SPL / 1V (RMS), i.e. the input level in dB SPL that corresponds to 1V (RMS). NB: This unit is only available if the Trace has been created from an impulse response
UNIT_JITTER_UI	Sets the Trace's Y unit to UI.

UNIT_JITTER_NS	Sets the Trace's Y unit to ns.
UNIT_PHASE_SAMPLES	Sets the Trace's Y unit to be samples.
UNIT_PHASE_DEGREES	Sets the Trace's Y unit to be degrees.
UNIT_PHASE_RADIANS	Sets the Trace's Y unit to be radians.
UNIT_PHASE_US	Sets the Trace's Y unit to be microseconds.
UNIT_PPM	Sets the Trace's Y unit to be ppm (parts per million).

NB: **When specified for a user-defined Trace, this unit is solely for display, and has no use for other purposes such as unit conversion.**

5.8.3.1.7 TRACE_On

Description

This property can be used to turn the current Trace on or off.

NB: **Note that if a Trace is turned off, it can no longer be the current Trace. The next Trace in the legend will be selected as the current Trace in this situation.**

Values

True	Turn the current Trace On.
False	Turn the current Trace Off.

5.8.3.1.8 TRACE_Comment

Description

This property can be used to set or read the comment for the current Trace.

Values

Any valid string can be entered.

5.8.3.1.9 TRACE_PrintStyle

Description

This property can be used to set the print style of the current Trace. This is the way that the Trace will be drawn when the Trace Window is printed or exported using TW_Print or TW_Export.

Values

TRACE_PRINTSTYLE_SOLID	Sets the Trace's print style to be a solid line.
TRACE_PRINTSTYLE_DOT	Sets the Trace's print style to be a dotted line.
TRACE_PRINTSTYLE_DASH	Sets the Trace's print style to be a dashed line.
TRACE_PRINTSTYLE_DASHDOT	Sets the Trace's print style to be a line consisting of alternating dashes and dots.
TRACE_PRINTSTYLE_DASHDOTDOT	Sets the Trace's print style to be a line consisting of alternating dashes followed by two dots.

5.8.3.1.10 TRACE_ShowTransformedData

Description

This property can be used to turn the current Trace on or off.

NB: **This property will be ignored unless the Trace has some transformations set up. See Trace Transformations for further details.**

Values

True	Shows the Trace after performing transformation on the data.
False	Shows the Trace without performing transformation on the data.

5.8.3.1.11 TRACE_MaxLimitBreached

Description

This property can be used to determine whether a Trace has breached its upper Limit Line. It can be set to False before performing an operation which could cause the limit to be breached, and then checked at the end to see if the limit has been breached.

Values

True	Trace's upper Limit Line has been breached.
False	Trace's upper Limit Line has not been breached.

NB: **If the Trace is a Sweep Trace, then this property will be reset to False automatically when a Sweep is started. For all other trace types, the property must be reset explicitly by the script.**

5.8.3.1.12 TRACE_MinLimitBreached

Description

This property can be used to determine whether a Trace has breached its lower Limit Line. It can be set to False before performing an operation which could cause the limit to be breached, and then checked at the end to see if the limit has been breached.

Values

True	Trace's lower Limit Line has been breached.
False	Trace's lower Limit Line has not been breached.

NB: **If the Trace is a Sweep Trace, then this property will be reset to False automatically when a Sweep is started. For all other trace types, the property must be reset explicitly by the script.**

5.8.3.1.13 TRACE_CursorOn

Description

This property can be used to turn the current Trace's Cursor on or off.

Values

True	Turn the Cursor On.
False	Turn the Cursor Off.

5.8.3.1.14 TRACE_CursorXValue

Description

This **read-only** property represents the X value of the current Trace's Cursor.

The value is returned in the current Cursor X unit, as specified by TRACE_CursorXUnit.

NB: **If the current Trace's Cursor is off, the value TRACE_NULL_VALUE will be returned.**

Values

The X value of the Cursor is represented as a double-precision floating point value.

5.8.3.1.15 TRACE_CursorXUnit

Description

This property represents the unit in which the Cursor X value (see TRACE_CursorXValue) is returned.

Values

The Cursor X unit must be a valid unit for the X axis of the Trace concerned. The following list details all the units available:

UNIT_MS	Sets the Cursor's X unit to milliseconds.
UNIT_FREQ_HZ	Sets the Cursor's X unit to Hz.
UNIT_DBFS	Sets the Cursor's X unit to dBFS.
UNIT_PERCENTFS	Sets the Cursor's X unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Cursor's X unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Cursor's X unit to Hex.
UNIT_V	Sets the Cursor's X unit to Volts.
UNIT_VRMS	Sets the Cursor's X unit to Volts, RMS.
UNIT_VP	Sets the Cursor's X unit to Volts, peak.
UNIT_VPP	Sets the Cursor's X unit to Volts, peak-to-peak.
UNIT_DBU	Sets the Cursor's X unit to dBu.
UNIT_DBV	Sets the Cursor's X unit to dBV.
UNIT_DBM	Sets the Cursor's X unit to dBm.

UNIT_W	Sets the Cursor's X unit to W.
UNIT_DBSPL	Sets the Cursor's X unit to dBSPL.
UNIT_DBR	Sets the Cursor's X unit to dBr (dB, relative to the reference amplitude specified using SA_RefAmpl).
UNIT_PERCENTREF	Sets the Cursor's X unit to a percentage of the reference amplitude, specified using SA_RefAmpl.
UNIT_RELATIVE_DB	Sets the Cursor's X unit to dB.
UNIT_RELATIVE_PERCENT	Sets the Cursor's X unit to percent.
UNIT_JITTER_UI	Sets the Cursor's X unit to UI.
UNIT_JITTER_NS	Sets the Cursor's X unit to ns.
UNIT_PHASE_SAMPLES	Sets the Cursor's X unit to be samples.
UNIT_PHASE_DEGREES	Sets the Cursor's X unit to be degrees.
UNIT_PHASE_RADIANS	Sets the Cursor's X unit to be radians.
UNIT_PHASE_US	Sets the Cursor's X unit to be microseconds.
UNIT_PPM	Sets the Cursor's X unit to be ppm (parts per million).

5.8.3.1.16 TRACE_CursorYValue

Description

This **read-only** property represents the Y value of the current Trace's Cursor.

The value is returned in the current Cursor Y unit, as specified by TRACE_CursorYUnit.

NB: **If the current Trace's Cursor is off, the value TRACE_NULL_VALUE will be returned.**

Values

The Y value of the Cursor is represented as a double-precision floating point value.

5.8.3.1.17 TRACE_CursorYUnit

Description

This property represents the unit in which the Cursor Y value (see TRACE_CursorYValue) is returned.

Values

The Cursor Y unit must be a valid unit for the Y axis of the Trace concerned. The following list details all the units available:

UNIT_MS	Sets the Cursor's Y unit to milliseconds.
UNIT_FREQ_HZ	Sets the Cursor's Y unit to Hz.
UNIT_DBFS	Sets the Cursor's Y unit to dBFS.
UNIT_PERCENTFS	Sets the Cursor's Y unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Cursor's Y unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Cursor's Y unit to Hex.
UNIT_V	Sets the Cursor's Y unit to Volts.
UNIT_VRMS	Sets the Cursor's Y unit to Volts, RMS.
UNIT_VP	Sets the Cursor's Y unit to Volts, peak.

UNIT_VPP	Sets the Cursor's Y unit to Volts, peak-to-peak.
UNIT_DBU	Sets the Cursor's Y unit to dBu.
UNIT_DBV	Sets the Cursor's Y unit to dBV.
UNIT_DBM	Sets the Cursor's Y unit to dBm.
UNIT_W	Sets the Cursor's Y unit to W.
UNIT_DBSPL	Sets the Cursor's Y unit to dBSPL.
UNIT_DBR	Sets the Cursor's Y unit to dBr (dB, relative to the reference amplitude specified using SA_RefAmpl).
UNIT_PERCENTREF	Sets the Cursor's Y unit to a percentage of the reference amplitude, specified using SA_RefAmpl.
UNIT_RELATIVE_DB	Sets the Cursor's Y unit to dB.
UNIT_RELATIVE_PERCENT	Sets the Cursor's Y unit to percent.
UNIT_JITTER_UI	Sets the Cursor's Y unit to UI.
UNIT_JITTER_NS	Sets the Cursor's Y unit to ns.
UNIT_PHASE_SAMPLES	Sets the Cursor's Y unit to be samples.
UNIT_PHASE_DEGREES	Sets the Cursor's Y unit to be degrees.
UNIT_PHASE_RADIANS	Sets the Cursor's Y unit to be radians.
UNIT_PHASE_US	Sets the Cursor's Y unit to be microseconds.
UNIT_PPM	Sets the Cursor's Y unit to be ppm (parts per million).

5.8.3.1.18 TRACE_MarksOn

Description

This property can be used to turn the current Trace's Marks on or off.

Values

True	Turn the Marks On.
False	Turn the Marks Off.

5.8.3.2 Methods

5.8.3.2.1 TRACE_DrawTrace

TRACE_DrawTrace ()

This method draws or redraws the Trace on the Trace Window.

Certain of the methods available to Traces (for example, TRACE_SetPoint) allow changing of the Trace's details without updating the display. This allows multiple changes to be made, and the display can then be updated in one go using this method. This reduces flickering and reduces unnecessary delay caused by updating the display every time a point is added.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.8.3.2.2 TRACE_SetColour

TRACE_SetColour (sRed, sGreen, sBlue)

This method sets the colour of a Trace on the Trace window.

By default, user-defined Traces start off as white. All other Traces have a default colour that can be changed.

Parameters

sRed	Specifies the red component of the colour, from 0 to 255.
sBlue	Specifies the blue component of the colour, from 0 to 255.
sGreen	Specifies the green component of the colour, from 0 to 255.

Right-clicking on a Trace in the Quick legend and selecting "Change Trace colour" will bring up a dialogue box which will allow you to find the red, green and blue components of common colours.

Return value

This method has no return value.

5.8.3.2.3 TRACE_SaveTrace

bRet = TRACE_SaveTrace (strFileName)

This method saves the current Trace to the specified file name.

Parameters

strFileName The file name to save this Trace to. Any valid file name can be used, enclosed in double quotation marks ("...").

If a full path name is specified, the system will save the file as specified. If a file name only is specified, then the system will look in the folder specified in the Options dialogue box for Traces (See OPT_TracesFolder).

If necessary, the system will automatically append the file extension for Trace files (".tra").

Return value

This method returns **True** if the Trace was saved successfully, or **False** if it failed. This may be because the file name passed was invalid.

5.8.3.2.4 TRACE_GetNumPoints

lNumPoints = TRACE_GetNumPoints ()

This method returns the number of points in the current Trace.

Parameters

This method has no parameters.

Return value

This method returns the number of points in the Trace.

5.8.3.2.5 TRACE_GetXValueAt

dXValue = TRACE_GetXValueAt (lPos)

This method returns the X value at the given point in the Trace. The value will be in the unit specified by TRACE_XUnit.

Parameters

lPos	The position in the Trace (zero-indexed) to get the X value for. This must be a value between 0 and one less than the value returned by TRACE_GetNumPoints.

Return value

This method returns the X value at the given position in the Trace.

5.8.3.2.6 TRACE_GetYValueAt

dXValue = TRACE_GetYValueAt (lPos)

This method returns the Y value at the given point in the Trace. The value will be in the unit specified by TRACE_YUnit.

Parameters

lPos	The position in the Trace (zero-indexed) to get the Y value for. This must be a value between 0 and one less than the value returned by TRACE_GetNumPoints.

Return value

This method returns the Y value at the given position in the Trace.

5.8.3.2.7 TRACE_SetPoint

bRet = TRACE_SetPoint (lPos, dXValue, dYValue, bUpdateDisplay)

This method allows you to set a value in a user-defined Trace. After calling TW_CreateTrace to create a user-defined Trace, this method allows specification of each of the points in the Trace.

NB: **This method will be ignored unless the Trace is a user-defined Trace or a Limit Line.**

Parameters

lPos	The position in the Trace (zero-indexed) to get the X value for. This must be a value between 0 and one less than the value returned by TRACE_GetNumPoints.
dXValue	The X value of the point to add. This can be any value, but should be a valid value for the unit specified using TRACE_XUnit.
dYValue	The Y value of the point to add. This can be any value, but should be a valid value for the unit specified using TRACE_YUnit.
bUpdateDisplay	**True** to update the Trace Window display after adding this point, or **False** to add the point without updating the display.

NB: **The TRACE_DrawTrace method is provided to allow multiple points to be added with *bUpdateDisplay* parameter set to False, and then all the drawing can be done in one go at the end to reduce flicker and save time.**

Return value

This method returns **True** if the point was added successfully, or **False** if it failed. This may be because the position passed was invalid.

5.8.3.2.8 TRACE_GetXRange

TRACE_GetXRange (pdMinValue, pdMaxValue)

This method gets the current X range of the Trace, in the current unit (as specified using the TRACE_XUnit property).

Parameters

pdMinValue	After this method is called, this parameter will hold the current minimum X value (in the unit specified by TRACE_XUnit).
pdMaxValue	After this method is called, this parameter will hold the current maximum X value (in the unit specified by TRACE_XUnit).

Return value

This method has no return value.

5.8.3.2.9 TRACE_GetFullXRange

TRACE_GetFullXRange (pdMinValue, pdMaxValue)

This method gets the full allowed X range of the Trace, in the current unit (as specified using the TRACE_XUnit property).

Parameters

pdMinValue	After this method is called, this parameter will hold the minimum allowed X value (in the unit specified by TRACE_XUnit).
pdMaxValue	After this method is called, this parameter will hold the maximum allowed X value (in the unit specified by TRACE_XUnit).

Return value

This method has no return value.

5.8.3.2.10 TRACE_SetXRange

bRet = TRACE_SetXRange (dMinValue, dMaxValue)

This method sets the range for the X axis of the current Trace. The minimum and maximum values must be entered in the unit specified using the TRACE_XUnit property.

Parameters

dMinValue	The minimum value for the Trace's X axis.
dMaxValue	The maximum value for the Trace's X axis.

NB: **For user-defined Traces, once the X range has been set, the Trace can be zoomed into a smaller range, but when zoomed out, it cannot go beyond the minimum and maximum specified here.**

Return value

This method returns **True** if the function is successful, or **False** if it fails. This may be because one or both of the values passed are invalid for the Trace's X unit.

5.8.3.2.11 TRACE_SetXIntervals

bRet = TRACE_SetXIntervals (sNumIntervals, bLog)

This method sets the number of intervals to display on the current Trace's X axis.

Parameters

sNumIntervals	The number of intervals to display on the Trace's X axis. Use **TRACE_INTERVALS_AUTO** to specify that the dScope should automatically calculate the intervals.
bLog	**True** to display the axis logarithmically, **False** to display it linearly.

NB: **If the scale is set to logarithmic, and the X scale starts at 0, the minimum X value will be adjusted when displayed to allow it to be shown logarithmically.**

Return value

This method returns **True** if the function is successful, or **False** if it fails. This may be because the number of intervals is invalid.

5.8.3.2.12 TRACE_GetXIntervals

TRACE_GetXIntervals (psNumIntervals, pbLog)

This method retrieves the number of intervals to display on the current Trace's X axis, and whether it is currently displayed as linear or logarithmic .

Parameters

psNumIntervals	This will return the number of intervals displayed on the Trace's X axis, or **TRACE_INTERVALS_AUTO** if the axis is set up to calculate intervals automatically.
pbLog	This will return **True** if the axis is currently displayed logarithmically, **False** if it is displayed linearly.

Return value

This method has no return value.

5.8.3.2.13 TRACE_AutoZoomX

TRACE_AutoZoomX ()

This method auto-zooms the X axis of the current Trace.

Automatic zooming of a Trace's X -axis will depend on the type of Trace. For Scope Traces (**TRACETYPE_SCOPE**) and CTA Traces (**TRACETYPE_CTA**), dScope will attempt to zoom to a couple of periods of the waveform. For other types of Trace, the X axis will be zoomed out to its maximum range.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.8.3.2.14 TRACE_DefaultZoomX

TRACE_DefaultZoomX ()

This method zooms the X axis of the current Trace to its default values.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.8.3.2.15 TRACE_GetYRange

TRACE_GetYRange (pdMinValue, pdMaxValue)

This method gets the current Y range of the Trace, in the current unit (as specified using the TRACE_YUnit property).

Parameters

pdMinValue	After this method is called, this parameter will hold the current minimum Y value (in the unit specified by TRACE_YUnit).
pdMaxValue	After this method is called, this parameter will hold the current maximum Y value (in the unit specified by TRACE_YUnit).

Return value

This method has no return value.

5.8.3.2.16 TRACE_GetFullYRange

TRACE_GetYRange (pdMinValue, pdMaxValue)

This method gets the maximum allowed Y range of the Trace, in the current unit (as specified using the TRACE_YUnit property).

Parameters

pdMinValue After this method is called, this parameter will hold the minimum allowed Y value (in the unit specified by TRACE_YUnit).

pdMaxValue After this method is called, this parameter will hold the maximum allowed Y value (in the unit specified by TRACE_YUnit).

Return value

This method has no return value.

5.8.3.2.17 TRACE_SetYRange

bRet = TRACE_SetYRange (dMinValue, dMaxValue)

This method sets the range for the Y axis of the current Trace. The minimum and maximum values must be entered in the unit specified using the TRACE_YUnit property.

Parameters

dMinValue The minimum value for the Trace's Y axis.

dMaxValue The maximum value for the Trace's Y axis.

NB: **For user-defined Traces, once the Y range has been set, the Trace can be zoomed into a smaller range, but when zoomed out, it cannot go beyond the minimum and maximum specified here.**

Return value

This method returns **True** if the function is successful, or **False** if it fails. This may be because one or both of the values passed are invalid for the Trace's Y unit.

5.8.3.2.18 TRACE_SetYIntervals

bRet = TRACE_SetYIntervals (sNumIntervals, bLog)

This method sets the number of intervals to display on the current Trace's Y axis.

Parameters

sNumIntervals The number of intervals to display on the Trace's Y axis. Use **TRACE_INTERVALS_AUTO** to specify that the dScope should automatically calculate the intervals.

bLog **True** to display the axis logarithmically, **False** to display it linearly.

NB: **If the scale is set to logarithmic, and the Y scale starts at 0, the minimum Y value will be adjusted when displayed to allow it to be shown logarithmically.**

Return value

This method returns **True** if the function is successful, or **False** if it fails. This may be because the number of intervals is invalid.

5.8.3.2.19 TRACE_GetYIntervals

TRACE_GetYIntervals (psNumIntervals, pbLog)

This method retrieves the number of intervals to display on the current Trace's Y axis, and whether it is currently displayed as linear or logarithmic .

Parameters

psNumIntervals	This will return the number of intervals displayed on the Trace's Y axis, or **TRACE_INTERVALS_AUTO** if the axis is set up to calculate intervals automatically.
pbLog	This will return **True** if the axis is currently displayed logarithmically, **False** if it is displayed linearly.

Return value

This method has no return value.

5.8.3.2.20 TRACE_AutoZoomY

TRACE_AutoZoomY ()

This method auto-zooms the Y axis of the current Trace.

When zooming a Trace's Y-axis automatically, dScope will attempt to set the minimum and maximum Y values to such that the full range of Trace Y values is shown on the screen.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.8.3.2.21 TRACE_DefaultZoomY

TRACE_DefaultZoomY ()

This method zooms the Y axis of the current Trace to its default values.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.8.3.2.22 TRACE_SetMinLimit

bRet = TRACE_SetMinLimit (strLimitFile)

This method sets the Lower Limit Line of the current Trace.

Parameters

strLimitFile

The file name of a Limit Line to use as this Trace's lower limit. Any valid file name can be used, enclosed in double quotation marks ("...").

The file name passed can be the name of a limit file (*.lmt), or the name of a script file used to create a Limit Table (*.dss).

If a full path name is not specified, then the system will look in the folder specified in the Options dialogue box for Limit Files (See OPT_LimitFilesFolder).

If necessary, the system will automatically append the file extension for Limit Table files (".lmt").

Return value

This method returns **True** if the limit was set successfully, or **False** if it failed. This may be because the file passed was invalid, or the limit file's units are incompatible with the current Trace's units.

5.8.3.2.23 TRACE_SetMaxLimit

bRet = TRACE_SetMaxLimit (strLimitFile)

This method sets the Upper Limit Line of the current Trace.

Parameters

strLimitFile

The file name of a Limit Line to use as this Trace's upper limit. Any valid file name can be used, enclosed in double quotation marks ("...").

The file name passed can be the name of a limit file (*.lmt), or the name of a script file used to create a Limit Table (*.dss).

If a full path name is not specified, then the system will look in the folder specified in the Options dialogue box for Limit Files (See

OPT_LimitFilesFolder).

If necessary, the system will automatically append the file extension for Limit Table files (".lmt").

Return value

This method returns **True** if the limit was set successfully, or **False** if it failed. This may be because the file passed was invalid, or the limit file's units are incompatible with the current Trace's units.

5.8.3.2.24 TRACE_GetMinLimitLine

sTraceID = TRACE_GetMinLimitLine ()

This method gets the Lower Limit Line of the current Trace.

Parameters

This method has no parameters.

Return value

This method returns the Trace ID of the Lower Limit Line for the current Trace, or **TRACE_NULL_ID** if the current Trace does not have a Lower Limit Line.

NB: **To set the current Trace to the limit line returned, you must use TW_SetCurrentTrace to the trace ID returned from this method.**

5.8.3.2.25 TRACE_GetMaxLimitLine

sTraceID = TRACE_GetMaxLimitLine ()

This method gets the Upper Limit Line of the current Trace.

Parameters

This method has no parameters.

Return value

This method returns the Trace ID of the Upper Limit Line for the current Trace, or **TRACE_NULL_ID** if the current Trace does not have a Upper Limit Line.

NB: **To set the current Trace to the limit line returned, you must use TW_SetCurrentTrace to the trace ID returned from this method.**

5.8.3.2.26 TRACE_GetCursorPos

lPos = TRACE_GetCursorPos ()

This method returns the zero-based position of the Cursor on the current Trace as a zero-based index into the Trace's data.

Parameters

This method has no parameters.

Return value

The cursor position is returned as a long integer value. If the Cursor is off, then -1 is returned.

5.8.3.2.27 TRACE_SetCursorPos

bRet = TRACE_SetCursorPos (lPos)

This method sets the position of the Cursor on the current Trace. If the Cursor is Off, this method will turn it on.

Parameters

lPos	The zero-based index into the Trace's data to set the Cursor position to. For example, if a Scope trace has been captured for an FFT of 4k (4096) points, then the Cursor position can be between 0 and 4095.

Return value

This method returns **True** if the method succeeded, or **False** if it failed. This may be because the position specified is not a valid position.

5.8.3.2.28 TRACE_AddMark

sMark = TRACE_AddMark (lPos, strLabel)

This method adds a Mark, together with a descriptive label, to the current Trace at the specified position.

Parameters

lPos	The zero-based index into the Trace's data to insert a Mark at. For example, if a Scope trace has been captured for an FFT of 4k (4096) points, then the Cursor position can be between 0 and 4095.
strLabel	A string specifying the label to give the mark. This can be any string up to 32 characters long.

Return value

This method returns the ID of the Mark added. This ID must be used when accessing this Mark's label using TRACE_SetMarkLabel or TRACE_GetMarkLabel, or when removing the Mark using TRACE_RemoveMark.

If the Mark could not be added successfully, then -1 is returned. This may happen because the position specified is not a valid position, or because the maximum number of marks for this Trace (20) has been reached.

5.8.3.2.29 TRACE_RemoveMark

bRet = TRACE_RemoveMark (sMark)

This method removes the specified Mark from the current Trace.

Parameters

sMark	The zero-based ID of the Mark to remove. This is the Mark ID returned by the TRACE_AddMark method.

Return value

This method returns **True** if the Mark was removed successfully, or **False** if it failed. This may be because the Mark ID specified is not a valid Mark.

5.8.3.2.30 TRACE_SetMarkLabel

bRet = TRACE_SetMarkLabel (sMark, strLabel)

This method sets the descriptive label of the specified Mark on the current Trace.

Parameters

sMark	The zero-based ID of the Mark whose label is to be set. This is the Mark ID returned by the TRACE_AddMark method.
strLabel	A string specifying the label to give the mark. This can be any string up to 32 characters long.

Return value

This method returns **True** if the Mark label was set successfully, or **False** if it failed. This may be because the Mark ID specified is not a valid Mark.

5.8.3.2.31 TRACE_GetMarkLabel

strLabel = TRACE_GetMarkLabel (sMark)

This method returns the descriptive label of the specified Mark on the current Trace.

Parameters

sMark	The zero-based ID of the Mark whose label is to be returned. This is the Mark ID returned by the TRACE_AddMark method.

Return value

This method returns the label of the specified mark, or "" (an empty string) if it failed. This may be because the Mark ID specified is not a valid Mark.

5.8.3.2.32 TRACE_RemoveAllMarks

TRACE_RemoveAllMarks ()

This method removes all Marks from the current Trace.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.8.4 Limit Table reference

Live Traces on the Trace Window can be given Limit Lines, to detect invalid data on the dScope input. Limit Lines can be created as a direct copy of a Trace's data, or they can take the form of tables of X and Y values that allow a few points to be specified and the data in between to be interpolated.

To create a Limit Table, the user can either draw it onto the screen using the "Edit Limit Line" option on the Trace Window, or can write a simple script to create a set of X and Y values that define a table. The following section is a reference for the creation of Limit Tables using a script.

Creating a Limit Table

A script can create a Limit Table using the following steps:
1) Initialise the table (by giving it a file name)
2) Set up the units that the data will be entered in
3) Set each point in the table.
4) Optionally, save the limit table (See LMT_SaveTable for details).

NB: **A Limit Table can be used as either an upper or a lower limit, for any type of Trace for which its units are valid.**

All Limit Table properties and methods must be prefixed with **"LimitTable."**

NB: **Writing a script to fill in a Limit Table will probably result in two separate files - the script file, and the actual Limit Table file itself.**

Within the Trace Window, these can be used interchangeably - the script file can be loaded in place of the table, and rather than directly copy the file's data into memory, the script will run and fill in the memory.

For example, let's say we write a script ("My Upper Limit.dss") that generates a Limit Table called "My Upper Limit.lmt".
In future sessions, you could use *either* of these two files when specifying a Limit Table for a Trace.

Properties

LMT_XUnit
LMT_YUnit

Methods

LMT_InitTable
LMT_AddPoint
LMT_RemovePoint
LMT_SaveTable

Example

The following example creates a simple limit table for an FFT trace between 0Hz and 24kHz, leaving a 1kHz gap for the peak.

```
' TYPE           Limit table
' DESCRIPTION    Creates a Limit table for an FFT Trace
'                for a 1kHz sine wave at -60dBFS

' *** Declarations ***
Option Explicit       ' Must declare vars before using
Dim strFileName       ' File name of limit table to create


' *** Main body of script ***
Sub dScope_Main

    ' Set up variables
    strFileName = "FFT Limit (dBFS).lmt"

    ' Create the limit table
    If Not LimitTable.LMT_InitTable(strFileName) Then
        MsgBox "Failed to create limit table"
    End If

    ' Set up the units
    LimitTable.LMT_XUnit = UNIT_FREQ_HZ
    LimitTable.LMT_YUnit = UNIT_DBFS

    ' Set up points before the peak...
    ' Note that we don't want a line drawn TO the first
    ' point, but from the first to the second...
    LimitTable.LMT_AddPoint 0.0,   -60.0, False
    LimitTable.LMT_AddPoint 900.0, -60.0, True

    ' ...and we DON'T join the next point up, since we
    ' need To leave a space for the FFT peak.
    LimitTable.LMT_AddPoint 1100.0,  -60.0, False
```

```
    LimitTable.LMT_AddPoint 24000.0, -60.0, True

End Sub ' dScope_Main
```

5.8.4.1 Properties

5.8.4.1.1 LMT_XUnit

Description

This property allows you to specify the unit that the Limit Table's X values are entered in.

Values

The unit entered can be any valid dScope unit, but it must be compatible with the X unit of the Trace that you will be applying this Limit Table to. (i.e. it must be possible to convert values from one unit to the other).

The following units are allowed:

UNIT_MS	Sets the Limit Table's X unit to milliseconds.
UNIT_FREQ_HZ	Sets the Limit Table's X unit to Hz.
UNIT_DBFS	Sets the Limit Table's X unit to dBFS.
UNIT_PERCENTFS	Sets the Limit Table's X unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Limit Table's X unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Limit Table's X unit to Hex.
UNIT_V	Sets the Limit Table's X unit to a voltage.
UNIT_VRMS	Sets the Limit Table's X unit to an RMS voltage.
UNIT_VP	Sets the Limit Table's X unit to a peak voltage.
UNIT_VPP	Sets the Limit Table's X unit to a peak-to-peak voltage.
UNIT_DBU	Sets the Limit Table's X unit to dBu.
UNIT_DBV	Sets the Limit Table's X unit to dBV.
UNIT_DBM	Sets the Limit Table's X unit to dBm.
UNIT_W	Sets the Limit Table's X unit to W.
UNIT_DBSPL	Sets the Limit Table's X unit to dBSPL.
UNIT_DBR	Sets the Limit Table's X unit to dBr (dB, relative to the reference amplitude specified using SA_RefAmpl).
UNIT_PERCENTREF	Sets the Limit Table's X unit to a percentage of the reference amplitude, specified using SA_RefAmpl.
UNIT_RELATIVE_DB	Sets the Limit Table's X unit to dB.
UNIT_RELATIVE_PERCENT	Sets the Limit Table's X unit to percent.
UNIT_JITTER_UI	Sets the Limit Table's X unit to UI.
UNIT_JITTER_NS	Sets the Limit Table's X unit to ns.
UNIT_PHASE_UI	Sets the Limit Table's X unit to a phase unit of UI.
UNIT_PHASE_PERCENT	Sets the Limit Table's X unit to a phase unit of percent.
UNIT_PHASE_SAMPLES	Sets the Limit Table's X unit to be samples.
UNIT_PHASE_DEGREES	Sets the Limit Table's X unit to be degrees.
UNIT_PHASE_RADIANS	Sets the Limit Table's X unit to be radians.
UNIT_PHASE_US	Sets the Limit Table's X unit to be microseconds.
UNIT_PPM	Sets the Limit Table's X unit to be ppm (parts per million).

5.8.4.1.2 LMT_YUnit

Description

This property allows you to specify the unit that the Limit Table's Y values are entered in.

Values

The unit entered can be any valid dScope unit, but it must be compatible with the Y unit of the Trace that you will be applying this Limit Table to. (i.e. it must be possible to convert values from one unit to the other).

The following units are allowed:

UNIT_MS	Sets the Limit Table's Y unit to milliseconds.
UNIT_FREQ_HZ	Sets the Limit Table's Y unit to Hz.
UNIT_DBFS	Sets the Limit Table's Y unit to dBFS.
UNIT_PERCENTFS	Sets the Limit Table's Y unit to %FS (percentage of full scale).
UNIT_FFS	Sets the Limit Table's Y unit to FFS (fraction of full scale).
UNIT_HEX	Sets the Limit Table's Y unit to Hex.
UNIT_V	Sets the Limit Table's Y unit to a voltage.
UNIT_VRMS	Sets the Limit Table's Y unit to an RMS voltage.
UNIT_VP	Sets the Limit Table's Y unit to a peak voltage.
UNIT_VPP	Sets the Limit Table's Y unit to a peak-to-peak voltage.
UNIT_DBU	Sets the Limit Table's Y unit to dBu.
UNIT_DBV	Sets the Limit Table's Y unit to dBV.
UNIT_DBM	Sets the Limit Table's Y unit to dBm.
UNIT_W	Sets the Limit Table's Y unit to W.
UNIT_DBSPL	Sets the Limit Table's Y unit to dBSPL.
UNIT_DBR	Sets the Limit Table's Y unit to dBr (dB, relative to the reference amplitude specified using SA_RefAmpl).
UNIT_PERCENTREF	Sets the Limit Table's Y unit to a percentage of the reference amplitude, specified using SA_RefAmpl.
UNIT_RELATIVE_DB	Sets the Limit Table's Y unit to dB.
UNIT_RELATIVE_PERCENT	Sets the Limit Table's Y unit to percent.
UNIT_JITTER_UI	Sets the Limit Table's Y unit to UI.
UNIT_JITTER_NS	Sets the Limit Table's Y unit to ns.
UNIT_PHASE_UI	Sets the Limit Table's Y unit to a phase unit of UI.
UNIT_PHASE_PERCENT	Sets the Limit Table's Y unit to a phase unit of percent.
UNIT_PHASE_SAMPLES	Sets the Limit Table's Y unit to be samples.
UNIT_PHASE_DEGREES	Sets the Limit Table's Y unit to be degrees.
UNIT_PHASE_RADIANS	Sets the Limit Table's Y unit to be radians.
UNIT_PHASE_US	Sets the Limit Table's Y unit to be microseconds.
UNIT_PPM	Sets the Limit Table's Y unit to be ppm (parts per million).

5.8.4.2 Methods

5.8.4.2.1 LMT_InitTable

bRetVal = LMT_InitTable (strFileName)

This method initialises a Limit Table, ready to write points into.

NB: **This method must be called first when creating a Trace Limit Table.**

Parameters

strFileName The name of the table file that should be created. Any valid file name can be used, enclosed in double quotation marks ("...").

If a full path name is not specified, then the system will create a file in the folder specified in the Options dialogue box for Limit Files (See OPT_LimitFilesFolder).

If necessary, the system will automatically append the file extension for Limit Table files (".lmt").

Return value

This method returns **True** if the table initialization completed successfully, or **False** if it failed for some reason.

5.8.4.2.2 LMT_AddPoint

bRetVal = LMT_AddPoint (dXValue, dYValue, bLineTo)

This method adds a point to the Limit Table.

Parameters

dXValue The X value of the point to write to the Limit Table.
dYValue The Y value of the point to write to the Limit Table.
bLineTo Whether to draw a line to this point. This parameter should *always* be **False** for the first point in a Limit Table, since a line cannot be drawn to the first point. Further points in the table will usually have this parameter set to **True**; however if there are any gaps in the Limit Line then the first point after the gap will have this parameter set to **False**.

Return value

This method returns **True** if the point was added correctly, or **False** if it failed. This may be because the value itself was invalid, or the position is not within the size of the data table (as specified using USR_InitTable)

NB: **This method will be ignored unless the Limit Table has been initialised using the LMT_InitTable method.**

5.8.4.2.3 LMT_RemovePoint

bRetVal = LMT_RemovePoint (dXValue)

This method removes the point with the given X value from the Limit Table.

Parameters

dXValue	The X value of the point to remove from the Limit Table.

Return value

This method returns **True** if the point was removed correctly, or **False** if it failed. This may be because the specified X value does not exist in the Limit Table.

NB: **This method will be ignored unless the Limit Table has been initialised using the LMT_InitTable method.**

5.8.4.2.4 LMT_SaveTable

bRetVal = LMT_SaveTable (strFileName)

This method saves a limit table to the specified file name.

The LMT_SaveTable method is only necessary if the Limit Table is being created in a script that is not a Limit Table script (see Types of dScope script) or when more than one table is created in a single script. Usually, a Limit Table is created from a Limit Table script; in this case, the table is saved automatically when the script finishes running. However, there may be occasions when it is necessary to create Limit Tables from a different script type; for example an Automation script. In this case, the same code is used to initialise the table (LMT_InitTable) and add/remove points (LMT_AddPoint and LMT_RemovePoint); However this LMT_SaveTable method must then be used to actually save the script.

Similarly, if a Limit Table script creates more than one table, only the last one created will be saved automatically when the script has finished running. To save any other Limit Tables created by the script, you must explicitly call the LMT_SaveTable method.

NB: **If this method is used from a Limit Table script, with the same filename passed to LMT_InitTable, then the table will no longer be saved automatically when the script finishes running.**

Parameters

strFileName The name of the table file that should be created. Any valid file name can be used, enclosed in double quotation marks ("...").

If a full path name is not specified, then the system will create a file in the folder specified in the Options dialogue box for Limit Files (See OPT_LimitFilesFolder).

If necessary, the system will automatically append the file extension for Limit Table files (".lmt").

Return value

This method returns **True** if the table was saved successfully, or **False** if it failed for some reason. This may be because the dScope cannot open the file with the name specified as a parameter to LMT_InitTable.

NB: **This method will be ignored unless the Limit Table has been initialised using the LMT_InitTable method.**

5.9 Readings

The Readings section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"Reading."**

To access the properties and methods of a Reading, you must first select the Reading that you wish to access using one of the following methods:

GetFirstReadingForResult
GetNextReadingForResult
SetCurrentReadingFromEventParam

For Readings created from FFT Detectors, the following methods should be used:

GetFirstFFTDetReading
GetNextFFTDetReading
SetCurrentReadingFromEventParam

For further information on accessing Readings, see Accessing Readings below.

Properties

RDG_Value
RDG_Description
RDG_ResolutionType
RDG_Resolution
RDG_ShowResultValue
RDG_FollowUnits
RDG_Unit
RDG_ShowUnit
RDG_Channel
RDG_ShowBarGraph
RDG_BarMinValue
RDG_BarMaxValue
RDG_BarNumSegments
RDG_LimitCheckingOn
RDG_MinLimit
RDG_MaxLimit
RDG_AlwaysDisplayLimitStatus
RDG_LimitAudibleAlarm
RDG_LimitChangeTextColour
RDG_LimitChangeBackgroundColour
RDG_LimitEventLog
RDG_MinLimitBreached
RDG_MaxLimitBreached
RDG_LastMinLimitBreachValue
RDG_LastMaxLimitBreachValue

Methods

Accessing Readings

The way that Reading automation works is that you must select a Reading before accessing its properties. Once you have selected a Reading, all properties and methods of a Reading will then access that selected Reading.

To access a Reading, you must specify which Result it was created from. If more than one Reading has been created from a single Result (for example, showing the same Result in different units), you can cycle through all the Readings until you find the one you want.

The GetFirstReadingForResult and GetNextReadingForResult methods can be used to cycle through the Readings. Usually, the GetFirstReadingForResult method will be sufficient; if it succeeds in finding a Reading, it will return **True** and from that point, all properties and methods of the Reading object will act on that Reading. Further calls to GetNextReadingForResult will go on cycling through the Readings; each time, if a Reading is found, it will set this as the current Reading and return **True**.

As an example, the following code snippet accesses the Reading created from channel A of the Continuous-Time Detector, and changes its unit to %.

```
If GetFirstReadingForResult(RESULT_CTD_CHA) Then
   Reading.RDG_Unit = UNIT_RELATIVE_PERCENT
End If
```

And the following example will change the text colour of all Readings created from the Signal Analyzer's channel A frequency:

```
If GetFirstReadingForResult(RESULT_SA_FREQ_CHA) Then
   Reading.RDG_SetTextColour 255, 255, 255
   While GetNextReadingForResult(RESULT_SA_FREQ_CHA)
      Reading.RDG_SetTextColour 255, 255, 255
   Wend
End If
```

NB:

Another way of accessing a Reading is from within an Event subroutine, fired when a Reading's limit is breached. For further details, see SetCurrentReadingFromEventParam, Event_ReadingMinLimit or Event_ReadingMaxLimit.

5.9.1 GetFirstReadingForResult

bRet = GetFirstReadingForResult (sResultID)

This method gets the first Reading created from the given Result, and if successful, sets it to be the current Reading for Reading automation.

For full details on how to access Readings, see Accessing Readings.

Parameters

sResultID	The Result ID to access the first Reading for. See Result IDs below for a list of allowed Result IDs.

Return value

This method returns **True** if it successfully found a Reading, or **False** otherwise.

NB: **If this method succeeds, it automatically sets the current Reading to the one found.**

Result IDs

The following Result IDs can be used for the ***sResultID*** parameter:

RESULT_DO_REFSYNCFRAMERATE	Searches for a Reading created from the Ref Sync source frame rate.
RESULT_DO_REFSYNCFRAMERATEDEVIATION	Searches for a Reading created from the Ref Sync source frame rate deviation.
RESULT_DI_FRAMERATE	Searches for a Reading created from the Digital Input frame rate.
RESULT_DI_FRAMERATEDEVIATION	Searches for a Reading created from the Digital Input frame rate deviation.
RESULT_DIC_AMPL	Searches for a Reading created from the Digital Input Carrier amplitude.
RESULT_DIC_JITTERAMPL	Searches for a Reading created from the Digital Input Carrier jitter amplitude.
RESULT_DIC_PHASE	Searches for a Reading created from the Digital Input Carrier phase w.r.t. ref.
RESULT_SA_RMSAMPL_CHA	Searches for a Reading created from the RMS amplitude of channel A of the Signal Analyzer.
RESULT_SA_RMSAMPL_CHB	Searches for a Reading created from the RMS amplitude of channel B of the Signal Analyzer.
RESULT_SA_RMSAMPL_SEL	Searches for a Reading created from the RMS amplitude of the selected channel of the Signal Analyzer.
RESULT_SA_RMSAMPL_NONSEL	Searches for a Reading created from the RMS amplitude of the non-selected channel of the Signal Analyzer.
RESULT_SA_FREQ_CHA	Searches for a Reading created from the frequency of channel A of the Signal Analyzer.
RESULT_SA_FREQ_CHB	Searches for a Reading created from the frequency of channel B of the Signal Analyzer.
RESULT_SA_FREQ_SEL	Searches for a Reading created from the frequency of the selected channel of the Signal

	Analyzer.
RESULT_SA_FREQ_NONSEL	Searches for a Reading created from the frequency of the non-selected channel of the Signal Analyzer.
RESULT_SA_PHASE	Searches for a Reading created from the inter-channel phase of the Signal Analyzer.
RESULT_CTD_CHA	Searches for a Reading created from channel A of the Continuous-Time Detector.
RESULT_CTD_CHB	Searches for a Reading created from channel B of the Continuous-Time Detector.
RESULT_CTD_SEL	Searches for a Reading created from the selected channel of the Continuous-Time Detector.
RESULT_CTD_NONSEL	Searches for a Reading created from the non-selected channel of the Continuous-Time Detector.

NB: **To get a Reading for a Result created from an FFT Detector, see GetFirstFFTDetReading and GetNextFFTDetReading.**

5.9.2 GetNextReadingForResult

bRet = GetNextReadingForResult (sResultID)

This method gets a subsequent Reading created from the given Result (after a call to GetFirstReadingForResult), and if successful, sets it to be the current Reading for Reading automation.

For full details on how to access Readings, see Accessing Readings.

Parameters

sResultID	The Result ID to access the Reading for. See Result IDs below for a list of allowed Result IDs.

Return value

This method returns **True** if it successfully found a Reading, or **False** otherwise.

NB: **If this method succeeds, it automatically sets the current Reading to the one found.**

Result IDs

The following Result IDs can be used for the ***sResultID*** parameter:

RESULT_DO_REFSYNCFRAMERATE	Searches for a Reading created from the Ref Sync source frame rate.
RESULT_DO_REFSYNCFRAMERATEDEVIATION	Searches for a Reading created from the Ref Sync source frame rate deviation.
RESULT_DI_FRAMERATE	Searches for a Reading created from the Digital Input frame rate.
RESULT_DI_FRAMERATEDEVIATION	Searches for a Reading created from the Digital

	Input frame rate deviation.
RESULT_DIC_AMPL	Searches for a Reading created from the Digital Input Carrier amplitude.
RESULT_DIC_JITTERAMPL	Searches for a Reading created from the Digital Input Carrier jitter amplitude.
RESULT_DIC_PHASE	Searches for a Reading created from the Digital Input Carrier phase w.r.t. ref.
RESULT_SA_RMSAMPL_CHA	Searches for a Reading created from the RMS amplitude of channel A of the Signal Analyzer.
RESULT_SA_RMSAMPL_CHB	Searches for a Reading created from the RMS amplitude of channel B of the Signal Analyzer.
RESULT_SA_RMSAMPL_SEL	Searches for a Reading created from the RMS amplitude of the selected channel of the Signal Analyzer.
RESULT_SA_RMSAMPL_NONSEL	Searches for a Reading created from the RMS amplitude of the non-selected channel of the Signal Analyzer.
RESULT_SA_FREQ_CHA	Searches for a Reading created from the frequency of channel A of the Signal Analyzer.
RESULT_SA_FREQ_CHB	Searches for a Reading created from the frequency of channel B of the Signal Analyzer.
RESULT_SA_FREQ_SEL	Searches for a Reading created from the frequency of the selected channel of the Signal Analyzer.
RESULT_SA_FREQ_NONSEL	Searches for a Reading created from the frequency of the non-selected channel of the Signal Analyzer.
RESULT_SA_PHASE	Searches for a Reading created from the inter-channel phase of the Signal Analyzer.
RESULT_CTD_CHA	Searches for a Reading created from channel A of the Continuous-Time Detector.
RESULT_CTD_CHB	Searches for a Reading created from channel B of the Continuous-Time Detector.
RESULT_CTD_SEL	Searches for a Reading created from the selected channel of the Continuous-Time Detector.
RESULT_CTD_NONSEL	Searches for a Reading created from the non-selected channel of the Continuous-Time Detector.

NB: **To get a Reading for a Result created from an FFT Detector, see GetFirstFFTDetReading and GetNextFFTDetReading.**

5.9.3 GetFirstFFTDetReading

bRet = GetFirstFFTDetReading (sResultID, sDetectorID)

This method gets the first Reading created from the given FFT Detector Result, and if successful, sets it to be the current Reading for Reading automation.

For full details on how to access Readings, see Accessing Readings.

Parameters

sResultID	The FFT Detector Result to access the first Reading for. See Result IDs below for a list of allowed Result IDs.
sDetectorID	The FFT Detector ID that the Reading was created from. This can be the ID of any current FFT Detector.

Return value

This method returns **True** if it successfully found a Reading, or **False** otherwise.

NB: **If this method succeeds, it automatically sets the current Reading to the one found.**

Result IDs

The following Result IDs can be used for the ***sResultID*** parameter:

RESULT_FFTD_CHA	Searches for a Reading created from channel A of the specified FFT Detector.
RESULT_FFTD_CHB	Searches for a Reading created from channel B of the specified FFT Detector.
RESULT_FFTD_SEL	Searches for a Reading created from the selected channel of the specified FFT Detector.
RESULT_FFTD_NONSEL	Searches for a Reading created from the non-selected channel of the specified FFT Detector.

5.9.4 GetNextFFTDetReading

bRet = GetNextFFTDetReading (sResultID, sDetectorID)

This method gets a subsequent Reading created from the given FFT Detector Result (after a call to GetFirstFFTDetReading), and if successful, sets it to be the current Reading for Reading automation.

For full details on how to access Readings, see Accessing Readings.

Parameters

sResultID	The FFT Detector Result to access the Reading for. See Result IDs below for a list of allowed Result IDs.
sDetectorID	The FFT Detector ID that the Reading was created from. This can be the ID of any current FFT Detector.

Return value

This method returns **True** if it successfully found a Reading, or **False** otherwise.

NB: **If this method succeeds, it automatically sets the current Reading to the one found.**

Result IDs

The following Result IDs can be used for the ***sResultID*** parameter:

RESULT_FFTD_CHA	Searches for a Reading created from channel A of the specified FFT Detector.
RESULT_FFTD_CHB	Searches for a Reading created from channel B of the specified FFT Detector.
RESULT_FFTD_SEL	Searches for a Reading created from the selected channel of the specified FFT Detector.
RESULT_FFTD_NONSEL	Searches for a Reading created from the non-selected channel of the specified FFT Detector.

5.9.5 SetCurrentReadingFromEventParam

bRet = SetCurrentReadingFromEventParam (lParam)

The dScope Event Manager can be set up to fire an event to a script when a Reading's upper or lower limit is breached. When it does so, the event subroutine is passed a parameter that represents the Reading whose limit was breached.

This method can be used from within one of these event subroutines to set the current Reading from the parameter passed. All further actions carried out on a Reading will affect the Reading that has been set using this method.

NB: **This method should ONLY be called from within either the Event_ReadingMinLimit or the Event_ReadingMaxLimit subroutine.**

Parameters

lParam	The parameter passed to the event subroutine, which should be passed unchanged to this method.

NB: **The lParam has no meaning to the script, other than as a parameter to pass to this method.**

Return value

This method returns **True** if the current Reading was successfully set, or **False** if the call failed (for example, if the lParam is invalid).

5.9.6 Properties

5.9.6.1 RDG_Value

Description

This **read-only** property represents the Result value of the Reading. If the Reading is set to follow the Result field's units (see RDG_FollowUnit), it will be in the Result's current unit. Otherwise, it will be in the unit specified by RDG_Unit.

Values

The Reading value is represented as a double-precision floating point value.

5.9.6.2 RDG_Description

Description

This property can be used to set or read the Reading's description.

Values

Any valid string can be entered.

5.9.6.3 RDG_ResolutionType

Description

This property specifies whether the resolution of the Reading (RDG_Resolution) is expressed in significant figures, or decimal places.

Values

RDG_RESOLUTION_DPS	The Reading should be displayed using RDG_Resolution to specify the number of decimal places.
RDG_RESOLUTION_SIGFIGS	The Reading should be displayed using RDG_Resolution to specify the number of significant figures.

5.9.6.4 RDG_Resolution

Description

This property specifies the number of decimal places or significant figures (as defined by RDG_ResolutionType) that the Reading's value should be displayed in.

Values

The Reading's resolution is a short integer value. It can be any number between 0 and 10 decimal places, or 1 and 10 significant figures.

5.9.6.5 RDG_ShowResultValue

Description

This property is used to select whether to show the Result value in the Reading.

Values

True	Show the Result value in the Reading.
False	Do not show the Result value in the Reading.

NB: **Either the Result, the min and max values (RDG_ShowMinAndMax) or the bar graph (RDG_ShowBarGraph) *must* be shown in a Reading.**

5.9.6.6 RDG_FollowUnit

Description

This property is used to select whether the Reading should follow the unit selected for the Result from which it was created. If the Reading is not following the Result's unit, it will be set as specified by the RDG_Unit property.

Values

True	The Reading should follow the unit selected for the Result from which it was created.
False	The Reading's unit should be as specified by the RDG_Unit property.

5.9.6.7 RDG_Unit

Description

This property is used to select the unit for the Reading.

NB: **This property is ignored if the Reading is set to follow the unit specified for the Result from which it was created (see RDG_FollowUnit).**

Values

The allowed values for the Sweep source unit depend on the Result from which this Reading was created.

If the Result is a frequency Result (**RESULT_DO_REFSYNCFRAMERATE**, **RESULT_DI_FRAMERATE**, **RESULT_SA_FREQ_CHA**, **RESULT_SA_FREQ_CHB**, **RESULT_SA_FREQ_SEL** or **RESULT_SA_FREQ_NONSEL**) then the only valid unit is **UNIT_FREQ_HZ**.

If the Result is **RESULT_DO_REFSYNCFRAMERATEDEVIATION** or **RESULT_DI_FRAMERATEDEVIATION**, then the only valid unit is **UNIT_PPM**.

If the Result is **RESULT_DIC_AMPL**, then the only valid unit is **UNIT_V**.

If the Result is **RESULT_DIC_JITTERAMPL**, then valid units are **UNIT_JITTER_NS** and **UNIT_JITTER_UI**.

If the Result is **RESULT_DIC_PHASE**, then valid units are **UNIT_PHASE_PERCENT**, **UNIT_PHASE_DEGREES** and **UNIT_PHASE_UI**.

If the Result is **RESULT_SA_PHASE**, then valid units are **UNIT_PHASE_DEGREES**, **UNIT_PHASE_RADIANS**, **UNIT_PHASE_US** and **UNIT_PHASE_SAMPLES**.

If the Result is **RESULT_SA_RMSAMPL_CHA**, **RESULT_SA_RMSAMPL_CHB**, **RESULT_SA_RMSAMPL_SEL**, **RESULT_SA_RMSAMPL_NONSEL**, **RESULT_CTD_CHA**, **RESULT_CTD_CHB**, **RESULT_CTD_SEL**, **RESULT_CTD_NONSEL**, **RESULT_FFTD_CHA**, **RESULT_FFTD_CHB**, **RESULT_FFTD_SEL**, or **RESULT_FFTD_NONSEL** then the allowed unit depends on whether the dScope analyzer is in jitter demodulation mode (see AI_Source). If so, then **UNIT_JITTER_NS** and **UNIT_JITTER_UI** are valid; otherwise, any amplitude unit is valid (**UNIT_DBFS, UNIT_PERCENTFS, UNIT_FFS, UNIT_HEX, UNIT_VRMS, UNIT_VP, UNIT_VP, UNIT_DBU, UNIT_DBV, UNIT_DBM, UNIT_W, UNIT_DBSPL**). Depending on the relativity of the Result, **UNIT_RELATIVE_DB** or **UNIT_RELATIVE_PERCENT** may be valid.

5.9.6.8 RDG_ShowUnit

Description

This property is used to select whether the Reading display should show the unit after the numerical Result (e.g "-14.32 dBFS" rather than just "-14.32")

Values

True	The Reading's display should contain the unit.
False	The Reading's display should contain the numerical Result only, and no unit.

5.9.6.9 RDG_Channel

Description

This property is used to select the channel for the Reading. It can be set to show either channel A or B, or can be set to follow the global channel selection (see SA_Channel) using the selected or non-selected channel.

If the global channel selection is set to both channels, then **CHANNEL_SEL** will set the channel to channel A, and **CHANNEL_NONSEL** will set the channel to channel B.

Values

CHANNEL_A	The Reading should always show the Result from channel A.
CHANNEL_B	The Reading should always show the Result from channel B.
CHANNEL_SEL	The Reading should always show the Result from the channel specified by the global channel selection.
CHANNEL_NONSEL	The Reading should always show the Result from the channel not specified by the global channel selection.

NB:

This property is ignored unless the Reading is from a Result that has an equivalent on the other channel.
For example, RESULT_SA_FREQ_CHA has the equivalent RESULT_SA_FREQ_CHB for the other channel; however RESULT_DIC_AMPL is a solitary Reading where the channel has no meaning.

5.9.6.10 RDG_ShowBarGraph

Description

This property is used to select whether to show a bar graph in the Reading.

Values

True	Show the bar graph in the Reading.
False	Do not show the bar graph in the Reading.

NB: Either the Result (RDG_ShowResultValue), the min and max values (RDG_ShowMinAndMax) or the bar graph *must* be shown in a Reading.

5.9.6.11 RDG_BarMinValue

Description

This property specifies the minimum value for the Reading's bar graph.

Values

The minimum bar graph value is represented as a double-precision floating point value. It should be a valid value for the unit of the Reading (either the unit for the Result from which the Reading was created, or the unit specified using RDG_Unit).

NB: This property is ignored unless the Reading is set up to display a bar graph (RDG_ShowBarGraph).

5.9.6.12 RDG_BarMaxValue

Description

This property specifies the maximum value for the Reading's bar graph.

Values

The maximum bar graph value is represented as a double-precision floating point value. It should be a valid value for the unit of the Reading (either the unit for the Result from which the Reading was created, or the unit specified using RDG_Unit).

NB: This property is ignored unless the Reading is set up to display a bar graph (RDG_ShowBarGraph).

5.9.6.13 RDG_BarNumSegments

Description

This property specifies the number of segments to split the Reading's bar graph into.

Values

The number of bar graph segments is a short integer value. It can be any number between 1 and 100.

NB: This property is ignored unless the Reading is set up to display a bar graph (RDG_ShowBarGraph).

5.9.6.14 RDG_LimitCheckingOn

Description

This property is used to select whether to turn on limit checking for this Reading.

Values

True	Turn on limit checking for this Reading.
False	Turn off limit checking for this Reading.

5.9.6.15 RDG_MinLimit

Description

This property specifies the minimum limit value for the Reading.

Values

The minimum limit value is represented as a double-precision floating point value. It should be a valid value for the unit of the Reading (either the unit for the Result from which the Reading was created, or the unit specified using RDG_Unit).

NB: This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).

5.9.6.16 RDG_MaxLimit

Description

This property specifies the maximum limit value for the Reading.

Values

The maximum limit value is represented as a double-precision floating point value. It should be a valid value for the unit of the Reading (either the unit for the Result from which the Reading was created, or the unit specified using RDG_Unit).

NB: This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).

5.9.6.17 RDG_AlwaysDisplayLimitStatus

Description

This property is used to select whether to always show the limit status (**high**, **low** or **OK**) on this Reading's display.

Values

True	Always display the limit status on this Reading.
False	Only display the limit status on this Reading when either the lower or upper limit are breached.

NB: **This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).**

5.9.6.18 RDG_LimitAudibleAlarm

Description

This property is used to select whether an audible alarm (beep) should sound when this Reading's lower or upper limit are breached.

Values

True	Sound an audible alarm when either of this Reading's limits are breached.
False	Do not sound an audible alarm when either of this Reading's limits are breached.

NB: **This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).**

5.9.6.19 RDG_LimitChangeTextColour

Description

This property is used to select whether to change the colour of the Reading's text when its lower or upper limit are breached.

To specify the colour to change to, use the RDG_SetLimitTextColour method.

Values

True	Change the text colour when a limit is breached.
False	Do not change the text colour when a limit is breached.

NB: **This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).**

5.9.6.20 RDG_LimitChangeBackgroundColour

Description

This property is used to select whether to change the colour of the Reading's background when its lower or upper limit are breached.

To specify the colour to change to, use the RDG_SetLimitBackgroundColour method.

Values

True	Change the background colour when a limit is breached.
False	Do not change the background colour when a limit is breached.

NB: **This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).**

5.9.6.21 RDG_LimitEventLog

Description

This property is used to select whether to write to the event log file when this Reading's lower or upper limit are breached.

The event log file to write to is specified in the Event Manager using EM_LogFile.

Values

True	Write to the event log file when a limit is breached.
False	Do not write to the event log file when a limit is breached.

NB: **This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).**

5.9.6.22 RDG_MinLimitBreached

Description

This property is used to detect whether a Reading's lower limit has been breached.

This property can also be set to **False**; this is useful to initialise the property so you can be sure that a breach has only occurred since you started the test.

This flag is "sticky", i.e. once set, it remains set so even if a single limit breached occurs over a period of time, this property is still set to True.

Values

True	The Reading's minimum limit has been breached.
False	The Reading's minimum limit has not been breached.

NB: **This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).**

Example

This code example takes Readings created from the Continuous-Time Detector, and checks for five seconds to see whether their limits were breached in that time.

```
bChABreach = False
bChBBreach = False

' *** Initialise breach flags... ***
If GetFirstReadingForResult(RESULT_CTD_CHA) Then
    Reading.RDG_MinLimitBreached = False
End If
If GetFirstReadingForResult(RESULT_CTD_CHB) Then
    Reading.RDG_MinLimitBreached = False
End If

' Wait for a while
Sleep(5000)

If GetFirstReadingForResult(RESULT_CTD_CHA) Then
    bChABreach = Reading.RDG_MinLimitBreached
End If
If GetFirstReadingForResult(RESULT_CTD_CHB) Then
    bChBBreach = Reading.RDG_MinLimitBreached
End If

If bChABreach Or bChBBreach Then
    MsgBox "Failed!"
Else
    MsgBox "Passed!"
End If
```

5.9.6.23 RDG_MaxLimitBreached

Description

This property is used to detect whether a Reading's upper limit has been breached.

This property can also be set to **False**; this is useful to initialise the property so you can be sure that a breach has only occurred since you started the test.

This flag is "sticky", i.e. once set, it remains set so even if a single limit breached occurs over a period of time, this property is still set to True.

Values

True	The Reading's maximum limit has been breached.
False	The Reading's maximum limit has not been breached.

NB: **This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).**

Example

This code example takes Readings created from the Continuous-Time Detector, and checks for five seconds to see whether their limits were breached in that time.

```
bChABreach = False
bChBBreach = False

' *** Initialise breach flags... ***
If GetFirstReadingForResult(RESULT_CTD_CHA) Then
    Reading.RDG_MaxLimitBreached = False
End If
If GetFirstReadingForResult(RESULT_CTD_CHB) Then
    Reading.RDG_MaxLimitBreached = False
End If

' Wait for a while
Sleep(5000)

If GetFirstReadingForResult(RESULT_CTD_CHA) Then
    bChABreach = Reading.RDG_MaxLimitBreached
End If
If GetFirstReadingForResult(RESULT_CTD_CHB) Then
    bChBBreach = Reading.RDG_MaxLimitBreached
End If

If bChABreach Or bChBBreach Then
    MsgBox "Failed!"
Else
    MsgBox "Passed!"
End If
```

5.9.6.24 RDG_LastMinLimitBreachValue

Description

This **read-only** property represents the value of this Reading the last time it breached its lower limit value. If the Reading is set to follow the Result field's units (see RDG_FollowUnit), it will be in the Result's current unit. Otherwise, it will be in the unit specified by RDG_Unit.

NB: **This property should only be checked after the Reading's lower limit has been breached (i.e. RDG_MinLimitBreached is True). If a Reading's lower limit has never been breached, then this property will simply return the current Reading value.**

Values

The last Reading value that breached the minimum limit is represented as a double-precision floating point value.

5.9.6.25 RDG_LastMaxLimitBreachValue

Description

This **read-only** property represents the value of this Reading the last time it breached its upper limit value. If the Reading is set to follow the Result field's units (see RDG_FollowUnit), it will be in the Result's current unit. Otherwise, it will be in the unit specified by RDG_Unit.

NB: **This property should only be checked after the Reading's upper limit has been breached (i.e. RDG_MaxLimitBreached is True). If a Reading's upper limit has never been breached, then this property will simply return the current Reading value.**

Values

The last Reading value that breached the upper limit is represented as a double-precision floating point value.

5.9.6.26 RDG_ShowMinAndMaxValues

Description

This property is used to select whether to show the minimum and maximum values that this Reading has reached since the last time they were reset (see RDG_ResetMinAndMax).

Values

True	Show the minimum and maximum values reached in the Reading.
False	Do not show the minimum and maximum values reached in the Reading.

NB: **Either the Result (RDG_ShowResultValue), the min and max values (RDG_ShowMinAndMax) or the bar graph (RDG_ShowBarGraph) *must* be shown in a Reading.**

5.9.6.27 RDG_MinValue

Description

This **read-only** property represents the minimum value that this Reading has reached since the last time the Reading's minimum and maximum values were reset (see RDG_ResetMinAndMax). It is returned in the unit specified by RDG_Unit.

Values

The minimum value is represented as a double-precision floating point value.

5.9.6.28 RDG_MaxValue

Description

This **read-only** property represents the maximum value that this Reading has reached since the last time the Reading's minimum and maximum values were reset (see RDG_ResetMinAndMax). It is returned in the unit specified by RDG_Unit.

Values

The maximum value is represented as a double-precision floating point value.

5.9.6.29 RDG_ShowMinAndMaxOnBarGraph

Description

This property is used to select whether to show the minimum and maximum values that this Reading has reached on the Reading's bar graph.

Values

True	Show the minimum and maximum values reached in the Reading on the bar graph.
False	Do not show the minimum and maximum values reached in the Reading on the bar graph.

NB: **This property is ignored unless the Reading is set up to display a bar graph (RDG_ShowBarGraph).**

5.9.6.30 RDG_ShowLimitsOnBarGraph

Description

This property is used to select whether to show the current limit values for this Reading on the Reading's bar graph.

Values

True	Show the Reading's current limit values on the bar graph.
False	Do not show the Reading's current limit values on the bar graph.

NB: **This property is ignored unless limit checking has been turned on for this Reading (see RDG_LimitCheckingOn).**

5.9.7 Methods

5.9.7.1 RDG_SetTextColour

RDG_SetTextColour (sRed, sGreen, sBlue)

This method sets the colour of the text in the Reading.

Parameters

sRed	Specifies the red component of the colour, from 0 to 255.
sBlue	Specifies the blue component of the colour, from 0 to 255.
sGreen	Specifies the green component of the colour, from 0 to 255.

Clicking on the [Text colour...] button on the Reading's Properties window will bring up a dialogue box which will allow you to find the red, green and blue components of common colours.

Return value

This method has no return value.

5.9.7.2 RDG_SetBackgroundColour

RDG_SetBackgroundColour (sRed, sGreen, sBlue)

This method sets the colour of the background of the Reading.

Parameters

sRed	Specifies the red component of the colour, from 0 to 255.
sBlue	Specifies the blue component of the colour, from 0 to 255.
sGreen	Specifies the green component of the colour, from 0 to 255.

Clicking on the [Background colour...] button on the Reading's Properties window will bring up a dialogue box which will allow you to find the red, green and blue components of common colours.

Return value

This method has no return value.

5.9.7.3 RDG_SetBarColour

RDG_SetBarColour (sRed, sGreen, sBlue)

This method sets the colour of the bar graph in the Reading.

Parameters

sRed	Specifies the red component of the colour, from 0 to 255.
sBlue	Specifies the blue component of the colour, from 0 to 255.
sGreen	Specifies the green component of the colour, from 0 to 255.

Clicking on the [Bar colour...] button on the Reading's Properties window will bring up a dialogue box which will allow you to find the red, green and blue components of common colours.

Return value

This method has no return value.

NB: **This property is ignored unless the Reading is set up to display a bar graph (RDG_ShowBarGraph).**

5.9.7.4 RDG_SetLimitTextColour

RDG_SetLimitTextColour (sRed, sGreen, sBlue)

This method sets the colour that the Reading's text should change to if the Reading's limit is breached.

Parameters

sRed	Specifies the red component of the colour, from 0 to 255.
sBlue	Specifies the blue component of the colour, from 0 to 255.
sGreen	Specifies the green component of the colour, from 0 to 255.

Clicking on the [Text colour...] button on the Reading's Properties window will bring up a dialogue box which will allow you to find the red, green and blue components of common colours.

Return value

This method has no return value.

NB: **This property is ignored unless limit checking is turned on for this Reading (see RDG_LimitCheckingOn), and the text colour is set to change when a limit is breached (RDG_LimitChangeTextColour).**

5.9.7.5 RDG_SetLimitBackgroundColour

RDG_SetLimitBackgroundColour (sRed, sGreen, sBlue)

This method sets the colour that the Reading's background should change to if the Reading's limit is breached.

Parameters

sRed	Specifies the red component of the colour, from 0 to 255.
sBlue	Specifies the blue component of the colour, from 0 to 255.
sGreen	Specifies the green component of the colour, from 0 to 255.

Clicking on the [Background colour...] button on the Reading's Properties window will bring up a dialogue box which will allow you to find the red, green and blue components of common colours.

Return value

This method has no return value.

NB: **This property is ignored unless limit checking is turned on for this Reading (see RDG_LimitCheckingOn), and the background colour is set to change when a limit is breached (RDG_LimitChangeBackgroundColour).**

5.9.7.6 RDG_ResetMinAndMaxValues

RDG_ResetMinAndMaxValues ()

This method resets the Reading's minimum and maximum values (See RDG_MinValue and RDG_MaxValue) to the current value in the Reading.

Parameters

This method has no parameters.

Return value

This method has no return value.

5.10 Options

The Options section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"Options."**

NB: **All Options details are stored in the PC's registry, and so are persistent between dScope sessions even if no Configuration is saved.**

Properties

OPT_RecentFileList
OPT_StartupConfiguration
OPT_StartupScript
OPT_ConfigurationsFolder
OPT_ScriptsFolder
OPT_LimitFilesFolder
OPT_WavetablesFolder
OPT_DataTablesFolder
OPT_TracesFolder
OPT_FFTWindowsFolder
OPT_WeightingFiltersFolder
OPT_EventLogsFolder
OPT_GraphExportsFolder
OPT_SampleBuffersFolder
OPT_UseCurrentFilesFolder
OPT_LockDALineUp
OPT_LockdBr
OPT_LockRefFreq
OPT_ShowHexNeg

OPT_RememberDetectorDetails
OPT_PanelsOnTop
OPT_UseSettlingsFromScripts
OPT_GangTraceChannels
OPT_DrawCurrentTraceBold

Methods

There are no methods available to control the Options.

5.10.1 Properties

5.10.1.1 OPT_RecentFileList

Description

This property allows specification of the number of "recently used files" to list on the bottom of the dScope File menu.

Values

The number of recently used files is entered as a short integer value. It can be any number between 0 and 16.

5.10.1.2 OPT_StartupConfiguration

Description

This property allows specification of a Configuration file to load when dScope is started.

Values

Any valid file name can be used, enclosed in double quotation marks ("..."). This should be the file name of a dScope Configuration file (*.dsc).

If a full path name is specified, the system will look for this exact file.

If a file name only is specified, then the system will look in the "Configurations" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default).

If necessary, the system will automatically append the correct filename extension (".dsc" for dScope Configuration files).

NB: **If both a startup script and a startup Configuration are specified, the Configuration will be loaded first, before the Script is run.**

5.10.1.3 OPT_StartupScript

Description

This property allows specification of a Script to run when dScope is started.

Values

Any valid file name can be used, enclosed in double quotation marks ("..."). This should be the file name of a dScope script file (*.dss).

If a full path name is specified, the system will look for this exact file.

If a file name only is specified, then the system will look in the "Scripts\Automation" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default).

If necessary, the system will automatically append the correct filename extension (".dss" for dScope script files).

NB: **If both a startup script and a startup Configuration are specified, the Configuration will be loaded first, before the Script is run.**

5.10.1.4 OPT_ConfigurationsFolder

Description

This property allows specification of the folder to use for Configuration files.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Configurations"), or a folder relative to the dScope program folder (for example, ".\My Configurations\"). The dScope program files folder is installed to "C:\Program Files\Prism Sound\dScope Series III" by default.

NB: **If the dScope is set up to use the current file's folder (see OPT_UseCurrentFilesFolder), then this Configurations folder will only be used if there is no Configuration currently loaded.**

5.10.1.5 OPT_ScriptsFolder

Description

This property allows specification of the folder to use for Scripts.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Scripts"), or a folder relative to the dScope program folder (for example, ".\My Scripts\"). The dScope program files folder is installed to "C:\Program Files\Prism Sound\dScope Series III" by default.

NB: **If the dScope is set up to use the current file's folder (see OPT_UseCurrentFilesFolder), then this Scripts folder will only be used in the Script Edit window if there is no script currently loaded.**

5.10.1.6 OPT_LimitFilesFolder

Description

This property allows specification of the folder to use for Limit files.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Limit files"), or a folder relative to the dScope program folder (for example, ".\My Limit files\"). The dScope program files folder is installed to "C:\Program Files\Prism Sound\dScope Series III" by default.

NB: **If the dScope is set up to use the current file's folder (see OPT_UseCurrentFilesFolder), then this Limit files folder will use the folder of the currently selected limit file, if one exists.**

5.10.1.7 OPT_WavetablesFolder

Description

This property allows specification of the folder to use for Generator wavetable files.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Wavetables"), or a folder relative to the dScope program folder (for example, ".\My Wavetables\"). The dScope program files folder is installed to "C:\Program Files\Prism Sound\dScope Series III" by default.

5.10.1.8 OPT_DataTablesFolder

Description

This property allows specification of the folder to use for Sweep data table files.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Sweep data tables"), or a folder relative to the dScope program folder (for example, ".\My Sweep tables\"). The dScope program files folder is installed to "C:\Program Files\Prism Sound\dScope Series III" by default.

5.10.1.9 OPT_TracesFolder

Description

This property allows specification of the folder to use for Trace files.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Traces"), or a folder relative to the dScope program folder (for example, ".\My Traces\"). The dScope program files folder is installed to "C:\Program Files\Prism Sound\dScope Series III" by default.

5.10.1.10 OPT_FFTWindowsFolder

Description

This property allows specification of the folder to use for user-defined FFT Window function files.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Window functions"), or a folder relative to the dScope program folder (for example, ".\My Window functions\"). The dScope program files folder is installed to "C:\Program Files\Prism Sound\dScope Series III" by default.

5.10.1.11 OPT_WeightingFiltersFolder

Description

This property allows specification of the folder to use for user-defined Weighting filter files.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Weighting filters"), or a folder relative to the dScope program folder (for example, ".\My Weighting filters\"). The dScope program files folder is installed to "C:\Program Files\Prism Sound\dScope Series III" by default.

5.10.1.12 OPT_EventLogsFolder

Description

This property allows specification of the folder to use for Event log files.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Event Logs"), or a folder relative to the dScope program folder (for example, ".\My Log files\"). The dScope program files folder is installed to "C:\Program Files\Prism

Sound\dScope Series III" by default.

5.10.1.13 OPT_GraphExportsFolder

Description

This property allows specification of the folder to use for files created by exporting from the Graph window, i.e. Windows metafiles created by the dScope.

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Graph exports"), or a folder relative to the dScope program folder (for example, ".\My Graph exports\"). The dScope program files folder is installed to "C:\Program Files \Prism Sound\dScope Series III" by default.

5.10.1.14 OPT_SampleBuffersFolder

Description

This property allows specification of the folder to use for sample buffers exported using the FFT Parameters panel (see FFTP_ExportSampleBuffer).

Values

Any valid folder name can be used, enclosed in double quotation marks ("..."). This can be a full path name (for example, "C:\dScope3\Sample buffer exports"), or a folder relative to the dScope program folder (for example, ".\My sample buffers\"). The dScope program files folder is installed to "C: \Program Files\Prism Sound\dScope Series III" by default.

5.10.1.15 OPT_UseCurrentFilesFolder

Description

This property selects whether to use the folder of the currently selected file in Load/Save dialogue boxes, rather than always opening at the default directory for that type of file.

For example, if this property is set to **True**, then clicking on the "Save Configuration" menu option will open the "Save" dialogue box at the folder in which the current Configuration file is stored.
If this option is set to **False**, however, the it will open at the folder specified by OPT_ConfigurationsFolder

Values

True	Always use the folder of the current file, if one exists.
False	Always use the default folder for this type of file.

5.10.1.16 OPT_LockDALineUp

Description

This property selects whether to lock the D/A line-up of the Signal Generator and the Signal Analyzer. If they are locked, then changing one will automatically change the value of the other to be the same.

If this property is set to **True**, the current Signal Generator D/A line-up will be copied into the Signal Analyzer.

Values

True	Lock together the D/A line-ups of the Signal Generator and Signal Analyzer.
False	Do not lock together the D/A line-ups of the Signal Generator and Signal Analyzer.

5.10.1.17 OPT_LockdBr

Description

This property selects whether to lock the reference amplitude of the Signal Generator and the Signal Analyzer. If they are locked, then changing one will automatically change the value of the other to be the same.

If this property is set to **True**, the current Signal Generator reference amplitude will be copied into the Signal Analyzer.

Values

True	Lock together the reference amplitudes of the Signal Generator and Signal Analyzer.
False	Do not lock together the reference amplitudes of the Signal Generator and Signal Analyzer.

5.10.1.18 OPT_LockRefFreq

Description

This property selects whether to lock the reference frequency of the Signal Generator and the Signal Analyzer. If they are locked, then changing one will automatically change the value of the other to be the same.

If this property is set to **True**, the current Signal Generator reference frequency will be copied into the Signal Analyzer.

Values

True	Lock together the reference frequencies of the Signal Generator and Signal Analyzer.
False	Do not lock together the reference frequencies of the Signal Generator and Signal Analyzer.

5.10.1.19 OPT_ShowHexNeg

Description

This property selects whether to show negative hex values with a negative sign, or just as their hex values.

For example, "**0x800001"** can also be represented as "**- 0x7FFFFF"**

Values

True	Show negative hex values preceded by a minus sign.
False	Show negative hex values as they are, without a minus sign.

5.10.1.20 OPT_RememberDetectorDetails

Description

When a function is selected on the Continuous-Time Detector or an FFT Detector, it runs a script that fills in the rest of the fields on the Detector (See Detector Functions for further details). However, these fields can then be changed and so the resulting Detector setup may be nothing like the function that is specified. In this case, the Detector's title bar shows an asterisk **(*)** to indicate that the settings have changed.

This property allows changes to be remembered when the same function is later re-selected, so that you don't have to change the function setup *every* time you re-select the same function.

For example, you may wish to select the "Amplitude" function on the Continuous-Time Detector, with a high-pass filter of 400Hz. You then change the function to measure something else, such as "Balance". When you re-select the "Amplitude" function, you wish the Detector to remember that the last time you used it, it had a 400Hz high-pass filter on it - in which case you need to ensure that the "Remember Detector details" option is turned on.

If however, the changes you make to Detectors are only temporary, you should turn this option off to ensure that re-selecting a function always resets the Detector fields to their default values.

NB: **This option will only remember Detector details within the current dScope session. If you wish to permanently change the details of a Detector function, you should change the Detector function script - see Detector Functions for further details.**

Values

True	Remember changes to Detector details the next time you re-select a function.
False	Do not remember changes to Detector details between function changes; always re-run the function script to fill in the default values.

5.10.1.21 OPT_UseSettlingsFromScripts

Description

Automation scripts often work by changing some dScope settings, and then reading a Result value. However, in some circumstances the value may be taken before it has settled from the change made to the settings. This will obviously result in an erroneous reading.

It's possible to introduce delays into the script to counter this (for example, using the Sleep method), but this may result in wasting unnecessary time. A better option may be to ensure that the Result has settled before the value is returned to the script.

This property allows you to specify that you want to apply the settling details set up for scripts (see Sweep Settling) to any Result value that a script asks for. In this way, you can guarantee that when asking for a Result, the value has settled to certain criteria and the script itself does not need to worry about checking whether a value has settled.

Note that this also applies to getting the value of a Reading (See RDG_Value), since a Reading is directly created from a Result.

Values

True	Use the Sweep settling details when reading a Result from a script.
False	Do not use any settling details when reading a Result from a script.

5.10.1.22 OPT_TriggerPointRelative

Description

This property selects whether to measurements involving a triggered sample buffer are shown relative to the trigger point (FFTP_TriggerPoint), rather than to the start of the buffer.

This affects the following measurements in dScope:
- Scope Trace X axis in ms or samples
- Entry of bins for the Window function for the impulse response (IR_StartWindowChA, etc.)

Values

True	Show measurements relative to the trigger point.
False	Show measurements relative to the start of the buffer.

5.10.1.23 OPT_UseLoadImpedance

Description

This property selects whether to take the load impedance into account when the Signal Generator amplitude is generated using a power unit of dBm or W (See SG_ChAAmplUnit). If this option is set, then the output voltage on the Analogue Outputs will be altered to ensure that it is the voltage over the specified load, rather than the open circuit voltage. The reference impedance (SG_RefImpedance) is used to specify the load impedance.

Values

True	Use the reference impedance as the load impedance to determine the voltage of the Analogue Outputs.
False	Use the reference impedance simply as a conversion factor to/from dBm and W.

5.10.1.24 OPT_WaitForMissingHardware

Description

This property selects whether the software should detect missing dScope hardware, and wait for it to be reconnected, rather than simply to display a failure error message.

This can be useful, for example, if running the dScope with a laptop when power is lost. The dScope hardware will shut down, but the software will continue to run. If the power then comes back on, the dScope will detect the return of the hardware and continue running from where it left off.

NB: **If a script is running at the time the hardware is detected as missing, the script ought to check for the hardware being missing using the IsHardwareMissing method.**

Values

True	Wait for a missing hardware to be reconnected, before continuing with measurements.
False	If an error is detected in communication with the hardware, show a failure message and stop execution.

5.10.1.25 OPT_PanelsOnTop

Description

This property selects whether settings and Reading panels should always be shown on top of the Trace Window or Carrier Display.

This maybe necessary as it's sometimes frustrating when clicking on the Trace Window, to find that other windows disappear behind it.

Values

True	Always show settings and Reading panels on top of the Trace Window.
False	Each panel is brought to the top when it becomes the active window.

5.10.1.26 OPT_GangYScales

Description

This property specifies that the Y scales of similar Traces should be ganged together when changing Y axes of a Trace.

For example, if you have created a series of similar Sweep Traces by starting them with the "Append" option turned on (See SW_Append), you would probably want the Y scales of all of them to change together. This property allows you to do just that.

Note that this only applies to Traces on the same channel. If you wish both channels to behave similarly, see OPT_GangTraceChannels.

Values

True	Gang together Y scales of all Traces of a similar type.
False	Do not gang together Y scales - each scale change will only affect the current Trace.

5.10.1.27 OPT_GangTraceChannels

Description

This property specifies that both channels should be ganged together when changing scales of a Trace.

For example, with this option turned on, then zooming into the X scale of the live Scope Trace on channel B would perform the same zoom on the X scale of the live Scope Trace on channel A.

To ensure that all Traces with similar Y scales also follow this pattern, see OPT_GangYScales.

Values

True	Gang together Traces on each channel so that scale changes are copied across to the other channel.
False	Do not gang together the Traces on both channels.

5.10.1.28 OPT_DrawCurrentTraceBold

Description

This property specifies that the currently selected Trace on the Trace window will be drawn as a bold line, for ease of recognition of the current Trace.

Values

True	Draw the current Trace as a bold line.
False	Draw the current Trace with the same width line as all other Traces.

5.11 Hardware

The Hardware section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"Hardware."**

Properties

There are no properties available to control the dScope hardware.

Methods

HW_GetMainTemp
HW_GetAnalogueTemp
HW_GetMainBoardSerialNum

5.11.1 Methods

5.11.1.1 HW_GetMainTemp

dTemp = HW_GetMainTemp ()

This method returns the temperature of the main board in the dScope hardware.

Parameters

This method has no parameters.

Return value

This method returns the temperature of the main board, in degrees centigrade. The temperature is returned as a double-precision floating point value.

5.11.1.2 HW_GetAnalogueTemp

dTemp = HW_GetAnalogueTemp ()

This method returns the temperature of the analogue board in the dScope hardware.

Parameters

This method has no parameters.

Return value

This method returns the temperature of the analogue board, in degrees centigrade. The temperature is returned as a double-precision floating point value.

5.11.1.3 HW_GetMainBoardSerialNum

sSerialNum = HW_GetMainBoardSerialNum ()

This method returns the Serial number of the dScope hardware. This Serial number is entered during the calibration process and stored in the calibration table of the main board in the hardware.

Parameters

This method has no parameters.

Return value

This method returns the Serial number of the dScope hardware, as a short integer.

5.12 dS-NET peripherals

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The dS-NET peripherals section of this reference contains details of all the properties and methods available to control peripherals attached to the dS-NET port on the dScope hardware.

Currently, the only peripherals available are the I/O Switcher and VSIO Adapter; however other peripherals may appear in future.

In a script, all properties and methods from this section must be prefixed with **"dSNet."**

Properties

DSNET_ShowErrorMessages

Methods

DSNET_Reset
DSNET_GetStatus

5.12.1 Types of dS-NET peripheral

The dS-NET connector on the back of the dScope hardware can accommodate several different dS-NET peripherals, daisy-chained together if necessary.

The general dS-NET class contains general methods and properties for resetting devices, and obtaining device status. These can be used by any dS-NET peripheral.

Within this class, different types of devices can exist. These can utilise the general dS-NET methods and properties, and also have a number of their own to manage this type of device.

At the bottom level are the specific devices themselves. These can use any of the methods and properties of their parent device type, and also have specific methods and properties of their own.

For example, the current device hierarchy is as follows:

dS-NET peripherals
(General methods, such as Reset, GetStatus)
Switchers
(Switcher-specific properties/methods)
I/O Switchers
(I/O Switcher-specific properties/methods)
Format converters
(Format converter-specific properties/methods)
VSIO Adapters
(VSIO Adapter-specific properties/methods)

5.12.2 Properties

5.12.2.1 DSNET_ShowErrorMessages

Description

This property allows you to specify whether or not to display error messages if a call to any dS-NET routines fails.

If you are performing several operations using various dS-NET devices, you may wish to detect presence or absence of a device by whether it returns **True** or **False** to a command; if this is the case, you probably wouldn't want an error message popping up telling you that the call had timed out waiting for a response from the hardware, so you would set this property to **False**.

This property is set to **True** by default.

NB: **If the Display method has been used to hide the dScope main window, then error messages will NOT be shown, regardless of the value of this property.**

Values

True	Shows error messages when a call to a dS-NET peripheral fails.
False	Does not show error messages when a call to a dS-NET peripheral fails.

5.12.3 Methods

5.12.3.1 DSNET_Reset

bRet = DSNET_Reset (sAddress, bOn, plStatus)

This method resets the specified dS-NET peripheral to either On or in Standby mode, and returns the current status of the device.

NB: **This method must be used for a device before any further methods will work on it.**

Parameters

sAddress	The address of the device (can be 0 to 63).
bOn	**True** to turn the device on, **False** to turn it off and put it into standby mode.
plStatus	A variable that will be filled in with the current status of the device. The status is a four-byte long integer, which will contain the device status as follows: **Byte 0** (`plStatus And &HFF`) Bits [7..4] are the device class Bits [3..0] are the type (See Types of dS-NET peripheral for further details) **Byte 1** (`(plStatus / 256) And &HFF`) Bits [7..4] are the firmware version number Bits [3..0] are the hardware version number **Byte 2** (`(plStatus / 65536) And &HFF`) Bit 0 specifies whether the device is On Bit 1 - if Bit 0 is 1, and Bit 1 is 1, specifies that device's settings are all at Reset state. **Byte 3** (`(plStatus / 16777216) And &HFF`) This byte is unused (0).

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because no device exists with the given address.

5.12.3.2 DSNET_GetStatus

bRet = DSNET_GetStatus (sAddress, plStatus)

This method returns the status of the specified dS-NET peripheral.

Parameters

sAddress	The address of the device (can be 0 to 63).
plStatus	A variable that will be filled in with the current status of the device. The status is a four-byte long integer, which will contain the device status as follows: **Byte 0** (`plStatus And &HFF`) Bits [7..4] are the device class Bits [3..0] are the type (See Types of dS-NET peripheral for further details) **Byte 1** (`(plStatus / 256) And &HFF`) Bits [7..4] are the firmware version number Bits [3..0] are the hardware version number **Byte 2** (`(plStatus / 65536) And &HFF`) Bit 0 specifies whether the device is On Bit 1 - if Bit 0 is 1, and Bit 1 is 1, specifies that device's settings are all at Reset state. **Byte 3** (`(plStatus / 16777216) And &HFF`) This byte is unused (0).

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because no device exists with the given address.

5.12.4 Channel Arrays

NB: This part of the dScope's scripting interface may not be available, depending on the dScope model number.

When testing using multiple channels, it is not unlikely that more than one device (such as a switcher) will be daisy-chained together to give several input or output channels. In this case, there are two ways of working - you can either calculate which device, group and bus should currently be selected, or you can set up a Channel Array.

A Channel Array makes script-writing easier, in that you set up an array in a Configuration or at the beginning of the script, containing all the channels that you want to include, and then further commands simply use the channel number in the array. This means that the dScope does all the calculations of which device, bus or group a channel is on, and the script can simply specify the channel number in the array.

Using Channel Arrays

Channel Arrays must be set using the dS-NET Setup window, or from a script using DSNET_DefineChannelArray and then adding buses/groups to the array using methods specific to the type of dS-NET device. Once this has happened, it is part of the dS-NET Setup and its details will be saved if the current setup is saved as a Configuration. See the section on the dS-NET Peripherals Setup dialogue box in the Operation manual for further details of setting up Channel Arrays.

Once a Channel Array has been defined, the script must select it as the current Channel Array before accessing its properties and methods. Once selected, all properties and methods of a Channel Array will then access that selected array until the current array is changed. To select the current array, use the DSNET_SetChannelArray method.

As an example, the following code snippet accesses a previously defined Channel Array called "Mono Inputs" and turns on channel 4 exclusively.

```
If dSNet.DSNET_SetChannelArray("Mono Inputs") Then
   ChannelArray.CA_ExclusiveChannel(4)
End If
```

Example

The following example considers four I/O Switchers (addresses 1..4) all set up to test a 64-channel mixing desk, with all desk outputs being routed to channel A of the dScope's Analogue Inputs.

The hard way of doing things would require you to calculate which device, bus and group you were using EVERY time you needed to select a different channel. You would also need to calculate which OTHER devices were in use, and turn off all the channels on those devices as well.

Using a Channel Array, you would simply define the array by adding each group (X and Y) on bus A to the array as follows:

```
' Define a mono array
dSNet.DSNET_DefineArray("desk64", False)
IOSwitcher.IOSWITCHER_AddToArray(1,
   IOSWITCHER_BUS_A, IOSWITCHER_GROUP_X, "desk64")
IOSwitcher.IOSWITCHER_AddToArray(1,
   IOSWITCHER_BUS_A, IOSWITCHER_GROUP_Y, "desk64")
IOSwitcher.IOSWITCHER_AddToArray(2,
   IOSWITCHER_BUS_A, IOSWITCHER_GROUP_X, "desk64")
IOSwitcher.IOSWITCHER_AddToArray(2,
   IOSWITCHER_BUS_A, IOSWITCHER_GROUP_Y, "desk64")
IOSwitcher.IOSWITCHER_AddToArray(3,
   IOSWITCHER_BUS_A, IOSWITCHER_GROUP_X, "desk64")
IOSwitcher.IOSWITCHER_AddToArray(3,
   IOSWITCHER_BUS_A, IOSWITCHER_GROUP_Y, "desk64")
IOSwitcher.IOSWITCHER_AddToArray(4,
   IOSWITCHER_BUS_A, IOSWITCHER_GROUP_X, "desk64")
IOSwitcher.IOSWITCHER_AddToArray(4,
   IOSWITCHER_BUS_A, IOSWITCHER_GROUP_Y, "desk64")
```

Switching a specified channel on exclusively would then be a single call :

```
' Set which array to use
dSNet.DSNET_SetChannelArray("desk64")
' Set the relevant channel, turning others off
ChannelArray.CA_ExclusiveChannel(29)
```

NB:

With the exception of Switcher Load relays, it is recommended that you EITHER use the Channel Array method, OR specify relay masks for individual Bus/Group combinations. If you decide to use some Channel Array commands mixed with relay mask commands, you may get unpredictable results.

Properties

There are no properties available to control Channel Arrays.

Methods

The following methods are used to set up Channel Arrays. In a script, they must be prefixed with **"dSNet."**

DSNET_DefineChannelArray
DSNET_RemoveChannelArray
DSNET_SetChannelArray

The following methods are used to manipulate the channels in a Channel Array. In a script, they must be prefixed with **"ChannelArray."**

CA_ExclusiveChannel
CA_NotChannel
CA_AddChannel
CA_RemoveChannel
CA_ClearChannels
CA_SetAllChannels
CA_Balance

5.12.4.1 Methods

5.12.4.1.1 DSNET_DefineChannelArray

bRet = DSNET_DefineChannelArray (strArray, bStereo, bExclusiveOnly)

This method allows you to set up an array of channels, which can then be accessed by the channel number in this array, rather than needing to calculate a channel's device address, bus and group each time you need to turn it on or off.

For details on how to use Channel Arrays from a script, see Channel Arrays.

NB:

If you define a Channel Array from a Script, you should remove it after use with the DSNET_RemoveChannelArray method. Otherwise, the array will remain in memory until the dScope program is exited.
This may also cause problems if you re-run a script, as it can add channels to an array several times. This would cause channels to be switched more than once unnecessarily for a single operation.

Parameters

strArray	The name to give the Channel Array.
bStereo	**True** to define a stereo array (i.e. successive channels are automatically set to be on bus A then bus B alternately); **False** to define a mono array.
bExclusiveOnly	**True** to specify that the array can only be used by turning channels on exclusively. If exclusive only, the only channel operations that can be done on the array are CA_ExclusiveChannel and CA_ClearChannels. This prevents more than one relay connection being driven at the same time, which can be problematic with certain devices, for example power amplifiers.

Return value

This method returns **True** if the array was defined successfully, or **False** if it fails.

5.12.4.1.2 DSNET_RemoveChannelArray

bRet = DSNET_RemoveChannelArray (strArray)

This method allows you to remove a Channel Array after you have finished using it (usually at the end of a script in which you have defined it).

This is necessary in order to free the memory that the dScope has used to store the array.

For details on how to use Channel Arrays, see Channel Arrays.

Parameters

strArray	The name of the Channel Array to remove.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has not been defined.

5.12.4.1.3 DSNET_SetChannelArray

bRet = DSNET_SetChannelArray (strArray)

This method sets the current Channel Array for use in a script. Subsequent Channel Array operations will act on this Channel Array, until the array is changed.

For details on how to use Channel Arrays from a script, see Channel Arrays.

Parameters

strArray	The name of the Channel Array to set as current.

Return value

This method returns **True** if the array was set successfully, or **False** if it fails. This may be because the array with the specified name has not been defined.

5.12.4.1.4 CA_ExclusiveChannel

bRet = CA_ExclusiveChannel (sChannel)

This method turns on exclusively the specified channel, turning off all other channels in the Channel Array.

NB: **Before this method can be used, the current Channel Array must be specified using DSNET_SetChannelArray.**

Parameters

sChannel	The channel to turn on. This must be a number between 1 and the number of channels in the array. NB: If the Channel Array has been set up as a stereo array, then the given channel will be turned on together with the other channel in the stereo pair.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the channel does not exist in the array.

5.12.4.1.5 CA_NotChannel

bRet = CA_NotChannel (sChannel)

This method turns off exclusively the specified channel, turning on all other channels in the Channel Array.

NB: **Before this method can be used, the current Channel Array must be specified using DSNET_SetChannelArray.**

Parameters

sChannel	The channel to turn off. This must be a number between 1 and the number of channels in the array. NB: If the Channel Array has been set up as a stereo array, then the given channel will be turned off together with the other channel in the stereo pair.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the channel does not exist in the array, or because the array has been set up to be "Exclusive only" (see DSNET_DefineChannelArray).

5.12.4.1.6 CA_AddChannel

bRet = CA_AddChannel (sChannel)

This method turns on the specified channel, without affecting the status of other channels in the Channel Array.

NB: **Before this method can be used, the current Channel Array must be specified using DSNET_SetChannelArray.**

Parameters

sChannel	The channel to turn on. This must be a number between 1 and the number of channels in the array. NB: If the Channel Array has been set up as a stereo array, then the given channel will be turned on together with the other channel in the stereo pair.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the channel passed does not exist in the array, or because the array has been set up to be "Exclusive only" (see DSNET_DefineChannelArray).

5.12.4.1.7 CA_RemoveChannel

bRet = CA_RemoveChannel (sChannel)

This method turns off the specified channel in a channel array, without affecting the status of other channels in the array.

NB: **Before this method can be used, the current Channel Array must be specified using DSNET_SetChannelArray.**

Parameters

sChannel	The channel to turn off. This must be a number between 1 and the number of channels in the array. NB: If the Channel Array has been set up as a stereo array, then the given channel will be turned off together with the other channel in the stereo pair.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the channel passed does not exist in the array, or because the array has been set up to be "Exclusive only" (see DSNET_DefineChannelArray).

5.12.4.1.8 CA_ClearChannels

bRet = CA_ClearChannels ()

This method turns off all channels in the array.

NB: **Before this method can be used, the current Channel Array must be specified using DSNET_SetChannelArray.**

Parameters

This method has no parameters.

Return value

This method returns **True** if it was successful, or **False** if it fails.

5.12.4.1.9 CA_SetAllChannels

bRet = CA_SetAllChannels ()

This method turns on all channels in the array.

NB: **Before this method can be used, the current Channel Array must be specified using DSNET_SetChannelArray.**

Parameters

This method has no parameters.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has been set up to be "Exclusive only" (see DSNET_DefineChannelArray).

5.12.4.1.10 CA_Balance

bRet = CA_Balance (bOn)

This method sets balance on or off for all channels in the array. This method currently only applies to Channel Arrays consisting of I/O Switchers.

NB: **Before this method can be used, the current Channel Array must be specified using DSNET_SetChannelArray.**

Parameters

bOn **True** to turn balance on, **False** to turn it off.

Return value

This method returns **True** if it was successful, or **False** if it fails.

5.12.5 Switchers

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The Switchers section of this reference contains details on how to control different types of switcher.

The only switcher currently available is the I/O Switcher. Further types of switcher may be added in the future.

Channel Arrays

Switchers can be set up to use Channel Arrays, which makes manipulating of switcher channels much easier and makes them available to Sweeps. See Channel Arrays for more details.

Properties

There are no properties available to control switchers.

Methods

NB: **The following methods are provided for legacy support only and should not be used.**

SWITCHER_ExclusiveChannel
SWITCHER_NotChannel
SWITCHER_AddChannel
SWITCHER_RemoveChannel
SWITCHER_ClearChannels
SWITCHER_Balance
SWITCHER_DefineArray
SWITCHER_DefineStereoArray
SWITCHER_RemoveArray

5.12.5.1 Methods

5.12.5.1.1 SWITCHER_ExclusiveChannel

NB:

This method is included for support of legacy scripts only.
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the CA_ExclusiveChannel method.

bRet = SWITCHER_ExclusiveChannel (strArray, sChannel)

This method turns on exclusively the specified channel in a Channel Array, turning off all other channels in the array.

For further details on how to set up an array of channels, see SWITCHER_DefineArray or SWITCHER_DefineStereoArray.

NB:

If the array has been set up as a stereo array (using SWITCHER_DefineStereoArray), then the given channel will be turned on, and the other channel in this stereo pair.

Parameters

strArray	The Channel Array that this channel is part of. This array must have been set up using SWITCHER_DefineArray or SWITCHER_DefineStereoArray.
sChannel	The channel to turn on. This must be a number between 1 and the number of channels in the array.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has not been defined, or the channel passed does not exist in the array.

5.12.5.1.2 SWITCHER_NotChannel

NB:

This method is included for support of legacy scripts only.
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the CA_NotChannel method.

bRet = SWITCHER_NotChannel (strArray, sChannel)

This method turns off exclusively the specified channel in a Channel Array, turning on all other channels in the array.

For further details on how to set up an array of channels, see SWITCHER_DefineArray or SWITCHER_DefineStereoArray.

NB:

If the array has been set up as a stereo array (using SWITCHER_DefineStereoArray), then the given channel will be turned off, and

the other channel in this stereo pair.

Parameters

strArray	The Channel Array that this channel is part of. This array must have been set up using SWITCHER_DefineArray or SWITCHER_DefineStereoArray.
sChannel	The channel to turn off. This must be a number between 1 and the number of channels in the array.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has not been defined, or the channel passed does not exist in the array.

5.12.5.1.3 SWITCHER_AddChannel

NB:

This method is included for support of legacy scripts only.
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the CA_AddChannel method.

bRet = SWITCHER_AddChannel (strArray, sChannel)

This method turns on the specified channel in a Channel Array, without affecting the status of other channels in the array.

For further details on how to set up an array of channels, see SWITCHER_DefineArray or SWITCHER_DefineStereoArray.

NB:

If the array has been set up as a stereo array (using SWITCHER_DefineStereoArray), then the given channel will be turned on, and the other channel in this stereo pair.

Parameters

strArray	The Channel Array that this channel is part of. This array must have been set up using SWITCHER_DefineArray or SWITCHER_DefineStereoArray.
sChannel	The channel to turn on. This must be a number between 1 and the number of channels in the array.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has not been defined, or the channel passed does not exist in the array.

5.12.5.1.4 SWITCHER_RemoveChannel

NB:

This method is included for support of legacy scripts only.
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the CA_RemoveChannel method.

bRet = SWITCHER_RemoveChannel (strArray, sChannel)

This method turns off the specified channel in a Channel Array, without affecting the status of other channels in the array.

For further details on how to set up an array of channels, see SWITCHER_DefineArray or SWITCHER_DefineStereoArray.

NB: **If the array has been set up as a stereo array (using SWITCHER_DefineStereoArray), then the given channel will be turned on, and the other channel in this stereo pair.**

Parameters

strArray	The Channel Array that this channel is part of. This array must have been set up using SWITCHER_DefineArray or SWITCHER_DefineStereoArray.
sChannel	The channel to turn off. This must be a number between 1 and the number of channels in the array.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has not been defined, or the channel passed does not exist in the array.

5.12.5.1.5 SWITCHER_ClearChannels

NB: **This method is included for support of legacy scripts only.**
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the CA_ClearChannels method.

bRet = SWITCHER_ClearChannels (strArray)

This method turns off all channels in the specified array.

For further details on how to set up an array of channels, see SWITCHER_DefineArray or SWITCHER_DefineStereoArray.

Parameters

strArray	The Channel Array to clear all channels of. This array must have been set up using SWITCHER_DefineArray or SWITCHER_DefineStereoArray.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has not been defined.

5.12.5.1.6 SWITCHER_Balance

NB:

This method is included for support of legacy scripts only.
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the CA_Balance method.

bRet = SWITCHER_Balance (strArray, bOn)

This method sets balance on or off for all channels in the specified array.

For further details on how to set up an array of channels, see SWITCHER_DefineArray or SWITCHER_DefineStereoArray.

Parameters

strArray	The Channel Array to set the balance for. This array must have been set up using SWITCHER_DefineArray or SWITCHER_DefineStereoArray.
bOn	**True** to turn balance on, **False** to turn it off.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has not been defined.

5.12.5.1.7 SWITCHER_DefineArray

NB:

This method is included for support of legacy scripts only.
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the DSNET_DefineChannelArray method.

bRet = SWITCHER_DefineArray (strArray)

This method allows you to set up an array of channels, which can then be accessed by the channel number in this array, rather than needing to calculate a channel's device address, bus and group each time you need to turn it on or off.

NB:

When you have finished using an array, you MUST remove it using SWITCHER_RemoveArray. Otherwise, the array will remain in memory until the dScope program is exited.
This may also cause problems if you re-run a script, as it can add channels to an array several times and channels may be switched more than once to perform a single operation.

Parameters

strArray	The name of the Channel Array to set the balance for. This can be any string, and will be the name used to refer to this array for future calls.

Return value

This method returns **True** if it was successful, or **False** if it fails.

5.12.5.1.8 SWITCHER_DefineStereoArray

NB: **This method is included for support of legacy scripts only.**
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the DSNET_DefineChannelArray method.

bRet = SWITCHER_DefineStereoArray (strArray)

This method allows you to set up an array of channels, which can then be accessed by the channel number in this array, rather than needing to calculate a channel's device address and bus each time you need to turn it on or off.
The array will be set up such that adding a set of channels to it will add alternate channels on different buses - for example, adding group X to a Stereo array will result in channels 1, 3, 5 and 7 being added on bus A, and channels 2, 4, 6 and 8 being added on bus B.

For details of how each type of switcher adds channels to a stereo array, see the correct page for the appropriate switcher (e.g. IOSWITCHER_AddToStereoArray)

NB: **When you have finished using an array, you MUST remove it using SWITCHER_RemoveArray. Otherwise, the array will remain in memory until the dScope program is exited.**
This may also cause problems if you re-run a script, as it can add channels to an array several times and channels may be switched more than once to perform a single operation.

Parameters

strArray	The channel array to define. This can be any string, and will be the name used to refer to this array for future calls.

Return value

This method returns **True** if it was successful, or **False** if it fails.

5.12.5.1.9 SWITCHER_RemoveArray

NB: **This method is included for support of legacy scripts only.**
Channel Arrays should now be set up and manipulated via the dSNet object (see Channel Arrays for details).
This method has been relaced by the DSNET_RemoveChannelArray method.

bRet = SWITCHER_RemoveArray (strArray)

This method allows you to remove an array of channels after you have finished using it (usually at the end of a script in which you have defined it).

This is necessary in order to free the memory that the dScope has used to store the array.

Parameters

strArray	The name of the Channel Array to remove.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the array has not been defined.

5.12.5.2 I/O Switchers

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The I/O Switchers section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"IOSwitcher."**

Properties

There are no properties available to control I/O Switchers.

Methods

IOSWITCHER_GetBusStatus
IOSWITCHER_GetFullStatus
IOSWITCHER_BusGroupSwitch
IOSWITCHER_BusLoad
IOSWITCHER_BusBalance
IOSWITCHER_GetBusDC
IOSWITCHER_AddToArray
IOSWITCHER_AddToStereoArray

5.12.5.2.1 Methods

5.12.5.2.1.1 IOSWITCHER_GetBusStatus

bRet = IOSWITCHER_GetBusStatus (sAddress, sBus, plStatus)

This method returns the status of the specified bus on the I/O Switcher with the given address.

Parameters

sAddress	The address of the I/O Switcher (can be 0 to 63).
sBus	The bus to get the status for. It can be **IOSWITCHER_BUS_A** or **IOSWITCHER_BUS_B**.
plStatus	A variable that will be filled in with the current status of the selected bus. The status is a four-byte long integer, which will contain the bus status as follows: **Byte 0** (`plStatus And &HFF`) Bits 0..7 are channels 1..8 of the X group. 1s mean that the relays are set (On); 0s mean that the relays are not set (Off). **Byte 1** (`(plStatus / 256) And &HFF`) Bits 0..7 are channels 1..8 of the Y group. 1s mean that the relays are set (On); 0s mean that the relays are not set (Off). **Byte 2** (`(plStatus / 65536) And &HFF`) Bit 0 set to 1 means that this bus is set to Balanced. Bit 1 set to 1 means that this bus is set as Loaded. **Byte 3** (`(plStatus / 16777216) And &HFF`) This byte is unused (0).

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because no device exists with the given address, or the specified device is not an I/O Switcher.

5.12.5.2.1.2 IOSWITCHER_GetFullStatus

bRet = IOSWITCHER_GetFullStatus (sAddress, plBusAStatus, plBusBStatus)

This method returns the status of both A and B buses of the I/O Switcher with the given address.

Parameters

sAddress	The address of the I/O Switcher (can be 0 to 63).
plBusAStatus	A variable that will be filled in with the current status of bus A. The status is a four-byte long integer, which will contain the bus status as follows: **Byte 0** (`plStatus And &HFF`) Bits 0..7 are channels 1..8 of the X group. 1s mean that the relays are set (On); 0s mean that the relays are not set (Off). **Byte 1** (`(plStatus / 256) And &HFF`) Bits 0..7 are channels 1..8 of the Y group. 1s mean that the relays are set (On); 0s mean that the relays are not set (Off). **Byte 2** (`(plStatus / 65536) And &HFF`) Bit 0 set to 1 means that this bus is set to Balanced. Bit 1 set to 1 means that this bus is set as Loaded. **Byte 3** (`(plStatus / 16777216) And &HFF`) This byte is unused (0).
plBusBStatus	A variable that will be filled in with the current status of bus B. The status is a four-byte long integer, which will contain channel B's bus status (as described above for bus A).

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because no device exists with the given address, or the specified device is not an I/O Switcher.

5.12.5.2.1.3 IOSWITCHER_BusGroupSwitch

bRet = IOSWITCHER_BusGroupSwitch(sAddress, sBus, sGroup, sMask)

This method sets the relays for a specified bus and group combination.

If the equipment connected to the switcher may be damaged by having more than one relay connected at a time (e.g. power amplifiers), you must make sure that you include an extra call to turn all relays OFF before turning the new relays on.
For example, calling this with a Mask value of 2 (channel 2 of this group only) and then with a value of 4 (channel 3 of this group only) may result in a very short period of time where both relays are connected.
To ensure that this does not happen, include a call to this method using a mask of 0 in between calls to turn new relays on. This will ensure that all connections are broken before any new ones are made.

Parameters

sAddress	The address of the I/O Switcher (can be 0 to 63).
sBus	The bus to switch the relays for. It can be **IOSWITCHER_BUS_A** or **IOSWITCHER_BUS_B**.
sGroup	The group to switch the relays for. It can be **IOSWITCHER_GROUP_X** or **IOSWITCHER_GROUP_Y**.
sMask	The mask to use to set the relays. This is an 8-bit value, where bits 0..7 correspond to relays 1..8 on this bus/group. For example, turning on relays 1, 3 and 4 for this bus/group would need bits 0, 2 and 3 to be set, i.e. a value of &H0D (Hex) or 13 (decimal).

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because no device exists with the given address, or the ***sBus*** or ***sGroup*** parameters are invalid.

5.12.5.2.1.4 IOSWITCHER_BusLoad

bRet = IOSWITCHER_BusLoad(sAddress, sBus, bOn)

This method sets the Load relays for the given bus on the specified device.

Parameters

sAddress	The address of the I/O Switcher (can be 0 to 63).
sBus	The bus to switch the Load relays for. It can be **IOSWITCHER_BUS_A** or **IOSWITCHER_BUS_B**.
bOn	**True** to switch the Load relays on; **False** to turn them off.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because no device exists with the given address, or the ***sBus*** or ***sGroup*** parameters are invalid.

5.12.5.2.1.5 IOSWITCHER_BusBalance

bRet = IOSWITCHER_BusBalance(sAddress, sBus, bOn)

This method sets the Balance relays for the given bus on the specified device.

Parameters

sAddress	The address of the I/O Switcher (can be 0 to 63).
sBus	The bus to switch the Balance relays for. It can be **IOSWITCHER_BUS_A** or **IOSWITCHER_BUS_B**.
bOn	**True** to switch the Balance relays on; **False** to turn them off.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because no device exists with the given address, or the ***sBus*** parameter is invalid.

5.12.5.2.1.6 IOSWITCHER_GetBusDC

bRet = IOSWITCHER_GetBusDC(sAddress, sBus, psPositiveDC, psNegativeDC)

This method returns the positive and negative DC voltages of the specified bus on the I/O Switcher with the given address.

NB:

This method is not implemented for I/O Switchers with hardware version A.
For I/O Switchers with hardware version C, and firmware version A or B, you may need to insert a delay of 250ms between subsequent calls to this method on the same switcher address.
To find out the hardware and firmware versions of dS-NET peripherals attached to the dScope, use the dS-NET Peripherals Setup dialogue box on the Utility menu.

Parameters

sAddress	The address of the I/O Switcher (can be 0 to 63).
sBus	The bus to get the DC voltage for. It can be **IOSWITCHER_BUS_A** or **IOSWITCHER_BUS_B**.
psPositiveDC	A variable that will be filled in with the current positive DC voltage on the specified bus. The voltage is a short integer, rounded to the nearest Volt.
psNegativeDC	A variable that will be filled in with the current negative DC voltage on the specified bus. The voltage is a short integer, rounded to the nearest Volt.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because no device exists with the given address, or the ***sAddress*** or ***sBus*** parameters are invalid.

5.12.5.2.1.7 IOSWITCHER_AddToArray

bRet = IOSWITCHER_AddToArray(sAddress, sBus, sGroup, strArray)

This method allows you to add a group of channels on a particular bus to a pre-defined Channel Array.

For details on how to use Channel Arrays, see Channel Arrays.

Parameters

sAddress	The address of the I/O Switcher (can be 0 to 63).
sBus	The bus to add the relays for. This can be **IOSWITCHER_BUS_A** or **IOSWITCHER_BUS_B**.
sGroup	The group to add the relays for. It can be **IOSWITCHER_GROUP_X** or **IOSWITCHER_GROUP_Y**.
strArray	The Channel Array to add the group to. This must be an array already defined using DSNET_DefineChannelArray.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the specified device does not exist, the sBus or sGroup parameters may be invalid, or the array may be undefined. This method also returns **False** if the array was set up as a stereo array (see DSNET_DefineChannelArray).

5.12.5.2.1.8 IOSWITCHER_AddToStereoArray

bRet = IOSWITCHER_AddToStereoArray(sAddress, sGroup, strArray)

This method allows you to add a group of channels on a particular bus to a pre-defined stereo Channel Array. This means that stereo pairs of channels will be added, the first to bus A and the second to bus B.

For example, adding group X to a Stereo array will result in channels 1, 3, 5 and 7 being added on bus A, and channels 2, 4, 6 and 8 being added on bus B.

For details on how to use Channel Arrays, see Channel Arrays.

Parameters

sAddress	The address of the I/O Switcher (can be 0 to 63).
sGroup	The group to add the relays for. It can be **IOSWITCHER_GROUP_X** or **IOSWITCHER_GROUP_Y**.
strArray	The Channel Array to add the group to. This must be an array already defined using DSNET_DefineChannelArray.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the specified device does not exist, the sGroup parameter may be invalid, or the array may be undefined. This method also returns **False** if the array was not set up as a stereo array (see DSNET_DefineChannelArray).

5.12.6 Format Converters

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The Format Converters section of this reference contains details on how to control different types of Format Converter.

The only Format Converter currently available is the VSIO Adapter. Further types of Format Converter may be added in the future.

5.12.6.1 VSIO Adapters

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The VSIO Adapters section of this reference contains details of the following properties and methods.

In a script, all properties and methods from this section must be prefixed with **"VSIOAdapter."**

Channel Arrays

The channels of a VSIO Adapter can be set up as a Channel Array, which makes manipulating of channels much easier and makes them available to Sweeps. See Channel Arrays for more details.

Properties

VSIO_EnableGenerator
VSIO_EnableAnalyzer
VSIO_AudioOn
VSIO_AudioVoltage
VSIO_ControlOn
VSIO_ControlVoltage

VSIO_SPIClockPolarity
VSIO_SPIClockPhase

The following properties available to both the Generator and Analyzer. In a script, these must be prefixed using **"VSIOAdapter.Generator."** or **"VSIOAdapter.Analyzer."** as required.

VSIO_SlotLength
VSIO_LSBFirst
VSIO_LeadPadLength
VSIO_DataLength
VSIO_TrailPadLength
VSIO_SlotsPerWire
VSIO_SignExtend
VSIO_SerialClockDir
VSIO_FrameClockInvert
VSIO_FrameClock1Bit
VSIO_FrameClockEarly
VSIO_FrameClockFreq
VSIO_BitClockInvert
VSIO_BitClockFreq
VSIO_MasterClockDir
VSIO_MasterClockMultiplier
VSIO_MasterClockFreq
VSIO_Delay

Methods

VSIO_SetCurrentDevice
VSIO_SetGeneratorRouting
VSIO_SetAnalyzerRouting
VSIO_SendSPIData
VSIO_SendI2CData

5.12.6.1.1 Properties

5.12.6.1.1.1 VSIO_EnableGenerator

Description

This property specifies whether to enable generation of audio data by the VSIO Adapter. If set to **False**, AES3 data is set to pass through the adapter.

Values

True	Enables generation of VSIO data.
False	Audio data is set to pass through the VSIO Generator (default).

5.12.6.1.1.2 VSIO_EnableAnalyzer

Description

This property specifies whether to enable analysis of audio data by the VSIO Adapter. If set to **False**, AES3 data is set to pass through the adapter.

Values

True	Enables analysis of VSIO data.
False	Audio data is set to pass through the VSIO Analyzer (default).

5.12.6.1.1.3 VSIO_AudioOn

Description

This property turns the Audio port of the VSIO Adapter on or off.

Values

True	Turns on the Audio port.
False	Turns off the Audio port.

It is VERY IMPORTANT that the correct voltage is selected for the Audio port (VSIO_AudioVoltage) BEFORE powering the port using this property. Failure to observe this may result in permanent damage to the VSIO Adapter, or the EUT, or both.

5.12.6.1.1.4 VSIO_AudioVoltage

Description

This property sets the voltage for the Audio port of the VSIO Adapter.

Values

VSIO_VSEL_1V8	Sets the voltage for the Audio port to 1.8V
VSIO_VSEL_2V5	Sets the voltage for the Audio port to 2.5V
VSIO_VSEL_3V3	Sets the voltage for the Audio port to 3.3V
VSIO_VSEL_5V0_CMOS	Sets the voltage for the Audio port to 5.0V (CMOS threshold)
VSIO_VSEL_5V0_TTL	Sets the voltage for the Audio port to 5.0V (TTL threshold)

It is VERY IMPORTANT that the correct voltage is selected for the Audio port BEFORE powering the port using VSIO_AudioOn. Failure to observe this may result in permanent damage to the VSIO Adapter, or the EUT, or both.

5.12.6.1.1.5 VSIO_ControlOn

Description

This property turns the Control port of the VSIO Adapter on or off.

Values

True	Turns on the Control port.
False	Turns off the Control port.

It is VERY IMPORTANT that the correct voltage is selected for the Control port (VSIO_ControlVoltage) BEFORE powering the port using this property. Failure to observe this may result in permanent damage to the VSIO Adapter, or the EUT, or both.

5.12.6.1.1.6 VSIO_ControlVoltage

Description

This property sets the voltage for the Control port of the VSIO Adapter.

Values

VSIO_VSEL_1V8	Sets the voltage for the Control port to 1.8V
VSIO_VSEL_2V5	Sets the voltage for the Control port to 2.5V
VSIO_VSEL_3V3	Sets the voltage for the Control port to 3.3V
VSIO_VSEL_5V0_CMOS	Sets the voltage for the Control port to 5.0V (CMOS threshold)
VSIO_VSEL_5V0_TTL	Sets the voltage for the Control port to 5.0V (TTL threshold)

It is VERY IMPORTANT that the correct voltage is selected for the Control port BEFORE powering the port using VSIO_ControlOn. Failure to observe this may result in permanent damage to the VSIO Adapter, or the EUT, or both.

5.12.6.1.1.7 VSIO_SlotLength

Description

This property specifies the Slot length of the VSIO data format, i.e. the number of bit periods in one 'Slot' (channel) of the serial audio multiplex.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

The Slot length can be set to **8**, **16**, **24** or **32** bit periods.

5.12.6.1.1.8 VSIO_SlotsPerWire

Description

This property specifies the number of Slots on each wire. The VSIO Adapter can generate different channel data on each of up to four wires, up to a total of 16 channels.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

2	Sets the data format to 2 Slots per wire (4 wires)
4	Sets the data format to 4 slots per wire (4 wires)
8	Sets the data format to 8 Slots per wire (2 wires)
16	Sets the data format to 16 slots per wire (1 wire)

5.12.6.1.1.9 VSIO_DataLength

Description

This property specifies the number of active audio bits in the multiplex.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

The data length can be a minimum of 8, and a maximum of 24 or the Slot length, whichever is lowest.

i.e. for a Slot length of 8, the data length must be 8 bits. For a Slot length of 16 or 24, the data length can be between 8 and 16 or 24 respectively. For a Slot length of 32, the data length can be between 8 and 24 bits.

5.12.6.1.1.10 VSIO_LeadPadLength

Description

This property specifies the position of the audio within the Slot.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

The lead pad length can be between 0 and the Slot length minus the data length.

For example, if the Slot length is 24, and the data length is 16, then the lead pad length can be between 0 and 8.

5.12.6.1.1.11 VSIO_TrailPadLength

Description

This **read-only** property returns the spacing between the end of the audio bits and the end of the Slot.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

The trail pad length will be between 0 and (Slot length minus data length minus lead pad length).

5.12.6.1.1.12 VSIO_LSBFirst

Description

This property specifies whether the audio bits in the Slot should be LSB first rather than MSB first.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

True	Audio bits are LSB first.
False	Audio bits are MSB first (default).

5.12.6.1.1.13 VSIO_SignExtend

Description

This property specifies whether the padding at the MSB end of the audio word should be filled with copies of the MSB (sign bit) instead of the usual zero value.

NB: **This property only applies to the Generator part of the VSIO Adapter object. From a script, it must be prefixed with "VSIOAdapter.Generator."**

Values

True	MSB padding is sign-extended.
False	MSB padding is not sign-extended (default)

5.12.6.1.1.14 VSIO_SerialClockDir

Description

This property sets the direction of the serial bit clock (SCK) and the wordclock/frame clock (LRCK).

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

VSIO_CLOCKDIR_IN	Sets the serial clocks to be received from the EUT by the VSIO Adapter.
VSIO_CLOCKDIR_OUT	Sets the serial clocks to be driven by the VSIO Adapter into the EUT (EUT is slaved to the VSIO/dScope).

5.12.6.1.1.15 VSIO_FrameClockInvert

Description

This property specifies whether to invert the frame clock.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

True	Frame clock is inverted.
False	Frame clock is not inverted (default).

5.12.6.1.1.16 VSIO_FrameClock1Bit

Description

This property specifies whether the frame clock should be one bit width, rather than an equal mark/space square wave.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

True	Frame clock is high (or low if inverted) for a single bit period.
False	Frame clock is high (or low if inverted) for half the duration of the multiplex frame (default).

5.12.6.1.1.17 VSIO_FrameClockEarly

Description

This property specifies whether the frame clock's active edge should occur one bit period *before* the beginning of the frame's data.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

True	Frame clock's active edge is one bit period before the data.
False	Frame clock's active edge coincides with the start of the multiplex frame (default).

5.12.6.1.1.18 VSIO_FrameClockFreq

Description

This **read-only** property returns the frequency of the frame clock (LRCK), in Hz.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

The frame clock frequency is returned as a long integer value.

NB: **This frequency is discriminated rather than precisely measured; i.e. it is returned as the nearest standard sample rate.**

5.12.6.1.1.19 VSIO_BitClockInvert

Description

This property specifies whether to invert the bit clock.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

True	Bit clock is inverted.
False	Bit clock is not inverted (default).

5.12.6.1.1.20 VSIO_BitClockFreq

Description

This **read-only** property returns the frequency of the bit clock (SCK), in Hz.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

The bit clock frequency is returned as a long integer value. It is equal to the Frame clock frequency (VSIO_FrameClockFreq) times Slot length (VSIO_SlotLength) times Slots per wire (VSIO_SlotsPerWire).

NB: **Depending on the sample rate, some combinations of frame clock frequency, Slot length and Slots per wire can exceed the maximum supported multiplex bit rate of 24.576Mbps. In this case, the outputs of the VSIO Adapter are indeterminate.**

5.12.6.1.1.21 VSIO_MasterClockDir

Description

This property sets the direction of the master clock (MCK).

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

VSIO_CLOCKDIR_NONE	No master clock is produced or expected by the VSIO Adapter.
VSIO_CLOCKDIR_IN	Sets the master clock to be received from the EUT by the VSIO Adapter.
VSIO_CLOCKDIR_OUT	Sets the master clock to be driven by the VSIO Adapter into the EUT.

5.12.6.1.1.22 VSIO_MasterClockMultiplier

Description

This property specifies the master clock multiplier (the ratio of MCK frequency to the sample rate).

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

The master clock multiplier can be set to **64**, **128**, **192**, **256**, **384** or **512**.

5.12.6.1.1.23 VSIO_MasterClockFreq

Description

This **read-only** property returns the frequency of the master clock (MCK), in Hz.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

Values

The master clock frequency is returned as a long integer value. It is equal to the Frame clock frequency (VSIO_FrameClockFreq) times the Frame clock multiplier (VSIO_MasterClockMultiplier).

NB: **Depending on the sample rate, some combinations of frame clock frequency and master clock multiplier can exceed the maximum supported multiplex bit rate of 24.576Mbps. In this case, the outputs of the VSIO Adapter are indeterminate.**

5.12.6.1.1.24 VSIO_Delay

Description

This allows the timing relationship between serial clock and data wires to be adjusted in nominal 7ns steps.

NB: **This property can apply to either the Generator or the Analyzer part of the VSIO Adapter object. You must specify which one you are using by prefixing it with "VSIOAdapter.Generator." or "VSIOAdapter.Analyzer."**

For further details, please see the VSIO Adapter Control dialogue box section in the Operation manual.

Setting a non-zero "data delay" value when using standard cabling, or setting an incorrect value in any case, is very likely to result in data transmission failure. For this reason, this property should only be used by experienced operators in exceptional circumstances.

Values

If data and clocks are the same direction, the data delay is set to **0ns** and cannot be changed. Otherwise, values of **-7**, **0**, **7**, **14**, **21**, **28** or **35** can be set. These values represent the delay in ns as a short integer value.

5.12.6.1.1.25 VSIO_SPIClockPolarity

Description

This property specifies whether to invert the polarity of the SPI clock.

Values

True	SPI clock is inverted (default).
False	SPI clock is not inverted.

5.12.6.1.1.26 VSIO_SPIClockPhase

Description

This property specifies whether to delay the phase of the SPI data with respect to the SPI clock.

Values

True	SPI data is delayed (default).
False	SPI data is not delayed.

5.12.6.1.2 Methods

5.12.6.1.2.1 VSIO_SetCurrentDevice

bRet = VSIO_SetCurrentDevice (sAddress)

This method selects the current VSIO Adapter for use by a script. Subsequent properties and methods of the VSIOAdapter object will act on the device selected.

Parameters

sAddress	The address of the VSIO Adapter to select as current.

Return value

This method returns **True** if the VSIO Adapter was set successfully, or **False** if it fails. This may be because the specified device cannot be found, or is not a VSIO Adapter.

5.12.6.1.2.2 VSIO_SetGeneratorRouting

bRet = VSIO_SetGeneratorRouting (sWire, sSlot, sRouting)

This method sets the routing for a Slot on one of the generator wires.

Parameters

sWire	The number of the wire to set routing for. This can be a number from 1 to the number of wires used. Note that the number of wires used is dependent on the Slots per wire (see VSIO_SlotsPerWire).
sSlot	The Slot to set routing for. This can be a number from 1 to the number of Slots per wire (see VSIO_SlotsPerWire).
sRouting	**VSIO_ROUTING_A** to route channel A of dScope's Signal Generator to this wire/Slot; **VSIO_ROUTING_B** to route channel B of dScope's Signal Generator to this wire/Slot; **VSIO_ROUTING_OFF** to mute the output on this wire/Slot.

Return value

This method returns **True** if the generator routing was set successfully, or **False** if it fails. This may be because the wire, Slot or routing specified are invalid.

5.12.6.1.2.3 VSIO_SetAnalyzerRouting

bRet = VSIO_SetAnalyzerRouting (sChannel, sWire, sSlot)

This method selects which Slot and wire to route to one of the dScope's Analyzer channels.

Parameters

sChannel	The channel to set the routing for. This may be **CHANNEL_A** or **CHANNEL_B**.
sWire	The number of the wire to set routing for. This can be a number from 1 to the number of wires used. Note that this is dependent on the number of Slots per wire (see VSIO_SlotsPerWire).
sSlot	The Slot to set routing for. This can be a number from 1 to the number of Slots per wire (see VSIO_SlotsPerWire).

Return value

This method returns **True** if the analyzer routing was set successfully, or **False** if it fails. This may be because the channel, wire or Slot specified are invalid.

5.12.6.1.2.4 VSIO_SendSPIData

bRet = VSIO_SendSPIData (sNumChars, pTxBuf, pRxBuf)

This method sends a sequence of control bytes to the EUT, in SPI mode.

Parameters

sNumChars	The number of characters to send.
pTxBuf	The buffer of characters to send. This must be the same size as ***sNumChars***.
pRxBuf	The buffer that will be filled in with the characters received. This must be the same size as ***sNumChars***.

Return value

This method returns **True** if the SPI control data was sent successfully, or **False** otherwise.

Example

The following example shows how to send SPI control data to the EUT:

```
Dim ToSend          ' Array of control bytes to send
Dim ToReceive       ' Array of control bytes to receive
Dim ToSendSize      ' Number of bytes to send
Dim str             ' String to display return value

' Set up control data (and initialise received data)
ToSend = Array(&HA5, &H42, &H85)
ToReceive = Array(&H00, &H00, &H00)
ToSendSize = 3

' Set the current VSIO Adapter...
VSIOAdapter.VSIO_SetCurrentDevice(2)

' Send the I2C Data
If VSIOAdapter.VSIO_SendSPIData(ToSendSize, ToSend, ToReceive) Then
    Dim i
    str = "Received: "
    For i = 0 To ToSendSize - 1
        str = str & "0x"
        str = str & Hex(CByte(ToReceive(i))) & " "
    Next
    MsgBox str
End If
```

5.12.6.1.2.5 VSIO_SendI2CData

bRet = VSIO_SendI2CData (sSlaveAddress, sNumChars, pTxBuf)

This method sends a sequence of control bytes to the EUT, in I2C mode.

Parameters

sSlaveAddress	The slave address to send the I2C control data to. This must be a number between 0 and 127 (0x00 and 0x7F Hex).
sNumChars	The number of characters to send.
pTxBuf	The buffer of characters to send. This must be the same size as ***sNumChars***.

Return value

This method returns **True** if the I2C control data was sent successfully, or **False** otherwise.

Example

The following example shows how to send 2 bytes of I2C control data to the EUT:

```
Dim ToSend          ' Array of control bytes to send
Dim ToSendSize      ' Number of bytes to send

' Set up data to send...
ToSend = Array(&H01, &H3A)
ToSendSize = 2

' Set the current VSIO Adapter...
VSIOAdapter.VSIO_SetCurrentDevice(0)

' Send the I2C Data
If VSIOAdapter.VSIO_SendI2CData(&H0E, ToSendSize, ToSend) Then
    MsgBox "ACK"
Else
    MsgBox "NACK"
End If
```

5.12.6.1.2.6 VSIO_AddToArray

bRet = VSIO_AddToArray(sAddress, sBus, sGroup, strArray)

This method allows you to treat the VSIO Adapter's outputs or inputs for channel A or B as a Channel Array. This will allow them to be set as a Sweep Source.

For details on how to use Channel Arrays, see Channel Arrays.

NB: **Only a single combination of bus and group can be added to a Channel Array containing VSIO Adapter details.**

Parameters

sAddress	The address of the VSIO Adapter (can be 0 to 63).
sBus	The bus to add the relays for. This can be **VSIO_ROUTING_A** or **VSIO_ROUTING_B**.
sGroup	The group to add the relays for. This can be the Generator (**VSIO_GROUP_GENERATOR**) or Analyzer (**VSIO_GROUP_ANALYZER**).
strArray	The Channel Array to add the group to. This must be an array already defined using DSNET_DefineChannelArray.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the specified device does not exist, the sBus or sGroup parameters may be invalid, or the array may be undefined. This method also returns **False** if the array was set up as a stereo array (see DSNET_DefineChannelArray).

5.12.6.1.2.7 VSIO_AddToStereoArray

bRet = VSIO_AddToStereoArray(sAddress, sGroup, strArray)

This method allows you to treat the VSIO Adapter's outputs or inputs as a stereo Channel Array. This will allow them to be set as a Sweep Source. Use as a stereo array means that channels 1, 3, 5... etc. will be treated as channel A, and channels 2, 4, 6... etc will be treated as channel B.

For details on how to use Channel Arrays, see Channel Arrays.

Parameters

sAddress	The address of the VSIO Adapter (can be 0 to 63).
sGroup	The group to add the relays for. This can be the Generator (**VSIO_GROUP_GENERATOR**) or Analyzer (**VSIO_GROUP_ANALYZER**).
strArray	The Channel Array to add the group to. This must be an array already defined using DSNET_DefineChannelArray.

Return value

This method returns **True** if it was successful, or **False** if it fails. This may be because the specified device does not exist, the sGroup parameter may be invalid, or the array may be undefined. This method also returns **False** if the array was not set up as a stereo array (see DSNET_DefineChannelArray).

5.13 Ports

NB: This part of the dScope's scripting interface may not be available, depending on the dScope model number.

The Ports part of the dScope allows access to the PC's Serial and Parallel ports.

For information about using the Ports object to access and control Serial ports, see the Serial Ports section.

Otherwise, the following properties and methods are available. When using these from a script, they must be prefixed with **"Ports."**

Properties

There are no properties available to control the Ports object.

Methods

PORTS_WriteValue

5.13.1 Methods

5.13.1.1 PORTS_WriteValue

bRetVal = PORTS_WriteValue (sPort, sValue)

This method writes a value to one of the PC's ports.

NB: **This method is only available on Windows 98 and Windows Millennium, and not on Windows 2000 or XP. This is because Windows 2000 and XP have tighter security, and don't allow access to low-level parts of the operating system.**

Most PCs have one or two serial ports, and at least one parallel port. The addresses of these ports can be determined by accessing the "Ports" section of the your PC's Device Manager (right-click on the "My Computer" icon on your desktop).

This method can be used to write to ANY part of the PC's memory. Be VERY careful when using it, as it is possible to cause yourself severe problems by writing to the wrong part of the PC's I/O space!

Parameters

sPort	The port to write to.
sValue	The value to write to the port. This may be a value between 0 and 255 (0x00 to 0xFF in hexadecimal)

Return value

This method returns **True** if the value was correctly written, or **False** if the value was not written. This may be because the value passed was invalid.

Example

The following code snippet writes to the PC's speaker ports to generate half a second of 1kHz sine wave.

```
' Set up high & low bytes
usFreq = (1193181 / 1000)
usMSFreq = (usFreq / 256)
usLSFreq = (usFreq Mod 256)

' Write to the port
Ports.PORTS_WriteValue &H43, &HB6
Ports.PORTS_WriteValue &H42, usLSFreq
Ports.PORTS_WriteValue &H42, usMSFreq
Ports.PORTS_WriteValue &H61, &H33
Sleep(500)
Ports.PORTS_WriteValue &H61, &H30
```

5.13.2 Serial Ports

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The dScope can access any of the PC's serial ports using the Microsoft Communications ActiveX

Control (MSComm).

Under normal circumstances, it would be up to the script writer to create and use the ActiveX control themselves. However, in the case of the Communications control, it must be licensed and will only work on a PC which has Visual Basic or Visual C++ installed. For this reason, the dScope wraps the MSComm control within its own interface.

Creating and accessing Serial Ports

For each Serial port required, you must create a Serial port object. To use a serial port that has been created in this, you must set it as the current Serial port. Once you have finished using the port object, it must be deleted to allow dScope to delete any memory associated with the port.

Example

The following example shows how to create and set up two serial ports; it then outputs a stream of bytes on port 1 and reads from port 2 before closing the ports.

```
' Create Serial port object for each port
If Not Ports.PORTS_CreateSerialPort(1) Then
    MsgBox "Failed to create serial port!"
    Exit Sub
End If
If Not Ports.PORTS_CreateSerialPort(2) Then
    MsgBox "Failed to create serial port!"
    Exit Sub
End If

' Set up each serial port in turn - firstly port 1
Ports.PORTS_SetSerialPort(1)
SerialPort.SP_Settings = "9600,N,8,1"
SerialPort.SP_PortOpen = True
SerialPort.SP_InBufferCount = 0

' Serial port 2
Ports.PORTS_SetSerialPort(2)
SerialPort.SP_Settings = "19200,N,8,1"
SerialPort.SP_PortOpen = True
SerialPort.SP_InBufferCount = 0

' Set up output stream for port 1
' This will actually turn on a dS-NET I/O Switcher with address 0,
' attached to the PC's serial port 1...
Ports.PORTS_SetSerialPort(1)
sAddress = 0
sChecksum = &H55 - (sAddress + &H01 + &HFF + &H01)
strOutputMessage = Chr(&H55) & Chr(sAddress) & Chr(&H01) & Chr(&HFF) _
                   & Chr(&H01) & Chr(sChecksum And &HFF) & Chr(&HAA)
SerialPort.SP_Output = strOutputMessage

' Wait for input on port 2
Ports.PORTS_SetSerialPort(2)
StartTime = Timer()
bTimeOut  = False

' Loop until any data got
Do
   bTimeOut = ((Timer() - StartTime) > 5)
Loop Until ((SerialPort.SP_InBufferCount >= 9) Or (bTimeOut))

' Tell the user what happened
If bTimeOut Then
   MsgBox "Timed out! " & vbCrLf & "Data in buffer is " & _
          CStr(SerialPort.SP_Input)
Else
```

```
        MsgBox "Data received: " & CStr(SerialPort.SP_Input)
    End If

    ' Close the ports again
    Ports.PORTS_SetSerialPort(1)
    SerialPort.SP_PortOpen = False
    Ports.PORTS_SetSerialPort(2)
    SerialPort.SP_PortOpen = False

    ' Delete the ports, now we've finished using them
    Ports.PORTS_DeleteSerialPort(1)
    Ports.PORTS_DeleteSerialPort(2)
```

5.13.2.1 PORTS_CreateSerialPort

bRet = PORTS_CreateSerialPort(sPortNum)

This method creates a Serial Port object which can then be used to access the specified port.

NB: **The Serial port object MUST be deleted using PORTS_DeleteSerialPort after use.**

Parameters

sPortNum	Pass the number of the Serial Port that you wish to create the Serial port object for.

Return value

This method will return **True** if the Serial port exists and was successfully set as the current port, or **False** otherwise.

NB: **Creating a Serial port object will automatically set this port as the current Serial port, so a call to PORTS_SetSerialPort is not necessary immediately after this call.**

5.13.2.2 PORTS_SetSerialPort

bRet = PORTS_SetSerialPort(sPortNum)

This method sets the current Serial port object. All subsequent Serial port properties (prefixed by **SerialPort.**) will act on this port.

NB: **The Serial Port object MUST have been created using PORTS_CreateSerialPort before this method can be used.**

Parameters

sPortNum	Pass the number of the Serial Port that you wish to set as the current port.

Return value

This method will return **True** if the Serial port object has been successfully created and was set as the current port, or **False** otherwise.

NB: **Creating a Serial port object using PORTS_CreateSerialPort will automatically set this port as the current Serial port, so a call to PORTS_SetSerialPort() is not necessary immediately after creation.**

5.13.2.3 PORTS_DeleteSerialPort

PORTS_DeleteSerialPort(sPortNum)

This method deletes the Serial port object created using PORTS_CreateSerialPort.

Parameters

sPortNum	Pass the number of the Serial Port object that you wish to delete.

Return value

This method has no return value.

5.13.2.4 Properties

5.13.2.4.1 SP_Settings

Description

Sets and returns the baud rate, parity, data bit, and stop bit parameters of the Serial port.

If value is not valid, the Serial port control will generate an error when the port is opened using SP_PortOpen.

Values

The value of this property must be a string with four settings in the format "BBBB,P,D,S", where BBBB is the baud rate, P is the parity, D is the number of data bits, and S is the number of stop bits. The default value is "9600,N,8,1"

Baud rate values

The baud rate can be any baud rate supported by the serial port. Typical values are 9600, 19200, 115200, etc.

Parity values

The following are valid Parity values:

E	Even
M	Mark
N	None (default)
O	Odd
S	Space

Data bit values

The following are valid data bit values:

4	
5	
6	
7	
8	(default)

Stop bit values

The following are valid stop bit values:

1	(default)
1.5	
2	

5.13.2.4.2 SP_Handshaking

Description

Sets and returns the hardware handshaking protocol.

This property refers to the internal communications protocol by which data is transferred from the hardware port to the receive buffer. When a character of data arrives at the serial port, the communications device has to move it into the receive buffer so that your program can read it. If there is no receive buffer and your program is expected to read every character directly from the hardware, you will probably lose data because the characters can arrive very quickly.

A **handshaking protocol** ensures data is not lost due to a buffer overrun, where data arrives at the port too quickly for the communications device to move the data into the receive buffer.

Values

0	**(comNone)**	(Default) No handshaking.
1	**(comXOnXOff)**	XON/XOFF handshaking.
2	**(comRTS)**	RTS/CTS (Request To Send/Clear To Send) handshaking.
3	**(comRTSXOnXOff)**	Both Request To Send and XON/XOFF handshaking.

5.13.2.4.3 SP_PortOpen

Description

Sets and returns the state of the communications port (open or closed).

Setting this property to **True** opens the port. Setting it to **False** closes the port and clears the receive and transmit buffers.

The Serial port device must support the current values in the SP_Settings property. If the SP_Settings property contains communications settings that your hardware does not support, your hardware may not work correctly.

If either the SP_DTREnable or the SP_RTSEnable properties is set to **True** before the port is opened, the properties are set to **False** when the port is closed. Otherwise, the DTR and RTS lines remain in their previous state.

Values

True	Port is opened.
False	Port is closed.

5.13.2.4.4 SP_InBufferSize

Description

Sets and returns the size of the receive buffer in bytes.

This property refers to the total size of the receive buffer. The default size is 1024 bytes. This should not be confused with the SP_InBufferCount property which reflects the number of characters currently waiting in the receive buffer.

NB: Note that the larger you make the receive buffer, the less memory you have available to your application. However, if your buffer is too small, it runs the risk of overflowing unless handshaking is used. As a general rule, start with a buffer size of 1024 bytes. If an overflow error occurs, increase the buffer size to handle your application's transmission rate.

Values

The input buffer size is represented as a long integer value.

5.13.2.4.5 SP_InBufferCount

Description

Returns the number of characters waiting in the receive buffer.

This property refers to the number of characters that have been received by the modem and are waiting in the receive buffer for you to take them out. You can clear the receive buffer by setting this property to 0.

NB: Do not confuse this property with the SP_InBufferSize property. The SP_InBufferSize property reflects the total size of the receive buffer.

Values

The input buffer count is represented as a long integer value.

5.13.2.4.6 SP_InputLen

Description

Sets and returns the number of characters the SP_Input property reads from the receive buffer.

The default value for this property is 0. Setting SP_InputLen to 0 causes the Serial port to read the entire contents of the receive buffer when SP_Input is used. If SP_InputLen characters are not available in the receive buffer, the SP_Input property returns a zero-length string (""). The user can optionally check the SP_InBufferCount property to determine if the required number of characters are present before using SP_Input.

This property is useful when reading data from a machine whose output is formatted in fixed-length blocks of data.

Values

The input length is represented as a long integer value.

5.13.2.4.7 SP_Input

Description

Returns and removes a stream of data from the receive buffer.

The SP_InputLen property determines the number of characters that are read by this property. Setting SP_InputLen to 0 causes this property to read the entire contents of the receive buffer.

Values

The SP_InputMode property determines the type of data that is retrieved with the SP_Input property. If SP_InputMode is set to 0 (text) then the SP_Input property returns text data in a Variant. If SP_InputMode is 1 (binary) then the SP_Input property returns binary data in an array of bytes in a Variant.

5.13.2.4.8 SP_NullDiscard

Description

Determines whether null characters are transferred from the port to the receive buffer. A null character is defined as ASCII character 0, Chr(0).

Values

True	Null characters are *not* transferred from the port to the receive buffer.
False	(Default) Null characters are transferred from the port to the receive buffer.

5.13.2.4.9 SP_OutBufferSize

Description

Sets and returns the size of the transmit buffer in bytes.

This property refers to the total size of the transmit buffer. The default size is 512 bytes. Do not confuse this property with the SP_OutBufferCount which reflects the number of bytes currently waiting in the transmit buffer.

NB: The larger you make the transmit buffer, the less memory you have available to your application. However, if your buffer is too small, you run the risk of overflowing unless you use handshaking. As a general rule, start with a buffer size of 512 bytes. If an overflow error occurs, increase the buffer size to handle your application's transmission rate.

Values

The output buffer size is represented as a long integer value.

5.13.2.4.10 SP_OutBufferCount

Description

Returns the number of characters waiting in the transmit buffer. You can also use this property to clear the transmit buffer by setting it to 0.

NB: Do not confuse this property with the SP_OutBufferSize property which reflects the total size of the transmit buffer.

Values

The output buffer size is represented as a long integer value.

5.13.2.4.11 SP_Output

Description

Writes a stream of data to the transmit buffer.

Values

This property can transmit text data or binary data. To send text data using this property, you must specify a Variant that contains a string. To send binary data, you must pass a Variant which contains a byte array.

Normally, if you are sending an ANSI string to an application, you can send it as text data. If you have data that contains embedded control characters, Null characters, etc., then you will want to pass it as binary data.

5.13.2.4.12 SP_CommEvent

Description

This read-only property returns the most recent communication event or error.

The Serial port object does not fire events to the script, so if errors or events need to be detected then this property must be checked after performing relevant operations.

Error Values

The following error value scan be returned:

1001	**(comEventBreak)**	A Break signal was received.
1004	**(comEventFrame)**	Framing Error. The hardware detected a framing error.
1006	**(comEventOverrun)**	Port Overrun. A character was not read from the hardware before the next character arrived and was lost.
1008	**(comEventRxOver)**	Receive Buffer Overflow. There is no room in the receive buffer.
1009	**(comEventRxParity)**	Parity Error. The hardware detected a parity error.
1010	**(comEventTxFull)**	Transmit Buffer Full. The transmit buffer was full while trying to queue a character.
1011	**(comEventDCB)**	Unexpected error retrieving Device Control Block (DCB) for the port.

Communication Event Values

The following values are noted when certain events occur:

1	**(comEventSend)**	There are fewer than SP_SThreshold number of characters in the transmit buffer.
2	**(comEventReceive)**	Received SP_RThreshold number of characters. This event is generated continuously until you use the Input property to remove the data from the receive buffer.
3	**(comEventCTS)**	Change in Clear To Send line.
4	**(comEventDSR)**	Change in Data Set Ready line. This event is only fired when DSR changes from 1 to 0.
5	**(comEventCD)**	Change in Carrier Detect line.
6	**(comEventRing)**	Ring detected. Some UARTs (universal asynchronous receiver-transmitters) may not support this event.
7	**(comEventEOF)**	End Of File (ASCII character 26) character received.

5.13.2.4.13 SP_Break

Description

Sets or clears the break signal state.

When set to **True**, the Break property sends a break signal. The break signal suspends character transmission and places the transmission line in a break state until you set the Break property to **False**.

Typically, you set the break state for a short interval of time, and only if the device with which you are communicating requires that a break signal be set.

Values

True	Sets the break signal state.
False	Clears the break signal state.

5.13.2.4.14 SP_CDHolding

Description

Determines whether the carrier is present by querying the state of the Carrier Detect (CD) line. Carrier Detect is a signal sent from a modem to the attached computer to indicate that the modem is online.

NB: It is especially important to trap a loss of the carrier in a host application, such as a bulletin board, because the caller can hang up (drop the carrier) at any time.

The Carrier Detect is also known as the Receive Line Signal Detect (RLSD).

Values

True	Carrier Detect line is high.
False	Carrier Detect line is low.

5.13.2.4.15 SP_CTSHolding

Description

Determines whether you can send data by querying the state of the Clear To Send (CTS) line. Typically, the Clear To Send signal is sent from a modem to the attached computer to indicate that transmission can proceed.

The Clear To Send line is used in RTS/CTS (Request To Send/Clear To Send) hardware handshaking. The SP_CTSHolding property gives you a way to manually poll the Clear To Send line if you need to determine its state.

Values

True	Clear to Send line is high.
False	Clear to Send line is low.

5.13.2.4.16 SP_ParityReplace

Description

Sets and returns the character that replaces an invalid character in the data stream when a parity error occurs.

The parity bit refers to a bit that is transmitted along with a specified number of data bits to provide a small amount of error checking. When you use a parity bit, the Serial port control adds up all the bits that are set (having a value of 1) in the data and tests the sum as being odd or even (according to the parity setting used when the port was opened).

By default, the control uses a question mark (?) character for replacing invalid characters. Setting

SP_ParityReplace to an empty string ("") disables replacement of the character where the parity error occurs. The SP_CommEvent property is set to comEventRXParity.

Values

The SP_ParityReplace character is used in a byte-oriented operation, and must be a single-byte character. You can specify any ANSI character code with a value from 0 to 255.

5.13.2.4.17 SP_DSRHolding

Description

Determines the state of the Data Set Ready (DSR) line. Typically, the Data Set Ready signal is sent by a modem to its attached computer to indicate that it is ready to operate.

This property is useful when writing a Data Set Ready/Data Terminal Ready handshaking routine for a Data Terminal Equipment (DTE) machine.

Values

True	Data Set Ready line is high.
False	Data Set Ready line is low.

5.13.2.4.18 SP_RTSEnable

Description

Determines whether to enable the Request To Send (RTS) line. Typically, the Request To Send signal that requests permission to transmit data is sent from a computer to its attached modem.

When this property is set to **True**, the Request To Send line is set to high (on) when the port is opened, and low (off) when the port is closed.
The Request To Send line is used in RTS/CTS hardware handshaking. The SP_RTSEnable property allows you to manually poll the Request To Send line if you need to determine its state.

For more information on handshaking protocols, see the SP_Handshaking property.

Values

True	Enables the Request To Send line.
False	(Default) Disables the Request To Send line.

5.13.2.4.19 SP_DTREnable

Description

Determines whether to enable the Data Terminal Ready (DTR) line during communications. Typically, the Data Terminal Ready signal is sent by a computer to its modem to indicate that the computer is ready to accept incoming transmission.

When SP_DTREnable is set to True, the Data Terminal Ready line is set to high (on) when the port is opened, and low (off) when the port is closed. When SP_DTREnable is set to False, the Data

Terminal Ready always remains low.
NB: In most cases, setting the Data Terminal Ready line to low hangs up the telephone.

Values

True	Enable the Data Terminal Ready line.
False	(Default) Disable the Data Terminal Ready line.

5.13.2.4.20 SP_EOFEnable

Description

This property determines if the Serial port looks for End Of File (EOF) characters during input. If an EOF character is found, the input will stop and the SP_CommEvent property will be set to set to comEventEOF.

When this property is set to **False**, the control will not scan the input stream for EOF characters.

Values

True	Input stops when an EOF character is found.
False	(Default) Input continues when an EOF character is found.

5.13.2.4.21 SP_InputMode

Description

Sets or returns the type of data retrieved by the SP_Input property. The data will either be retrieved as string or as binary data in a byte array.

Use **comInputModeText** for data that uses the ANSI character set. Use **comInputModeBinary** for all other data such as data that has embedded control characters, Nulls, etc.

Values

0	**(comInputModeText)**	(Default) Data is retrieved through the SP_Input property as text.
1	**(comInputModeBinary)**	Data is retrieved through the SP_Input property as binary data.

5.14 User-defined tables

Various parts of the dScope software allow entry of user-defined parameters that are basically lists or tables of data. For example: a Generator wavetable is a table of sample values; an FFT Window function is a table of gain factors, and a Sweep data table is a table of Sweep source values.

All of these tables can be created using the dScope's scripting system. This allows simple construction of tables by simply writing a script to fill in values in the table. Most of the table-creation script types work in a similar way, but certain specific features of each table have their own individual script functions which are listed under the different reference sections.

In general, a script can create a table using the following steps:
1) Initialise the table (by giving it a file name, and a size)
2) Set up specific parameters (for example, units for a Sweep data table)
3) Set each point in the table.

Step (3) may simply write a list of points to the table, or (for example, in the case of Generator wavetables) may be involve some kind of function used to calculate the values in the table.

The following sections are references of how to write user-defined scripts to create tables for the various parts of the dScope:

Generator wavetable reference
FFT Detector Weighting filter reference
FFT Window function reference
Sweep data table reference

Trace Limit Tables are similar to other types of user-defined table, but differ in some important ways, They are covered as part of the Trace Window, under Limit Table reference.

In a script, all properties and methods from this section must be prefixed with **"UserTable."**

NB:

Writing a script to fill in a table will probably result in two separate files - the script file, and the actual table file itself.
In most places in the dScope system, these can be used interchangeably - the script file can be loaded in place of the table, and rather than directly copy the file's data into memory, the script will run and fill in the memory. This has the advantage that the script can query other parts of the dScope system, but the disadvantage that it is much slower.

For example, let's say we write a script ("My Wavetable.dss") that generates a user-defined wavetable called "My Wavetable.wfm".
In future sessions, you could use *either* of these two files in the Signal Generator "User Wavetable" field - the ".wfm" file will load more quickly, but the ".dss" file will run the script when selected, so that it could find out (for example) the current Digital Output frame rate as it runs, using this information to change the way it generates the wavetable.

5.14.1 Standard Methods

5.14.1.1 USR_InitTable

bRetVal = USR_InitTable (strFileName, lNumPoints)

This method *must* be called first when creating any user-defined table, and will initialize the table ready to write values into.

NB:

If this method is called more than once from within a script, any details of a previous table will be *deleted*. If you wanted to keep the previous details, you must ensure that the USR_SaveTable method has been used to save the previous table before initializing the table again.

Parameters

strFileName	This parameter specifies the name of the table file that should be created. If a full file and path are not specified, the system will attempt to create the file by appending the default file extension for the table type, and using the default folder for this type of file based on the script type creating the file (see Types of dScope script for further information).
lNumPoints	The number of points or values that the table will contain.

NB: **If a file name is specified, then the user-defined table will be saved to this file name when the script finishes running (unless the USR_SaveTable method is used directly by the script to save the table, or the script is not the correct type for this user-defined table)**

Return value

This method returns **True** if the table initialisation completed successfully, or **False** if it failed. This may be because the file name is invalid, or the memory for the table cannot be allocated.

5.14.1.2 USR_SetValue

bRetVal = USR_SetValue (dValue)

This method writes a value to the next available position in the data table. The first position is slot 0, and the last valid position is (lNumPoints - 1), where lNumPoints is the parameter passed to USR_InitTable.

Repeated calls to this method will put values into subsequent slots in the data table.

Parameters

dValue	The value to write to the data table.

Return value

This method returns **True** if the value was correctly written, or **False** if the value was not written. This may be because the value itself was invalid, or you may have tried to set more values than the table can contain (as specified using USR_InitTable).

5.14.1.3 USR_SetValueAt

bRetVal = USR_SetValueAt (lPos, dValue)

This method writes a value to the specified position in the data table. The first position is slot 0, and the last valid position is (lNumPoints - 1), where lNumPoints is the parameter passed to USR_InitTable.

Parameters

lPos	The position in the data table to write to.
dValue	The value to write to the data table.

Return value

This method returns **True** if the value was correctly written, or **False** if the value was not written. This may be because the value itself was invalid, or the position is not within the size of the data table (as specified using USR_InitTable).

5.14.1.4 USR_SetValues

bRetVal = USR_SetValues (lNumValues, dValue)

This method writes a series of values to the next ***lNumValues*** positions in the data table. The first position in the table is slot 0, and the last valid position is (lNumPoints - 1), where lNumPoints is the parameter passed to USR_InitTable.

Repeated calls to this method will put values into subsequent slots in the data table.

Parameters

lNumValues	The number of entries (of value ***dValue***) to write to the data table.
dValue	The value to write to the data table.

Return value

This method returns **True** if the values were correctly written, or **False** if the values were not written. This may be because the value itself was invalid, or you may have tried to set more values than the table can contain (as specified using USR_InitTable).

5.14.1.5 USR_SetValuesAt

bRetVal = USR_SetValueAt (lStartPos, lEndPos, dValue)

This method writes a series of values to all positions in the data table between those specified. The first position in the table is slot 0, and the last valid position is (lNumPoints - 1), where lNumPoints is the parameter passed to USR_InitTable.

Parameters

lStartPos	The first position in the data table to write to.
lEndPos	The last position in the data table to write to.
dValue	The value to write to the data table.

Return value

This method returns **True** if the values were correctly written, or **False** if they were not written. This may be because the value itself was invalid, or the positions do not fall within the size of the data table (as specified using USR_InitTable).

5.14.1.6 USR_SaveTable

bRetVal = USR_SaveTable (strFileName)

This method saves the user-defined table to the specified file name.

The USR_SaveTable method is only necessary if the user-defined table is being created in a script that is not a user-defined table script (see Types of dScope script) or when more than one table is created in a single script. Usually, a user-defined table is created from a specific type of script; in this case, the table is saved automatically when the script finishes running. However, there may be occasions when it is necessary to create tables from a different script type (for example an Automation script), or when you may wish to create more than one table from the same script. In this case, the same code is used to initialise the table (USR_InitTable) and add/remove points (USR_SetValue, etc); However this USR_SaveTable method must then be used to actually save the table.

NB: **If this method is used from the correct type of user-defined script, with the same filename passed to USR_InitTable, then the table will no longer be saved automatically when the script finishes running.**

Parameters

strFileName	The name of the table file that should be created. Any valid file name can be used, enclosed in double quotation marks ("..."). If a full path name is not specified, then the system will create a file in the folder specified in the Options dialogue box for the relevant type of user-defined table (see OPT_WavetablesFolder, OPT_FFTWindowsFolder, OPT_WeightingFiltersFolder or OPT_DataTablesFolder). If necessary, the system will automatically append the file extension for the relevant type of user-defined table.

Return value

This method returns **True** if the table was saved successfully, or **False** if it failed for some reason. This may be because the dScope cannot open the file with the name specified.

NB: **This method will be ignored unless the user-defined table has been initialised using the USR_InitTable method.**

5.14.2 Generator wavetable reference

Table values

Generator wavetables consist of a table of 24-bit or 48-bit sample values. They can be entered as double-precision floating-point numbers, but each sample value will be truncated to 24 or 48 bits before storage, since this is the size of the sample values used in the generated wavetable.

Other details

The other details needed for a user-defined wavetable are:

- Whether the sample values should be stored as 24-bit or 48-bit (this will usually depend on whether the wavetable is being generated for digital or analogue analysis).
- Whether the table should be used directly as a table of sample values, or whether it should utilize the amplitude entered on the Signal Generator panel to adjust the level of the sample values.
- If the amplitude from the Signal Generator is to be used, the difference between the amplitude entered and the maximum sample value from the table.

NB: **If you change a Generator wavetable, the new wavetable must be re-loaded into the Signal Generator panel before the new wavetable will be used.**

Methods

The following methods are available for use with generator wavetables:

USR_InitTable
USR_SetMaxAmpl
USR_SetAmplitudeUse
USR_GetGeneratorChannel
USR_SetValue
USR_SetValueAt
USR_SetValues
USR_SetValuesAt
USR_SetDefaultAmplitude
USR_SetDefaultAmplitudeUnit
USR_SetPseudoCrestFactor
USR_MinimizeCrestFactor

Example

The following example creates a simple burst script, with a number of periods of sine wave at a high amplitude followed by a number of periods at a lower amplitude:

```
' TYPE             Generator wavetable
' DESCRIPTION      Generates a 'burst' script

' *** Declarations ***
Option Explicit        ' Must declare vars before using

Dim lNumSamples        ' No. samples in buffer
Dim lNumSamplesOn      ' No. samples in 'burst' part
Dim sChannel           ' Channel we're generating for
Dim bOn                ' Whether gen is ON for channel
Dim l                  ' Loop var
Dim dSample            ' Value of current sample
Dim phi                ' Phase
Dim prc                ' Used to calc sample value
Dim dFreq              ' Frequency of burst part
Dim dAmplBurst         ' Ampl of the 'burst' part of
                       ' waveform, in dBFS
Dim dAmplSpace         ' Amplitude of the 'space' part
                       ' of waveform, in dBFS
Dim fs                 ' Output frame rate
Dim iBurstPeriods      ' Number of periods in 'burst'
Dim iSpacePeriods      ' Number of ms in the 'space'
Dim PI
Dim strFileName        ' File to create
```

```
' *** Main body of script ***

' Variables you can change to affect Burst
dAmplBurst      = -18.0
dAmplSpace      = -28.0
dFreq           = 1000.0
iBurstPeriods   = 25
iSpacePeriods   = 40

' Other variables
PI          = 4 * Atn(1)        ' PI
fs          = 48000.0           ' Current output fs
prc         = (2 * PI * dFreq) / fs
strFileName = "Burst.wfm"

' Total No. samples is the No. of samples in
' the No. of periods specified, plus the No. of
' samples in the Space period
lNumSamples = ((iBurstPeriods * fs) / dFreq) + _
   ((iSpacePeriods * fs) / dFreq)

' Convert amplitudes from dBFS to factors
dAmplBurst = 10 ^ (dAmplBurst / 20)
dAmplSpace = 10 ^ (dAmplSpace / 20)

' How many samples are 'On'?
lNumSamplesOn = (iBurstPeriods * fs) / dFreq

' Initialise the user-defined table
If Not UserTable.USR_InitTable(strFileName, _
lNumSamples) Then
    MsgBox "Failed to create wavetable"
    Automation.AUT_StopScript()
End If

' Max value we're entering is 1.0
UserTable.USR_SetMaxAmpl(1.0)

' Create the waveform
For l = 0 To lNumSamples - 1

    ' Phase info
    phi = (2 * PI * l) / fs

    ' Get sample value
    If l < lNumSamplesOn Then
        dSample = Sin(l * prc + phi) * dAmplBurst
    Else
        dSample = Sin(l * prc + phi) * dAmplSpace
    End If

    ' Write the sample To the buffer
    If Not UserTable.USR_SetValue(dSample) Then
        MsgBox "Failed to write to user table"
          Automation.AUT_StopScript()
    End If

Next
```

NB: **You can cut and paste examples like this from the help file into the dScope Script Edit window.**

5.14.2.1 Methods

5.14.2.1.1 USR_SetMaxAmpl

bRetVal = USR_SetMaxAmpl (dMaxAmpl)

When writing a Generator wavetable script, the dScope converts values to gains between 0.0 and 1.0 to store in the generated table. Setting the maximum amplitude simply tells the script what value will equate to a gain value of 1.0.

For example, if you want to define your wavetable using sample values from 0x000000 to 0x7FFFFF, then you would call

```
UserTable.USR_SetMaxAmpl(&H7FFFFF)
```

Parameters

dMaxAmpl	This parameter specifies the maximum amplitude to use for the table.

Return value

This method returns **True** if the amplitude was set correctly, or **False** if it failed for some reason.

NB: **This method must be called BEFORE any values have been written to the table.**

5.14.2.1.2 USR_SetAmplUse

bRetVal = USR_SetAmplUse (bUseAmpl)

This method allows you to specify whether the values in the Generator wavetable should be treated as sample values, or whether the user can alter the amplitude of the generated table by using the Signal Generator's amplitude field (SG_ChAAmpl or SG_ChBAmpl).

Parameters

False	Use the sample values as they have been entered, i.e. ignore the Signal Generator amplitude value.
True	Use the Signal Generator amplitude value to alter the amplitude of the wavetable. The difference between the amplitude entered and the maximum sample value in the table is defined using USR_SetPseudoCrestFactor.

Return value

This method returns **True** if the amplitude use was set correctly, or **False** if it failed for some reason. This may be because an invalid parameter was passed.

5.14.2.1.3 USR_GetGeneratorChannel

sChannel = USR_GetGeneratorChannel()

This method allows the script to determine which of the generator channels the script is currently running for. This allows a single script to be run, which can take different action depending on which channel's signal is currently being generated.

Parameters

This method has no parameters.

Return value

This method returns **CHANNEL_A** (0) if the script is currently being run from channel A of the generator, or **CHANNEL_B** (1) if being run from channel B.

5.14.2.1.4 USR_SetDefaultAmpl

bRetVal = USR_SetDefaultAmpl (dDefaultAmpl)

This method allows a Generator wavetable script to set up the default Signal Generator amplitude when it is run. This simply means that the script writer can choose a sensible starting amplitude, which will be reflected in the Signal Generator settings. The user can then override this amplitude if needed by altering the Signal Generator's amplitude using SG_ChAAmpl or SG_ChBAmpl.

The default amplitude is specified in the unit specified by USR_SetDefaultAmplUnit.

Parameters

dDefaultAmpl	A double-precision value specifying the default amplitude to use for the Signal Generator.

Return value

This method returns **True** if the default amplitude was set correctly, or **False** if it failed for some reason. This may be because an invalid amplitude was passed for the current unit (specified by USR_SetDefaultAmplUnit).

NB: **This method is ignored unless the Amplitude Use has been set to True (see USR_SetAmplUse).**

5.14.2.1.5 USR_SetDefaultAmplUnit

bRetVal = USR_SetDefaultAmplUnit (sDefaultAmplUnit)

This method specifies the unit that the default Signal Generator amplitude (set using USR_SetDefaultAmpl) is entered in.

See USR_SetDefaultAmpl for a description of the default amplitude.

Parameters

sDefaultAmplUnit	Specifies the unit that the default amplitude is entered in. See Units below for a list of valid units.

Return value

This method returns **True** if the default amplitude was set correctly, or **False** if it failed for some reason. This may be because an invalid amplitude was passed for the current unit (specified by USR_SetDefaultAmplUnit).

NB: **This method is ignored unless the Amplitude Use has been set to True (see USR_SetAmplUse).**

Units

The following units are valid for the default amplitude:

UNIT_DBFS	Sets the default amplitude unit to dBFS.
UNIT_PERCENTFS	Sets the default amplitude unit to %FS (percentage of full scale).
UNIT_FFS	Sets the default amplitude unit to FFS (fraction of full scale).
UNIT_HEX	Sets the default amplitude unit to Hex.
UNIT_VRMS	Sets the default amplitude unit to an RMS voltage.
UNIT_VP	Sets the default amplitude unit to a peak voltage.
UNIT_VPP	Sets the default amplitude unit to a peak-to-peak voltage.
UNIT_DBU	Sets the default amplitude unit to dBu.
UNIT_DBV	Sets the default amplitude unit to dBV.
UNIT_DBM	Sets the default amplitude unit to dBm.
UNIT_W	Sets the default amplitude unit to W.
UNIT_DBSPL	Sets the default amplitude unit to dBSPL.

5.14.2.1.6 USR_SetPseudoCrestFactor

bRetVal = USR_SetPseudoCrestFactor (dPseudoCrestFactor)

When using a generated wavetable in the Signal Generator (See SG_ChAUserWaveform or SG_ChBUserWaveform), the dScope allows you to use the Signal Generator amplitude to adjust the amplitude of the generated table.

If you are defining a signal with more than one tone, you must specify a "pseudo-crest factor" for the wavetable - a value representing the ratio between the maximum sample value and the amplitude that will be entered in the Signal Generator (SG_ChAAmpl or SG_ChBAmpl).

This "pseudo-crest factor" is specified as the ratio of the maximum possible sample value (usually 1.0), to the maximum sample value in the generated buffer.

For example, if you specify a single-tone waveform, then the pseudo-crest factor would probably be 1.0 (i.e. a full-scale amplitude of 0dBFS entered in the Signal Generator would adjust the table so that the maximum sample value is at 0x7FFFFF or 1.0).

However, if you create a multi-tone signal, where the combination of tones means that the maximum sample value is at 0x200000 (0.25), then you would calculate the pseudo crest factor as 1.0 / 0.25, i. e. 4.00.

NB: **This property is ignored unless you have specified that the wavetable should use the Signal Generator amplitude (see USR_SetAmplUse).**

Parameters

dPseudoCrestFactor	A double-precision value specifying the pseudo-crest factor for this wavetable. See above for a description of the Pseudo-Crest Factor and how to calculate it.

Return value

This method returns **True** if the pseudo-crest factor was set correctly, or **False** if it failed. This may be because this command is being used in a script that is not a Generator wavetable script.

5.14.2.1.7 USR_MinimizeCrestFactor

bRetVal = USR_MinimizeCrestFactor (paFrequencies, paPhaseOffsets, sNumTries, bRandomize, lBufferSize, dSampleRate)

When writing a multi-tone Generator wavetable script, the signal will include several tones at different frequencies. The phase offset of these frequencies may be important, as it is possible (particularly with linearly-separated tones) to end up with a signal with a huge Crest Factor, i.e. ratio of peak-sample to RMS value of the tone.

The dScope provides this method to allow you to pass it a list of the tone frequencies, and have it return the phase offsets that will result in the minimum crest factor for a signal containing these tones.

The minimization of the crest factor can only find a best-guess solution, and there are two methods it can use:

1) Random

This method involves picking random phases for each frequency, and calculating the crest factor. This is repeated the specified number of times, and at the end, the best result is the one returned.

2) Slot-in

This method gives the lowest frequency tone a phase offset of 0, and puts it into the buffer. It then takes each of the other frequencies in turn, and finds the best position for this tone within the buffer. It then adds this frequency to the buffer, and goes on to the next tone.

The random method is more likely to find a better solution eventually, but may require a large number of tries. The slot-in method will find a pretty good result fairly quickly, but may miss some better options because once it has placed a frequency, it does not go back to it later.

Parameters

paFrequencies	An array of frequencies that will constitute this signal. Each value in this array must be a double-precision value.
paPhaseOffsets	The array of phase offsets that will be returned. These phase offsets will be the number of bins to offset the corresponding frequency by.
sNumTries	The number of tries to use for the randomization process, or the step

	interval to use when using the slot-in method. For either method, the larger the number, the better the result will be, but the longer it will take.
bRandomize	**True** to use the random method, **False** to use the slot-in method. See above for descriptions of each of these methods of crest factor minimization.
lBufferSize	The size of the buffer containing the multi-tone signal, in samples.
dSampleRate	The sample rate for which this multi-tone is being created.

Return value

This method returns **True** if the crest factor was minimized successfully, or **False** if it failed.

Example

The following code gives an example of how to call this method, based on an array of tones.

```
' TYPE           Generator wavetable
' DESCRIPTION    Generates the wavetable for a multi-tone
'                signal
' AUTHOR         Generated automatically by dScope software
'                (c) Prism Sound Ltd, 2002
'
' *** Declarations ***
Dim Tones(11)
Dim Freqs(11)
Dim Phase(11)
PI            = 4 * Atn(1)
dFirstFreq    = 20.51
dLastFreq     = 20000.98
iSamplingFreq = 48000
iTones        = 12
bLog          = True
dAmpl         = 0.10000000
iSamples      = 16384
bDigital      = True
bCrosstalk    = False
eChannel      = UserTable.USR_GetGeneratorChannel()
dMaxValue     = 1.00
sFileName     = "Multi-tone"
dMaxSample    = 0.00
bMinCrestFact = False
bRandomize    = False
bAutoCalcAmpl = False

' We ensure that all tones are in even OR odd bins.
iRem = 0
If bCrosstalk Then
   If (eChannel = CHANNEL_B) Then
      iRem = 1
      sFileName = sFileName & " (ChB)"
   Else
      sFileName = sFileName & " (ChA)"
   End If
End If

' Calc locations of tones
f = dFirstFreq
If bLog Then
   If iTones > 1 Then
      g = (dLastFreq / dFirstFreq) ^ (1 / (iTones-1))
   For i = 0 To (iTones-1)
      Tones(i) = ShiftBin(Int((iSamples * SyncFreq(f) /
                          iSamplingFreq) + 0.5) - 1, iRem)
      Freqs(i) = (iSamplingFreq * Tones(i)) / iSamples
```

```
      Phase(i) = 0.0
      f = f * g
   Next
Else
   If iTones > 1 Then
      g = (dLastFreq - dFirstFreq) / (iTones-1)
   For i = 0 To (iTones-1)
      Tones(i) = ShiftBin(Int((iSamples * SyncFreq(f) /
                          iSamplingFreq) + 0.5) - 1, iRem)
      Freqs(i) = (iSamplingFreq * Tones(i)) / iSamples
      Phase(i) = 0.0
      f = f + g
   Next
End If

' Initialise the table
UserTable.USR_InitTable sFileName, iSamples

' Set Max Amplitude to 1.0
UserTable.USR_SetMaxAmpl(dMaxValue)

' Ensure that the generator amplitude will be used
UserTable.USR_SetAmplUse(True)

' Set the default amplitude and unit
UserTable.USR_SetDefaultAmpl(-20.00)
UserTable.USR_SetDefaultAmplUnit(UNIT_DBFS)

' Minimize the crest factor
If bMinCrestFact Then

    ' THIS IS THE CALL TO MINIMIZE THE CREST FACTOR
   dMaxCrestFact = UserTable.USR_MinimizeCrestFactor (Freqs,
      Phase, 10, bRandomize, iSamples, iSamplingFreq)

   If (bRandomize And bAutoCalcAmpl) Then
      dAmpl = (dMaxValue / dMaxCrestFact)
      ' Re-set the default amplitude and unit
      UserTable.USR_SetDefaultAmpl(dAmpl)
      UserTable.USR_SetDefaultAmplUnit(UNIT_FFS)
   End If
End If

' Calc the sine components
dP = (2 * PI) / iSamplingFreq
For i = 0 To (iSamples-1)

   ' Calc the value
   dValue = 0.0
   For j = 0 To (iTones-1)
         ' THIS LINE USES THE PHASE OFFSET RETURNED FOR
      ' THIS FREQUENCY
      dValue = dValue + (dAmpl *
         Sin((i + Phase(j)) * dP * Freqs(j)))
   Next

   ' Note max value so far
   If Abs(dValue) > dMaxSample Then
      dMaxSample = Abs(dValue)
   End If

   ' Set the value in the table
   UserTable.USR_SetValue(dValue)

Next

' Set the pseudo crest factor (i.e. the ratio of peak sample
' to the amplitude of a tone)
UserTable.USR_SetPseudoCrestFactor(dMaxSample / dAmpl)
```

```
Function SyncFreq(dFreq)
   ' Calc cycles that fit into one buffer
   iCycles = Int(((dFreq * iSamples) / iSamplingFreq) + 0.5)
   ' Disallow DC
   If iCycles = 0 Then
      iCycles = 1
   End If
   ' Use integer val to find new frequency
   SyncFreq = (iSamplingFreq * iCycles) / iSamples
End Function

Function ShiftBin (iVal, iRem)
   ' If odd, shift up a bin
   If ((iVal Mod 2) <> iRem) Then
      If (iVal = ((iSamples / 2) - 1)) Then
         ShiftBin = iVal-1
      Else
         ShiftBin = iVal+1
      End If
   Else
      ShiftBin = iVal
   End If
End Function
```

5.14.3 FFT Detector Weighting Filter reference

Table values

FFT Detector Weighting filters are simply a table of gain values, usually from 0.0 to 1.0 (although some filters can include gain factors greater than 1.0). They are entered as double-precision floating-point numbers.

When an FFT buffer is captured, each FFT Detector must apply its filters (including the weighting filter) to the buffer. To do this, it simply goes through the FFT buffer and applies the gain factors from the weighting filter buffer.

NB: **The number of gain values in the weighting filter buffer must be a multiple of, or be exactly divisible by, the number of points in the FFT buffer.**

For example, if the number of FFT points is 4k (4096), and the weighting filter has only 1k (1024) points, then the dScope will apply each value in the weighting filter buffer to 4 consecutive points of the FFT buffer.

On the other hand, if the number of points in the weighting filter is MORE than the number of FFT points, then the dScope will miss out points in the weighting filter when applying the gain factors, ensuring that the FFT buffer and weighting filter start and end at the same place.

NB: **If you change a Weighting filter, the new filter must be re-loaded into the FFT Detector panel before the new filter will be used.**

Methods

The following methods are available for use with FFT Detector weighting filters:

USR_InitTable
USR_SetValue
USR_SetValueAt
USR_SetValues
USR_SetValuesAt

Example

The following example creates a simple brick wall weighting filter, for use at an input sample rate of 96kHz:

```
' TYPE          FFT Detector Weighting filter
' DESCRIPTION   Ideal brick wall filter, with a corner
'               frequency at 20kHz.
'               For use at a sample rate of 96K.
'               Note that this weighting filter will have
'               to be re-selected every time the FFT
'               length is changed.

' *** Declarations ***
Option Explicit         ' Declare vars before using them.
Dim iBufferLength       ' Length of FFT buffer
Dim sFileName           ' File name of filter to create
Dim iCutoffPoint        ' Corner frequency
Dim iIndex               ' Loop variable

' Access the Buffer Length
iBufferLength = FFTParameters.FFTP_NumPoints / 2

' Set the file name
sFileName = "Brick Wall20_96_new.wtg"

' Initialise the table
UserTable.USR_InitTable ".\" & sFilename, iBufferLength

' Where's the cutoff point
iCutoffPoint = iBufferLength * 20000 / 48000

' Create filter - gain of 1.0 up to cutoff point,
' then 0.0 afterwards...
UserTable.USR_SetValuesAt 0, iCutoffPoint, 1.0
UserTable.USR_SetValuesAt iCutoffPoint+1, iBufferLength-1, 0.0
```

5.14.4 FFT Window Function reference

Table values

FFT Window functions are simply a table of gain values from 0.0 to 1.0, which are applied to the sample buffer before it is processed to create the FFT buffer. Window function gain values are entered as double-precision floating-point numbers.

These gain factors need to be applied to the sample buffer, whose length is the same as the number of FFT points.

NB: **If you change a Window function, the new Window function must be re-loaded into the FFT Parameters panel before the new Window function will be used.**

Methods

The following methods are available for use with FFT Window functions:

USR_InitTable
USR_SetValue
USR_SetValueAt
USR_SetValues
USR_SetValuesAt
USR_SetWindowWidth

Example

The following script shows how to create a simple Triangular Window function.

```
' TYPE          FFT Window
' DESCRIPTION   Simple triangular Window function


' *** Declarations ***
Option Explicit     ' Must declare vars before using
Dim iFFTLength      ' Points for FFT
Dim dBinMiddle      ' Value at middle of bin
Dim iIndex          ' Loop variable
Dim sFileName       ' Name of file to create

' *** Main body of script ***

' Set the file name
sFileName = "Triangular window"

' Get the length of the buffer
iFFTLength = FFTParameters.FFTP_NumPoints

' Initialise the table
UserTable.USR_InitTable sFileName, iFFTLength

' Set the notch width
UserTable.USR_SetWindowWidth(10)

' Fill in each bin...
For iIndex = 0 To ((iFFTLength / 2) - 1)

    ' Get Bin Middle
    dBinMiddle = iIndex + 0.5

    ' Set value for this bin
    UserTable.USR_SetValueAt iIndex, _
       (2 * dBinMiddle / iFFTLength)
    UserTable.USR_SetValueAt ((iFFTLength - 1) - iIndex), _
       (2 * dBinMiddle / iFFTLength)

Next
```

5.14.4.1 Methods

5.14.4.1.1 USR_SetWindowWidth

bRetVal = USR_SetWindowWidth (sWindowWidth)

One of the options for the width of an FFT Detector's band pass or band reject filter (see FFTD_BPBRBandwidth) is "notch". This means that the filter ONLY filters out (or in) the bins that constitute the peak of the FFT signal, and no other bins. To do this, the dScope uses its knowledge of the currently selected Window function to determine how many bins the peak covers.

Obviously if the Window function is user-defined, the dScope has no way of knowing this window width, and so the writer of a Window function script must specify the window width to be stored as part of the table. This method allows the user to do just that.

The easiest way of calculating the notch width for a Window function is to create the Window function without using this function, then select the Window function in the FFT Parameters. Look at the FFT on the Trace window, and use the cursor to count the number of bins in the notch. Then insert a call to this method in the script, and run it again to re-create the Window function with a notch of the correct width.

NB: **The window width is not necessary unless you will be using the "notch" filter width on an FFT Detector. It has no effect on the FFT display of the Trace window.**

Parameters

sWindowWidth The window width to use for this Window function data table, in bins.

Return value

This method returns **True** if the window width was successfully written, or **False** if the call failed. This may be because the method is used on a table that is not an FFT Detector Window function.

5.14.5 Sweep data table reference

Table values

Sweep data tables allow specification of a series of points to use for the X-axis (source) of a Sweep. Rather than just allow the software to choose a linear of logarithmic series of points, a data table can be used to specify only those points that are needed.

When creating a Sweep data table, the user must firstly decide which of the "types" of Sweep the data table is replacing - for example, generator frequency, or carrier jitter amplitude. Then the unit must be specified, and the points subsequently added are assumed to be in that unit.

NB: **If you change a Sweep data table, the new data table must be re-loaded into the Sweep Setup panel before the new table will be used.**

Methods

The following methods are available for use with Sweep data tables:

USR_InitTable
USR_SetValue
USR_SetValueAt
USR_SetValues
USR_SetValuesAt
USR_SetSweepSource
USR_SetSweepSourceUnit

Example

The following script shows how to create a simple data table that uses specific frequencies, rather than an entire range where many of the frequencies are irrelevant.

```
' TYPE                Sweep data table
' DESCRIPTION         Data table containing specific
'                     frequencies

' *** Declarations ***
Option Explicit        ' Must declare vars before using
Dim strFileName        ' File name of table to be created
Dim iNumPoints         ' Number of points in the table
```

```
' Set up the file name
strFileName = "frequency data table"

' Initialise the number of points
iNumPoints = 9

' *** Main body of script ***

' Create the table with three points
If Not UserTable.USR_InitTable(strFileName, _
iNumPoints) Then
    MsgBox "Failed to create data table!"
End If

' Set the Sweep source to be generator frequency
' (both channels), with a unit of Hz
UserTable.USR_SetSweepSource(SW_SOURCE_GENFREQ_BOTH)
UserTable.USR_SetSweepSourceUnit(UNIT_FREQ_HZ)

' Set the points in the table.
' Firstly, some low frequencies....
UserTable.USR_SetValue(20.0)
UserTable.USR_SetValue(30.0)
UserTable.USR_SetValue(40.0)

' Then some frequencies around 1kHz...
UserTable.USR_SetValue(990.0)
UserTable.USR_SetValue(1000.0)
UserTable.USR_SetValue(1010.0)

' ...then some higher frequencies
UserTable.USR_SetValue(10000.0)
UserTable.USR_SetValue(15000.0)
UserTable.USR_SetValue(20000.0)
```

NB: **The values must be written into the Sweep Data table in consecutive order (either increasing or decreasing), otherwise the drawing of the Sweep trace may not work correctly.**

5.14.5.1 Methods

5.14.5.1.1 USR_SetSweepSource

bRetVal = USR_SetSweepSource (sSweepSource)

This method allows the writer of a Sweep data table to specify which Sweep source the table will emulate.

Parameters

sSweepSource	The Sweep source to use for this data table. It can be any of the values listed in Sweep Sources, below.

Return value

This method returns **True** if the Sweep source was successfully written, or **False** if the call failed. This may be because the value passed is invalid, or because the method is used in a script that is not a Sweep data table script.

Sweep sources

The ***sSweepSource*** parameter can have any of the following values:

SW_SOURCE_GENFREQ_CHA	Sets the data table to contain generator frequencies for channel A.
SW_SOURCE_GENFREQ_CHB	Sets the data table to contain generator frequencies for channel B.
SW_SOURCE_GENFREQ_BOTH	Sets the data table to contain generator frequencies for both channels.
SW_SOURCE_GENAMPL_CHA	Sets the data table to contain generator amplitudes for channel A.
SW_SOURCE_GENAMPL_CHB	Sets the data table to contain generator amplitudes for channel B.
SW_SOURCE_GENAMPL_BOTH	Sets the data table to contain generator amplitudes for both channels.
SW_SOURCE_GENDCOFFSET	Sets the data table to contain Digital Outputs DC offset values.
SW_SOURCE_JITTERFREQ	Sets the data table to contain Digital Output Carrier jitter frequency values.
SW_SOURCE_JITTERAMPL	Sets the data table to contain Digital Output Carrier jitter amplitude values.
SW_SOURCE_CTD_BPBRFREQ	Sets the data table to contain Continuous-Time Detector band pass/band reject frequency values.
SW_SOURCE_FFTD_BPBRFREQ	Sets the data table to contain FFT Detector band pass/band reject frequency values.

NB: **This method must be called BEFORE any values have been written to the table.**

5.14.5.1.2 USR_SetSweepSourceUnit

bRetVal = USR_SetSweepSourceUnit (sUnit)

This method allows the writer of a Sweep data table to specify which unit the Sweep source values are specified in.

Parameters

sUnit	The unit to use for the specified Sweep source (specified using USR_SetSweepSource). It can be any of the values listed under Units, below.

Return value

This method returns **True** if the Sweep source unit was successfully written, or **False** if the call failed. This may be because the value passed is invalid, or because the method is used in a script that is not a Sweep data table script.

Units

The allowed values for the Sweep source unit depend on the Sweep source selected using USR_SetSweepSource.

If the Sweep source is set up to be a frequency Sweep (**SW_SOURCE_GENFREQ_CHA**, **SW_SOURCE_GENFREQ_CHB**, **SW_SOURCE_GENFREQ_BOTH**, **SW_SOURCE_JITTERFREQ**, **SW_SOURCE_CTD_BPBRFREQ** or **SW_SOURCE_FFTD_BPBRFREQ)** then the only allowed unit is **UNIT_FREQ_HZ**. (This unit is selected by default for these Sweep sources and does not need to be set explicitly).

If the Sweep source is set up to be the generated amplitude (**SW_SOURCE_GENAMPL_CHA**, **SW_SOURCE_GENAMPL_CHB** or **SW_SOURCE_GENAMPL_BOTH**) then the following units are allowed:

UNIT_DBFS	Sets the data table source unit to dBFS.
UNIT_PERCENTFS	Sets the data table source unit to %FS (percentage of full scale).
UNIT_FFS	Sets the data table source unit to FFS (fraction of full scale).
UNIT_HEX	Sets the data table source unit to Hex.
UNIT_VRMS	Sets the data table source unit to an RMS voltage.
UNIT_VP	Sets the data table source unit to a peak voltage.
UNIT_VPP	Sets the data table source unit to a peak-to-peak voltage.
UNIT_DBU	Sets the data table source unit to dBu.
UNIT_DBV	Sets the data table source unit to dBV.
UNIT_DBM	Sets the data table source unit to dBm.
UNIT_W	Sets the data table source unit to W.
UNIT_DBSPL	Sets the data table source unit to dBSPL.

If the Sweep source is set to **SW_SOURCE_GENDCOFFSET**, then the following units are allowed:

UNIT_DBFS	Sets the data table source unit to dBFS.
UNIT_PERCENTFS	Sets the data table source unit to %FS (percentage of full scale).
UNIT_FFS	Sets the data table source unit to FFS (fraction of full scale).
UNIT_HEX	Sets the data table source unit to Hex.

If the Sweep source is set to **SW_SOURCE_JITTERAMPL**, then the following units are allowed:

UNIT_JITTER_NS	Sets the data table source unit to ns.
UNIT_JITTER_UI	Sets the data table source unit to UI.

NB: **This method must be called BEFORE any values have been written to the table.**

5.15 Events

NB: **This part of the dScope's scripting interface may not be available, depending on the dScope model number.**

The dScope software has certain events, as defined in the Event Manager, that can be set up to take certain action when the event occurs. For example, the system can be set to write to a log file whenever a Reading's lower or upper limit is breached.

This section of the dScope reference contains details of all the events that can be fired to a script, and the event handler subroutines that are needed to handle them.

Methods

StartTimer
EndTimer
FireEvent

Events

Event_ChAValidBit
Event_ChBValidBit
Event_CarrierInputLocking
Event_CarrierBiphase
Event_CarrierBlockLength
Event_CarrierEyeNarrowing
Event_CarrierAsync
Event_ChannelCheckFailed_ChA
Event_ChannelCheckFailed_ChB
Event_CS_ProfBit
Event_CS_CopyrightBit
Event_CS_Emphasis
Event_CS_ChannelMode
Event_CS_CRCError
Event_CS_ANotEqualToB
Event_Trigger
Event_BufferProcessed
Event_ReadingMinLimit
Event_ReadingMaxLimit
Event_TraceMinLimit
Event_TraceMaxLimit
Event_SweepStarted
Event_SweepStepDone
Event_SweepFinished
Event_SweepSense
Event_Timer
Event_Keypress
Event_Scripted

5.15.1 Methods

5.15.1.1 StartTimer

lTimerID = StartTimer (lNumMilliseconds)

This method starts a timer that will go off at intervals of the specified number of milliseconds. When this timer goes off, the software will fire a "Timer" event, to which it will pass the timer ID returned from the StartTimer function.

NB: **All timers that are started must be ended with EndTimer. Otherwise, the timers will still exist even after the script has finished running.**

Parameters

lNumMilliseconds	Specifies the timer interval, in milliseconds. Note that this is only accurate to around the nearest 50ms.

Return value

The timer ID used to reference this timer in the script. This value should be stored for use later on, when stopping the timer, or can be used to differentiate between different timers in the "Timer" event.

5.15.1.2 EndTimer

EndTimer (lTimerID)

This method stops the specified timer, preventing it causing any more timer events.

Parameters

lTimerID	The ID of the timer to stop (This value is the timer ID returned by the StartTimer function).

Return value

This method has no return value.

5.15.1.3 FireEvent

FireEvent (lParam)

This method allows the user to fire an event from within a script. This can be useful if the script is event-driven and an event needs to be triggered when some automation procedure is finished.

This method causes the Scripted event to be fired.

Parameters

lParam	This is a user-defined long integer parameter that will be passed as a parameter to the Scripted event. It can be any valid long integer value (from -2,147,483,648 to 2,147,483,647)

Return value

This method has no return value.

5.15.2 Events

5.15.2.1 Event_ChAValidBit

Sub Event_ChAValidBit ()
...
End Sub

This event handler is called when the channel A Valid bit (see DI_ChAValid) changes state.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.2 Event_ChBValidBit

Sub Event_ChBValidBit ()
...
End Sub

This event handler is called when the channel B Valid bit (see DI_ChBValid) changes state.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.3 Event_CarrierInputLocking

Sub Event_CarrierInputLocking ()
...
End Sub

This event handler is called when the locked/unlocked state of the Digital Input Carrier changes (see DI_InputUnlocked).

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.4 Event_CarrierBiphase

Sub Event_CarrierBiphase()
...
End Sub

This event handler is called when the state of the biphase violation flag on the Digital Input changes (see DI_BiphaseViolation).

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.5 Event_CarrierBlockLength

Sub Event_CarrierBlockLength()
...
End Sub

This event handler is called when the state of the block length error flag on the Digital Input changes (see DI_BlockLengthError).

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.6 Event_CarrierEyeNarrowing

Sub Event_CarrierEyeNarrowing()
...
End Sub

This event handler is called when the state of the eye-narrowing error flag on the Digital Input changes (see DI_EyeNarrowing).

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.7 Event_CarrierAsync

Sub Event_CarrierAsync()
...
End Sub

This event handler is called when the state of the asynchronous w.r.t. generator error flag on the Digital Input changes (see DI_Asynchronous).

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.8 Event_ChAChannelCheckFailed

Sub Event_ChAChannelCheckFailed()
...
End Sub

NB: **Prior to V1.30, this event's name was Event_ChannelCheckFailed_ChA. For legacy reasons, this old event name will work from internal dScope scripts.**

This event handler is called when the state of channel A's Channel Check failure flag on the Digital Input changes (see DI_ChannelCheckFailedChA).

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.9 Event_ChBChannelCheckFailed

Sub Event_ChBChannelCheckFailed()
...
End Sub

NB: **Prior to V1.30, this event's name was Event_ChannelCheckFailed_ChB. For legacy reasons, this old event name will work from internal dScope scripts.**

This event handler is called when the state of channel B's Channel Check failure flag on the Digital Input changes (see DI_ChannelCheckFailedChB).

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.10 Event_CSProfBit

Sub Event_CSProfBit()
...
End Sub

NB: **Prior to V1.30, this event's name was Event_CS_ProfBit. For legacy reasons, this old event name will work from internal dScope scripts.**

This event handler is called when the state of the Input Channel Status Professional/Consumer bit changes.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.11 Event_CSCopyrightBit

Sub Event_CSCopyrightBit()
...
End Sub

NB: **Prior to V1.30, this event's name was Event_CS_CopyrightBit. For legacy reasons, this old event name will work from internal dScope scripts.**

This event handler is called when the state of the Consumer Input Channel Status copyright bit changes.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.12 Event_CSEmphasis

Sub Event_CSEmphasis()
...
End Sub

NB: **Prior to V1.30, this event's name was Event_CS_Emphasis. For legacy reasons, this old event name will work from internal dScope scripts.**

This event handler is called when the state of the Input Channel Status emphasis bits changes.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.13 Event_CSChannelMode

Sub Event_CSChannelMode()
...
End Sub

NB: **Prior to V1.30, this event's name was Event_CS_ChannelMode. For legacy reasons, this old event name will work from internal dScope scripts.**

This event handler is called when the state of the Professional Input Channel Status channel mode bits changes.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.14 Event_CSCRCError

Sub Event_CSCRCError()
...
End Sub

NB: **Prior to V1.30, this event's name was Event_CS_CRCError. For legacy reasons, this old event name will work from internal dScope scripts.**

This event handler is called when the state of the Professional Input Channel Status CRC error flag changes.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.15 Event_CSANotEqualToB

Sub Event_CSANotEqualToB()
...
End Sub

NB: **Prior to V1.30, this event's name was Event_CS_ANotEqualToB. For legacy reasons, this old event name will work from internal dScope scripts.**

This event handler is called when the Input Channel Status changes so that the channels are not equal.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.16 Event_Trigger

Sub Event_Trigger()
...
End Sub

This event handler is called when the trigger goes off and the sample buffer is captured.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.17 Event_BufferProcessed

Sub Event_BufferProcessed()
...
End Sub

This event handler is called when the FFT calculation has been performed on the captured sample buffer.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.18 Event_ReadingMinLimit

Sub Event_ReadingMinLimit (lParam)
...
End Sub

This event is called when a Reading's lower limit is breached.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

lParam	This is a long integer, which is related to the Reading that fired the event. The Reading that it represents can be set as the current Reading using SetCurrentReadingFromEventParam.

Return value

This event handler has no return value.

5.15.2.19 Event_ReadingMaxLimit

Sub Event_ReadingMaxLimit (lParam)
...
End Sub

This event is called when a Reading's upper limit is breached.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

lParam	This is a long integer, which is related to the Reading that fired the event. The Reading that it represents can be set as the current Reading using SetCurrentReadingFromEventParam.

Return value

This event handler has no return value.

5.15.2.20 Event_TraceMinLimit

Sub Event_TraceMinLimit (lParam)
...
End Sub

This event is called when a Trace's lower Limit Line is breached.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

lParam	This is a long integer, which is related to the Trace that fired the event. The Trace that it represents can be set as the current Trace using TW_SetCurrentTraceFromEventParam.

NB: **The lParam is NOT the same as a Trace ID, and should not be used as such.**

Return value

This event handler has no return value.

5.15.2.21 Event_TraceMaxLimit

Sub Event_TraceMaxLimit (lParam)
...
End Sub

This event is called when a Trace's upper Limit Line is breached.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

lParam	This is a long integer, which is related to the Trace that fired the event. The Trace that it represents can be set as the current Trace using TW_SetCurrentTraceFromEventParam.

NB: **The lParam is NOT the same as a Trace ID, and should not be used as such.**

Return value

This event handler has no return value.

5.15.2.22 Event_SweepStarted

Sub Event_SweepStarted()
...
End Sub

This event handler is called when a Sweep is started.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.23 Event_SweepStepDone

Sub Event_SweepStepDone()
...
End Sub

This event handler is called when a Sweep step has completed.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.24 Event_SweepFinished

Sub Event_SweepFinished()
...
End Sub

This event handler is called when a Sweep has finished.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.25 Event_SweepSense

Sub Event_SweepSense()
...
End Sub

This event handler is called when a Sweep with a sense source (see SW_SweepSource) has sensed a new data point.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.26 Event_Timer

Sub Event_Timer (lTimerID)
...
End Sub

This event is called when any timer (started by StartTimer()) goes off.

The script can have any number of timers set up, and all will call this routine when the timer goes off. The lTimerID parameter will denote which timer it is.

NB: **The Windows timers used by the dScope scripting only have an accuracy of around 50ms.**

Parameters

lTimerID	The ID of the timer that has gone off. This will be the value returned by StartTimer. NB: Timers exist across scripts, so this event may be fired with a timer ID that has been created for a different script. For this reasson, it is important to check the value of lTimerID agains the timer ID returned by StartTimer.

Return value

This event handler has no return value.

5.15.2.27 Event_Keypress

Sub Event_Keypress()
...
End Sub

This event handler is called when the key set up to fire events (F2) has been pressed.

NB: **For this event to be fired, the Event Manager must be on (see EM_On) and this event must be turned on.**

Parameters

This event handler has no parameters.

Return value

This event handler has no return value.

5.15.2.28 Event_Scripted

Sub Event_Scripted (lParam)
...
End Sub

This event is called when the "scripted" event is fired, i.e. an event that is called manually by calling FireEvent(...) from a script.

Parameters

lParam	This is a long integer, which is the user-defined parameter passed to FireEvent(...).

Return value

This event handler has no return value.

5.16 dScope Application

This section of the dScope reference contains details of properties and methods pertaining the the dScope application as a whole, i.e. general items that are not specific to a particular area of the system.

Methods

Display
SetPage
LoadConfiguration
SaveConfiguration
GetConfiguration
CloseApplication
Sleep
GetSecurityLevel
GetSoftwareVersion
IsInitialised
MsgBoxWithTimeOut
LastResultSettled
ShowHelpTopic

Properties

ShowMessages
ModelNumber
ShowUserBar

5.16.1 Properties

5.16.1.1 ShowMessages

Description

This property allows you to specify whether the dScope should show error messages when a script attempts to set a property to an invalid value or passes an invalid parameter to a method. By default, error messages are shown; however there may be circumstances where you want to use success or failure to determine whether a parameter set by the script is correct.

For example, you could see whether a specific soundcard is available on the system by using the line

```
SI_Soundcard = "My soundcard"
```

If the property is then successfully set to this value, then the soundcard must exist; otherwise, it does not. However, you would not want a script to display an error message if this command failed, so you could turn off showing of error messages around the call, as follows:

```
ShowMessages = False
SI_Soundcard = "My soundcard"
If SI_Soundcard = "My soundcard" Then
    ' Success
Else
    ' Failure
End If
ShowMessages = True
```

NB: **Care must be taken when using this function. Turning off this property may result in the script not working correctly, without any indication of the reasons for failure!**

Values

True	Error messages will be displayed when a script tries to set a property to an invalid value, or tries to pass invalid parameters to a method
False	No error messages will be shown.

NB: **If the dScope window is hidden (using DISPLAY_HIDE), this will also stop any error messages being displayed by the software.**

5.16.1.2 ModelNumber

Description

This **read-only** property allows the writer of a script to determine which dScope hardware model number is currently installed. Not all dScope features are available for all Model numbers. For full details, see Model numbers in the Operation manual.

Values

MODELNUMBER_ANALOGUE	Indicates that the hardware defines the dScope Series IIIA (analogue only) set of features.
MODELNUMBER_ANALOGUEPLUS	Indicates that the hardware defines the dScope Series IIIA+ (analogue-plus) set of features.
MODELNUMBER_ANALOGUEANDDIGITAL	Indicates that the hardware defines the full set of features (analogue and digital).

5.16.1.3 ShowUserBar

Description

This property allows you to specify whether the User-defined button bar should be displayed or not. The details of the User-defined button bar are loaded from the Registry before it is shown.

Values

True	Show the User bar, after reloading its details from the Registry.
False	Hide the User bar.

5.16.2 Methods

5.16.2.1 Display

Display (sDisplay)

This method allows the dScope main window to be hidden, minimized or maximized from a script.

Parameters

sDisplay	This parameter specifies how the dScope should be displayed. It can have one of the values listed in the Display options section below.

NB: If the dScope window is hidden (using **DISPLAY_HIDE**), this will also stop any error messages being displayed by the software.
If this parameter is used, then the return values from all functions should be checked to see whether the function has succeeded.

Return value

This method has no return value.

Display options

The ***sDisplay*** parameter can have one of the following values:

DISPLAY_SHOW	Shows the dScope main window. The dScope will be shown by default, and this value will probably only be necessary to show the window after it has been hidden
DISPLAY_HIDE	Hides the dScope main window.
DISPLAY_MINIMIZED	Minimizes the dScope main window. This is the best option to use if you want the main window to be out of the way, as it will still display any error messages in the dScope application caused by the script.
DISPLAY_MAXIMIZED	Maximizes the dScope main window (increases it to the same size as the Windows desktop).

5.16.2.2 SetPage

SetPage (sPage)

This method sets the current Page of the dScope display, as shown by the Page tabs in the bottom right hand corner of the dScope Status bar.

Parameters

sPage	The Page to set. It must be a valid Page number from 1 to 5.

Return value

This method has no return value.

NB: **This method will fail if there is currently a Print Preview or Export Preview window open.**

5.16.2.3 LoadConfiguration

bRet = LoadConfiguration (strFileName)

This method loads the specified Configuration file.

Parameters

strFileName File name of the file to load, enclosed in double quotation marks ("...").

If a full path name is specified, the system will look for this exact file.

If a file name only is specified, then the system will look in the "Configurations" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default).

If necessary, the system will automatically append the correct filename extension (".dsc" for Configuration files).

Return value

This method returns **True** if the file loaded successfully, or **False** if it failed.
This may happen if the file does not exist or is corrupted.

5.16.2.4 SaveConfiguration

bRet = SaveConfiguration (strFileName)

This method saves the current Configuration to the specified file name.

Parameters

strFileName	File name to save this Configuration to, enclosed in double quotation marks ("..."). If a full path name is specified, the system will save this exact file. If a file name only is specified, then the system will save the file in the "Configurations" subfolder of the folder containing the dScope program files (installed to "C:\Program Files\Prism Sound\dScope Series III" by default). If you don't specify the file extension (".dsc" for Configuration files), the system will automatically append it.

Return value

This method returns **True** if the Configuration was saved successfully, or **False** if it failed.
This may happen if the file path specified is invalid.

5.16.2.5 GetConfiguration

str = GetConfiguration (bFullPath)

This method returns the file name or full path of the currently loaded Configuration file.

Parameters

bFullPath	**True** to return the full path name of the Configuration, or **False** to return just the file name (e.g. "~Default.dsc").

Return value

This method returns the file name or full path name of the currently loaded Configuration, or "" if no Configuration is currently loaded.

5.16.2.6 CloseApplication

CloseApplication ()

This method closes the dScope application. It is the equivalent of clicking on the X button in the top right-hand corner of the dScope main window.

NB: **This function should NOT be used when running a script from within dScope. It is designed for use when controlling the dScope externally, for example from a C ++ or Delphi program. When the external program runs, it can start up the dScope software; the CloseApplication function allows the dScope software to be closed when the external program exits.**

Parameters

This method has no parameters.

Return value

This method has no return value.

5.16.2.7 Sleep

Sleep (lNumMilliseconds)

This method causes the script to wait for the specified time, in the meantime allowing other threads to run (for example, the FFT calculation thread, or background thread reading data from the hardware).

If the script simply sits in a loop to wait (for example, in a `while` loop), then Windows will keep the script thread running continuously and so the dScope software will not perform any necessary background processing.

The **Sleep** function ensures that other parts of the dScope software are able to run and collect data from the hardware.

Parameters

lNumMilliseconds	This specifies the time to wait, in milliseconds. Note that this time is approximate, as Windows will wait for *at least* this amount of time before returning control to the script. A value of 0 can be used to simply ask other threads to process before returning. NB: The Windows timer is only accurate to around the nearest 50ms.

Return value

This method has no return value.

5.16.2.8 GetSecurityLevel

sLevel = GetSecurityLevel()

This method returns the security level of the user currently logged on.

This method will typically be used in scripts which involve recalibration of the software and other advanced features, and should not be necessary in normal use.

Parameters

This method has no parameters.

Return value

The security level of the user currently logged on.

5.16.2.9 GetSoftwareVersion

GetSoftwareVersion(ucMajor, ucMinor, ucLetter)

This method obtains the version number of the software currently running. It can be used to check whether a certain feature is available to the scripting, based on which version of the software it was introduced in.

Parameters

ucMajor	Pass a variable that will be set to the major version number.
ucMinor	Pass a variable that will be set to the minor version number.
ucLetter	Pass a variable that will be set to the ASCII character of the version letter.

For example, if the current software version is "1.03a", then calling this method will result in the ucMajor variable being set to 1, the ucMinor variable being set to 3, and the ucLetter variable being set to 97, which is the ASCII character for a lower-case 'a'.

Return value

This method has no return value.

5.16.2.10 IsInitialised

bRet = IsInitialised ()

When the dScope software runs, it has a certain amount of initialization to perform. This method ensures that when running the dScope from an external application, this initialization gets done before the external program continues.

For example, from C++:

```
while (!dscope.IsInitialised()) {
   // Keep looping until it's initialised...
} // End (while)
```

NB: **Note that it is NOT necessary to call this function when running a VBScript from within the dScope. It is only used when calling the dScope externally.**

Parameters

This method has no parameters.

Return value

This method returns **False** while the software is not initialised, and **True** once it is.

5.16.2.11 MsgBoxWithTimeOut

sRet = MsgBoxWithTimeOut (strMsg, lTimeout, sButtons, strTitle)

This method displays a standard message box. The message box is displayed for a certain amount of time and then closes automatically.

Clicking on a button on the message box will close the message box before the specified time has elapsed.

Parameters

strMsg	Message string to display, up to 1024 characters. If strMsg consists of more than one line, you can separate the lines using a carriage return character (**Chr**(13)), a linefeed character (**Chr**(10)), or carriage return–linefeed character combination (**Chr**(13) & **Chr**(10)) between each line.
lTimeout	The time to display the message box for, in milliseconds. Note that if the user clicks a button before this period is up, the message box will be closed. The Windows timer is only accurate to around the nearest 50ms.
sButtons	Value specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. See the Button values section for allowed values.
strTitle	String displayed in the title bar of the dialogue box.

Return value

This function returns the button that was pressed by the user, or 0 if no button was pressed and the timeout expired.
See Return values for possible return values.

Button values

The **sButtons** argument settings are:

Constant	Description
vbOKOnly	Display **OK** button only
vbOKCancel	Display **OK** and **Cancel** buttons.
vbAbortRetryIgnore	Display **Abort**, **Retry**, and **Ignore** buttons.
vbYesNoCancel	Display **Yes**, **No**, and **Cancel** buttons.
vbYesNo	Display **Yes** and **No** buttons.
vbRetryCancel	Display **Retry** and **Cancel** buttons.
vbCritical	Display Critical Message icon.
vbQuestion	Display Warning Query icon.
vbExclamation	Display Warning Message icon.
vbInformation	Display Information Message icon.
vbDefaultButton1	First button is default.
vbDefaultButton2	Second button is default.
vbDefaultButton3	Third button is default.
vbDefaultButton4	Fourth button is default.
vbApplicationModal	Application modal; the user must respond to the message box before continuing work in the current application.

vbSystemModal	System modal; all applications are suspended until the user responds to the message box.

The first group of values (**vbOkOnly** to **vbRetryCancel**) describes the number and type of buttons displayed in the dialogue box; the second group (**vbCritical** to **vbInformation**) describes the icon style; the third group (**vbDefaultButton1** to **vbDefaultButton4**) determines which button is the default; and the fourth group (**vbApplicationModal** and **vbSystemModal**) determines the modality of the message box. When adding numbers to create a final value for the argument **sButtons**, use only one number from each group.

Return values

Value	Description
0	No button was pressed; the message box closed because the timeout expired.
vbOK	**OK** button was pressed.
vbCancel	**Cancel** button was pressed.
vbAbort	**Abort** button was pressed.
vbRetry	**Retry** button was pressed.
vbIgnore	**Ignore** button was pressed.
vbYes	**Yes** button was pressed.
vbNo	**No** button was pressed.

5.16.2.12 LastResultSettled

bSettled = LastResultSettled ()

When a Result is read using automation, this method can be called to see whether the Result had settled or not.

Note that settling depends on a number of factors:

If the dScope Options have been set up to "Use settling from scripts", then the script will actually follow settings entered under "Settling" available from the Sweep menu.
In this case, the script may read a certain number of Results, and will perform some sort of tolerance calculation on it.
If not, then the system will simply read one Result and it is up to the script to determine whether or not the Result has settled. In this case LastResultSettled will always return **True**.

Parameters

This method has no parameters.

Return value

This method returns **True** if the last Result settled, or **False** if it timed out.

5.16.2.13 ShowHelpTopic

ShowHelpTopic (strHelpFile, strHelpTopic)

This method opens a help file, and displays the specified help topic from it.

Parameters

strHelpFile

File name of the help file, enclosed in double quotation marks ("...").

If a full path name is specified, the system will look for this exact file.

If a file name only is specified, then the system will look in the dScope program folder (installed to "C:\Program Files\Prism Sound\dScope Series III" by default).

If necessary, the system will automatically append the correct filename extension (".chm" for Help files).

strHelpTopic

The name of the help topic to display. This will usually have the extension ". htm".

NB: You can open the help topic at a specific "Anchor" point by appending "#Anchor" to the help topic name, where "Anchor" is the name of the anchor or bookmark in the topic.

Return value

This method has no return value.

5.16.2.14 IsHardwareMissing

bRet = IsHardwareMissing()

If the dScope hardware is turned off while the software is still running, and the OPT_WaitForMissingHardware option has been set, then this method will return **True**. Otherwise, while communication with the hardware is established, it will return **False**.

NB: **If the dScope was STARTED without hardware connected, this method will return False. It will only return True if the hardware has been successfully started, and the software has subsequently lost communication with it (for example, if the hardware is unplugged while the software is still running).**

Parameters

This method has no parameters.

Return value

This method returns **True** if communication with the hardware has been lost, or **False** otherwise.

5.16.2.15 ConfigHasUnsupportedSettings

bRet = ConfigHasUnsupportedSettings()

When a Configuration is loaded that has been saved on hardware with a different Model Number, there may be settings in the Configuration that cannot be applied for the current hardware. For example:

- Digital Inputs are selected for Analysis, but the Configuration is loaded while using IIIA ('Analogue Only') hardware;
- A Configuration contains more than two FFT Detectors, but is loaded on dScope IIIE ('Essentials') hardware
- A Signal Generator function is specified which is not supported on IIIA ('Analogue Only') hardware;

For full details of what is supported on different hardware, see Model Numbers in the Operation manual.

If there are settings or Desktop windows that cannot be displayed for the current Model Number, then this property will return **True**. Usually, this will also have displayed an error message to the user, or displayed a Warning in the status bar.

Parameters

This method has no parameters.

Return value

This method returns **True** if any settings in the Configuration could not be applied, or **False** if the Configuration has loaded successfully.

Part 6

Common scripting tasks

6 Common scripting tasks

This section describes some common scripting tasks that can be performed by using other applications or features built into the Windows operating system:

Automation of Microsoft Word
Automation of Microsoft Excel
Automation of Microsoft Access
Writing to a file
Printing

6.1 Automation of Microsoft Word

The following automation fragment shows how a script can write simple details to a Word document from a script.

This code fragment can be copy-and-pasted into your own scripts in the Script Edit window.

```
' Create the Word object
Set WordDocument = CreateObject("Word.Application")

' Make word visible
WordDocument.Application.Visible = True

' Create a new document
Set oDoc = WordDocument.Documents.Add

' Output the text to word
WordDocument.Selection.Font.Size = 12
WordDocument.Selection.TypeText
      "PRISM SOUND DSCOPE SERIES III TEST RESULTS"
WordDocument.Selection.TypeText vbCrLf & vbCrLf
WordDocument.Selection.Font.Size = 10
WordDocument.Selection.TypeText
      "Enter your results here..."

' Save the document.
WordDocument.ActiveDocument.SaveAs "C:\WordTest.doc"

' Close Word with the Quit method
WordDocument.Application.Quit
```

6.2 Automation of Microsoft Excel

The following automation fragment shows how a script can write simple details to an Excel spreadsheet.

This code fragment can be copy-and-pasted into your own scripts in the Script Edit window.

```
' during initialization...
Dim ExcelSheet
Set ExcelSheet = CreateObject("Excel.Sheet")

' Make Excel visible through the Application Object...
ExcelSheet.Application.Visible = True

' Place some text in the first cell of the sheet...
```

```
ExcelSheet.ActiveSheet.Cells(1,1).Value =
            CStr(SignalAnalyzer.SA_ChAFreq)

' when finished, save the sheet...
ExcelSheet.SaveAs "C:\TEST.XLS"

' and close Excel with the Quit method on
' the Application Object
ExcelSheet.Application.Quit
```

6.3 Automation of Microsoft Access

The following automation fragment shows how a script can write simple details to an Access database from a script.

This brief example creates a new record at the end of the "Calibration" table in the database "dS3Test", then enters the ChA frequency Result in the "Frequency" field of the record. The database, table and record structure must already exist.

Note that for Office 97 and later, you will need to access the specific object "DAO.DBEngine.36" which is the actual name for the database engine; it no longer works using a version-independent "DAO.DBEngine".

This code fragment can be copy-and-pasted into your own scripts in the Script Edit window.

```
' During initialization...
Dim wrkJet          ' Jet Database Engine Workspace
                    ' object
Dim dbsTest         ' Database object
Dim rstCalibration ' Table object

' Create Jet database Workspace...
' Note : For Office 97 and later, you may need to use
' the following line instead:
' Set wrkJet = CreateObject('DAO.DBEngine.36')
Set wrkJet = CreateObject("DAO.DBEngine")

' open database...
Set dbsTest = wrkJet.OpenDatabase("c:\dS3Test.mdb")

' open record set, 'Calibration' table...
Set rstCalibration = dbsTest.OpenRecordSet("Calibration")

' Go to the last record (with this method you can
' examine data en route)...
rstCalibration.MoveFirst      ' Start at first record..
While Not(rstCalibration.EOF) ' Search whole table
      rstCalibration.MoveNext    ' Next record
Wend

' put the data in the last record...
rstCalibration.Fields("Frequency") =
      SignalAnalyzer.SA_ChAFreq
rstCalibration.Update

' when data updates are complete...
rstCalibration.Close
dbsTest.Close
```

6.4 Writing to a file

The following automation script fragment is intended to show how simple details can be written to a text file from a script.

This code fragment can be copy-and-pasted into your own scripts in the Script Edit window.

```
' During initialization...
Dim fso, MyFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile = fso.CreateTextFile("c:\testfile", True)

' Writing the data...
MyFile.WriteLine("The A-channel RMS amplitude is " &
      CStr(SignalAnalyzer.SA_ChARMSAmpl))

' When writing data is finished...
MyFile.Close
```

6.5 Printing

The following automation fragment shows how a script can perform simple printing using the LPT port on your computer.

This code fragment can be copy-and-pasted into your own scripts in the Script Edit window.

```
' during initialization...
Dim fso, MyFile
Const ForWriting = 2
Const FormFeed = 12
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile = fso.OpenTextFile("LPT1", ForWriting)

' Actual printing of the data...
MyFile.Write("The A-channel RMS amplitude is " &
      CStr(SignalAnalyzer.SA_ChARMSAmpl))

' Ejecting the paper...
Myfile.Write(Chr(FormFeed))

' when printing is finished...
MyFile.Close
```

Part 7

The ScriptDlg ActiveX control

7 The ScriptDlg ActiveX control

The **ScriptDlg** ActiveX control has been designed to improve the user-interface options of the dScope scripting language, VBScript. By default, VBScript simply allows a simple message box (**MsgBox**) or text input control (**InputBox**) which are not particularly flexible or tidy.

The **ScriptDlg** ActiveX control allows you to create simple dialogue boxes from within a script. These can contain edit controls, buttons, check boxes, and even slider controls and selection lists.

For a brief introduction to using the ScriptDlg ActiveX control, see the Introduction section.

For a full list of the controls, properties and methods available, see the ScriptDlg reference section.

7.1 Introduction

A **ScriptDlg** object is created from a script using the standard methods for creating ActiveX controls - using "CreateObject". This function will return a reference to the object in question, which is then used in every further operation involving the **ScriptDlg** object.

```
Set form = CreateObject("ScriptDlg.Form")
```

Once the form object has been created, you will need to decide whether you want to handle events in your form (button clicks, list box selections, etc). This will usually be the case, so you will need to initialise the object's Event Handler. To do this, pass the object you just created to the InitEventHandler method:

```
form.InitEventHandler(form)
```

Now it's time to add some controls. To do this, you need to call Add... for each control that you want in your dialogue box:

```
form.AddStatic "static1", "This is some text"
form.AddPushButton "button1", "Click me"
```

Note that the first parameter to each of these Add... methods is the name of the button. This will be the name used when referring to the control throughout the script.

Finally, now our controls are set up, we need to tell the ActiveX control to actually display the dialogue box:

```
form.Display()
```

If you run the script, you should now see a dialogue box that looks something like the following:

Note that you didn't have to tell the dialogue where to put the controls, as it positions and sizes them automatically. You can set positions and sizes if you want to though - see the ScriptDlg reference for further details.

7.2 ScriptDlg reference

This reference section describes all of the **ScriptDlg**' s properties and methods. It also describes events that can be fired from the controls on a dialogue.

The following objects are described as part of this reference:

Form
Push buttons
Edit controls
Static (text) controls
Check boxes
Radio buttons
Slider controls
Drop-list controls
List box controls
Bitmap controls
Progress controls
ScrollBar controls

7.2.1 Form

The form is the main dialogue box for the ScriptDlg control.

Methods

InitEventHandler
SetEvent_OnCreate
SetEvent_OnClose
SetEvent_OnHelp
Display
Close
NewRow
SetFont
StartButtonGroup
Sleep
Minimize
Restore
ShowHelpTopic
ShowBrowseDlg
SetBackgroundColour
GetVersion

AddPushButton
AddStatic
AddEdit
AddCheckBox
AddRadioButton
AddSlider
AddDropList
AddListBox
AddBitmap
AddProgress
AddScrollBar

Properties

Title
XPos
YPos
Width
Height
Modal
IsActive
DynamicallyResize
VerticalSpacing
HorizontalSpacing
RunFromdScope

BrowsePath
BrowseTitle
BrowseFileFilter

Events

OnCreate
OnClose
OnHelp

7.2.1.1 Properties

7.2.1.1.1 XPos

Description

This property represents the X position of the form, in pixels.

If this property is not explicitly set, the form will be centred horizontally on the screen.

Values

Any numerical value can be used.

NB:

If the X position specified is greater than the current X range of the screen resolution, the form will be off the right edge of the screen and therefore unusable!

The same applies if the X position is less than zero, as it will cause the dialog to be started off the left of the screen.

7.2.1.1.2 YPos

Description

This property represents the Y position of the form, in pixels.

If this property is not explicitly set, the form will be centred vertically on the screen.

Values

Any numerical value can be used.

NB: **If the Y position specified is greater than the current Y range of the screen resolution, the form will be off the bottom of the screen and therefore unusable!**

The same applies if the Y position is less than zero, as it will cause the dialog to be started off the top of the screen.

7.2.1.1.3 Width

Description

This property represents the width of the form, in pixels.

If this property is not explicitly specified, then the form will be resized automatically according to the controls on it. Note that this may make it larger than the screen.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified is greater than the current X range of the screen resolution, the form will be larger than the screen. This may cause controls on the form to be lost at the edges.**

7.2.1.1.4 Height

Description

This property represents the height of the form, in pixels.

If this property is not explicitly specified, then the form will be resized automatically according to the controls on it. Note that this may make it larger than the screen.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified is greater than the current Y range of the screen resolution, the form will be larger than the screen. This may cause controls on the form to be lost at the top and bottom.**

7.2.1.1.5 Modal

Description

This property specifies whether the form should be **modal** (i.e. no actions can be taken on other applications while the dialogue box is shown), or **modeless** (other applications can be used while the dialogue box is open)

NB: **If the script containing the ScriptDlg form must process events while the ScriptDlg is open, then the Modal property MUST be set to False.**

By default, the form will be modal.

NB: **If the ScriptDlg form is modeless (i.e. the Modal property is set to False) then the script MUST contain code after the Display method is called, to ensure that the script does not end before the ScriptDlg form is closed.**
It does this by using the IsActive property.

Values

True	Sets the form to be modal
False	Sets the form to be modeless

7.2.1.1.6 IsActive

Description

This **read-only** property specifies whether a modeless form (i.e. one with its Modal property set to **False**) is still active.

If the form is set to modeless, this property MUST be checked to ensure that the script does not end before the form has been closed.

For example :

```
' Create and set up parts of the form, including:
form.Modal = False

' Display the form
form.Display()

' Wait until the user has closed the form
While form.IsActive
    ' Wait around until finished.
    ' This will allow events to happen, controls on the
    ' form to be updated, etc.
Wend
```

NB: **If the ScriptDlg is modeless, and it is NOT run from within the dScope, the RunFromdScope property MUST be set to False. Otherwise, the ScriptDlg will not respond to any mouse clicks or other Windows events, and will appear to hang.**

Values

True	Returned if the form is still active
False	Returned if the form is no longer active

7.2.1.1.7 Title

Description

This property represents the title of the form, i.e. the text that will appear in the title bar.

Values

Any string can be used as the form's title.

7.2.1.1.8 DynamicallyResize

Description

This property specifies whether the form should be dynamically resized as controls are added to it.

By default, the form will be dynamically resizable.

NB: **If you do not specify a Width for the form, the form will be dynamically resized regardless of the value of this property.**

Values

True	Sets the form to be dynamically resizable
False	Sets the form to be statically sized, i.e. the form will have the width specified by the Width property.

7.2.1.1.9 VerticalSpacing

Description

This property specifies the vertical spacing between controls on the form, in pixels.

The default value for this property is 3 pixels.

NB: **This value is also used for the vertical margins on the form, i.e. the distance that controls will be positioned from the top and bottom of the form.**

Values

Any numerical value greater than 0 and less than 100 is allowed.

7.2.1.1.10 HorizontalSpacing

Description

This property specifies the horizontal spacing between controls on the form, in pixels.

The default value for this property is 3 pixels.

NB: **This value is also used for the horizontal margins on the form, i.e. the distance that controls will be positioned from the left and right hand sides of the form.**

Values

Any numerical value greater than 0 and less than 100 is allowed.

7.2.1.1.11 RunFromdScope

Description

This property can be set to specify whether the ScriptDlg form has been instantiated from within the dScope Series III software.

This property is only currently useful in the following situation: when a script that is run from within the dScope stops, it automatically closes down any ScriptDlg forms that are open. If a ScriptDlg's **RunFromdScope** property is set to **False**, then it will *not* be shut down automatically.

NB: Note that if this property is set to **True**, then any function key that is pressed (F1..F12, or <Ctrl>, <Alt> or <Shift> together with F1..F12) will be passed on the running dScope program. This allows function keys to work even if a ScriptDlg form has the focus instead of the dScope program.

Values

True	Specifies that this ScriptDlg form has been created from within a dScope script (default).
False	Specifies that this ScriptDlg form has been created as a standalone entity, i. e. from *outside* the dScope.

7.2.1.1.12 BrowsePath

Description

This property can be used to set the foile/folder path that the Browse dialogue box will use (See ShowBrowseDlg).

This property should be set before calling ShowBrowseDlg to the path that the Browse dialog box should be opened in. If a file is selected and [OK] is pressed, **this property will be overwritten** with the full path name of the file selected.

Values

Any valid file path can be used. If a folder is specified, then the Browse dialogue box will open in that folder; if a full file path is specified, then the file name will be selected by default in the dialogue box.

7.2.1.1.13 BrowseTitle

Description

This property can be used to set the title of the Browse dialogue box (See ShowBrowseDlg).

By default, the title of the Browse dialogue box will be "Open" or "Save As", depending on the ***bOpen*** parameter passed to ShowBrowseDlg.

Values

Any valid string can be used as the title of the Browse dialogue box. If set to a null string ("") then the default title will be used ("Open" or "Save As", depending on the ***bOpen*** parameter passed to ShowBrowseDlg).

7.2.1.1.14 BrowseFileFilter

Description

This property can be used to set the file filter to use in the Browse dialogue box (See ShowBrowseDlg). This limits the types of files that will appear in the dialogue box, based on their file extensions. By default, the file filter is empty ("") and will show all files (*.*).

This property can be set to **"Folders"** to cause the Browse dialogue box to browse for folders rather than files.

If this property is being used to specify a file filter, then it must be specified as one or more pairs of strings, separated by the '|' character. The first string in the string pair describes the filter; the second string indicates the file extension to use. Multiple extensions may be specified using ';' as the delimiter. The string ends with two '|' characters.

For example, to allow two selectable filters: for dScope script files, or "All files", you should specify the following string:

```
strFilter = "Scripts (*.dss; *.vbs)|*.dss; *.vbs|"
strFilter = strFilter & "All files (*.*)|*.*||"
```

Values

Either the string "Folders" should be used, or any string containing a valid set of file filters as described above. If set to a null string ("") then the default file filter will be used ("All files (*.*)").

7.2.1.2 Methods

7.2.1.2.1 InitEventHandler

InitEventHandler (formobject)

This method is used to initialize the part of the form that handles events. It must be called if the script is to respond to any button clicking, list-box selection changing, etc.

Parameters

formobject	The form object that has been created. This will be the value returned by the CreateObject function.

Return value

This method has no return value.

Example

The following code will create a form object called "form1" and initialize the object so that events will be fired.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)
```

7.2.1.2.2 SetEvent_OnCreate

SetEvent_OnCreate (eventref)

This method is used to specify the function that will be called when the form is initially created. The form will be initialized with all the controls on it, and then this event will be fired in case the script needs to do any further processing.

See OnCreate event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the form is created. This must be passed in the form **GetRef ("form_OnCreate")**.

Return value

This method has no return value.

Example

The following code will create a form object called "form1", initialize the object so that events will be fired, and set an event that is fired when the dialogue box is created.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.SetEvent_OnClose GetRef("form1_OnCreate")

form1.Display()

Sub form1_OnCreate()
MsgBox "The form has been created"
End Sub
```

7.2.1.2.3 SetEvent_OnClose

SetEvent_OnClose (eventref)

This method is used to specify the function that will be called when the form is closed.

See OnClose event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the form is closed. This must be passed in the form **GetRef ("form_OnClose")**.

Return value

This method has no return value.

Example

The following code snippet will create a form object called "form1", initialize the object so that events will be fired, and set an event that is fired when the dialogue box is closed.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.SetEvent_OnClose GetRef("form1_OnClose")

form1.Display()

Sub form1_OnClose(sValue)
      MsgBox "The form has been closed"
End Sub
```

7.2.1.2.4 SetEvent_OnHelp

SetEvent_OnHelp (eventref)

This method is used to specify the function that will be called when the F1 key is pressed while a Form is open and is the currently active window.

See OnHelp event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the F1 key is pressed. This must be passed in the form **GetRef ("form_OnHelp")**.

Return value

This method has no return value.

Example

The following code snippet will create a form object called "form1", initialize the object so that events will be fired, and set an event that is fired when the F1 key is pressed.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.SetEvent_OnHelp GetRef("form1_OnHelp")

form1.Display()

Sub form1_OnHelp()
      Call form1.ShowHelpTopic("MyFormHelp.chm", "Index.htm")
End Sub
```

7.2.1.2.5 Display

Display ()

Once controls have been set up on a ScriptDlg form, this method actually displays the form.

NB: **If the ScriptDlg form has been set up as Modal, the script will not return from this method until the ScriptDlg has been closed.**

This is the point at which all controls' automatic sizes and positions are calculated, so ALL controls must have been added to the form before this method is used.

Parameters

This method has no parameters

Return value

This method has no return value.

7.2.1.2.6 Close

Close (sValue)

This method can be called to close the form.

Parameters

sValue	A value associated with the closing of the form. For example, you could use this parameter to determine which button was used to close the form. This parameter is passed as a parameter to the OnClose event.

Return value

This method has no return value.

7.2.1.2.7 NewRow

NewRow ()

This method can be called to start a new row in the form.

By default, automatic positioning of controls will add them across the form from left to right. If this methods is called, the next control will revert to being on the next row down, starting on the left hand side of the form.

NB: **Starting a new row will set the position of the next control to be HorizontalSpacing pixels from the left hand side of the form, and VerticalSpacing pixels below the bottom of the largest control in the previous row.**

Parameters

This method has no parameters.

Return value

This method has no return value.

7.2.1.2.8 SetFont

SetFont (strFontName, sSize, sStyle)

This method can be called to set the font on the form.

Parameters

strFontName	The name of the font. This can be any one of the standard Windows font names, such as "Arial", "Courier", etc.
sSize	The size of the font, in points. The default font size is the same as the system font (usually 10 points).
sStyle	The style of the font to set. It can have any combination of the styles listed under Font Styles below.

Return value

This method has no return value.

Font Styles

The following font styles, or combinations of styles, are allowed for the ***sStyle*** parameter:

fontBold	**1**	Sets the font to be **Bold**.
fontItalic	**2**	Sets the font to be *italic*.

Example

Use the following line to set the font of the form to a 12-point, bold and italic, Arial font.

```
form.SetFont "Arial", 12, 3
```

7.2.1.2.9 StartButtonGroup

StartButtonGroup ()

This method can be called to start a group of radio buttons.

When radio buttons are put on a form, by default they will all be in the same group. This means that when any radio button is the group is selected, all other radio buttons will automatically be un-selected.

This method allows creation of more than one set of selection criteria on a form.

NB: **This method does not have to be on the line before the first radio button in the group; in fact a radio button group can also contain other controls. All non-radio-button controls will be ignored for purposes of determining the buttons in a group.**

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

Use the following code to start two groups of radio buttons, each containing three buttons.

```
form.AddStatic "static1", "First set of buttons"
form.NewRow
' Note that 'StartButtonGroup' is not necessary here
' because they are all in the same group until we
' call StartButtonGroup to start a new one.
form.AddRadioButton "radio1", "selection 1"
form.AddRadioButton "radio2", "selection 2"
form.AddRadioButton "radio3", "selection 3"
form.NewRow

form.AddStatic "static2", "Second set of buttons"
form.NewRow
form.StartButtonGroup
form.AddRadioButton "radio4", "selection 4"
form.AddRadioButton "radio5", "selection 5"
form.AddRadioButton "radio6", "selection 6"
```

7.2.1.2.10 Sleep

Sleep (lNumMilliseconds)

This method causes the script to wait for the specified time, in the meantime allowing other threads to run (for example, any of the dScope program threads).

If the script simply sits in a loop to wait (for example, in a `while` loop), then Windows will keep the script thread running continuously and so no other software used with the script will be able to run. The **Sleep** function ensures that other software can run while the script containing the ScriptDlg form is waiting.

Parameters

lNumMilliseconds	This specifies the time to wait, in milliseconds. Note that this time is approximate, as Windows will wait for *at least* this amount of time before returning control to the script. A value of 0 can be used to simply ask other threads to process before returning. NB: The Windows timer is only accurate to around the nearest 50ms.

Return value

This method has no return value.

7.2.1.2.11 Minimize

Minimize ()

This method can be called to minimize the ScriptDlg form. This will cause the form to be displayed as an icon in the Windows toolbar at the bottom of the screen.

Once minimized, the ScriptDlg form can be restored to its original position using the Restore method.

Parameters

This method has no parameters.

Return value

This method has no return value.

7.2.1.2.12 Restore

Restore ()

This method can be called to restore a minimized ScriptDlg form to its original position.

Parameters

This method has no parameters.

Return value

This method has no return value.

7.2.1.2.13 ShowHelpTopic

ShowHelpTopic (strHelpFile, strHelpTopic)

This method opens a help file, and displays the specified help topic from it.

Parameters

strHelpFile	Full path of the help file, enclosed in double quotation marks ("..."). This will usually have the extension ".chm".
strHelpTopic	The name of the help topic to display, enclosed in double quotation marks ("..."). This will usually have the extension ".htm". NB: You can open the help topic at a specific "Anchor" point by appending "#Anchor" to the help topic name, where "Anchor" is the name of the anchor or bookmark in the topic.

Return value

This method has no return value.

7.2.1.2.14 ShowBrowseDlg

ShowBrowseDlg (bOpen)

This method opens a standard Windows dialogue box to allow selection of a file for opening or saving. If a file is selected, the BrowsePath property will contain the full path and filename of the file selected.

If the BrowseFileFilter property is set to the text "**Folders**", then this method will open a dialogue box for browsing *folders* rather than files.

NB: **The BrowsePath property is also used to determine the folder in which the Browse dialogue box opens; if a file is selected and the OK button is pressed, it will be overwritten with the details of the selected file.**

Parameters

bOpen	**True** to show an "Open" dialogue box; **False** to show a "Save As" dialogue box. This parameter is ignored if the BrowseFileFilter property is set to "Folders".

Return value

This method has no return value.

7.2.1.2.15 SetBackgroundColour

SetBackgroundColour (sRed, sGreen, sBlue)

This method is used to set the background colour of this form.

Parameters

sRed	The red component of the colour to set the background to (0 - 255).
sGreen	The green component of the colour to set the background to (0 - 255).
sBlue	The blue component of the colour to set the background to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set a form's background colour to red.

```
form1.SetBackgroundColour 255, 0, 0
```

7.2.1.2.16 GetVersion

GetVersion(sMajor, sMax)

This method is used to get the version details of the ScriptDlg control. The version details are returned as a major and minor version number; for example a version number of 1.00 will return a major version of 1 and a minor version of 0; Version 0.54 will return a major version of 0 and a minor version of 54.

Parameters

sMajor	After this method is called, this parameter will hold the major version number of the ScriptDlg control.
sMinor	After this method is called, this parameter will hold the minor version number of the ScriptDlg control.

Return value

This method has no return value.

7.2.1.2.17 AddPushButton

AddPushButton strName, strText [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a push-button to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Push buttons for the properties and methods available to the button once it has been added to the form.

Parameters

strName	The name of the button. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
strText	The text to put on the button.
sWidth	(*optional*) The width of the button, in pixels. Use -1 to automatically size the button to the width of the text.
sHeight	(*optional*) The height of the button, in pixels. Use -1 to automatically size the button to the height of the text.
sXPos	(*optional*) The X position of the button, in pixels from the left-hand side of the form. Use -1 to automatically position the button to the right of the previous control.
sYPos	(*optional*) The Y position of the button, in pixels from the left-hand side of the form. Use -1 to automatically position the button at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a push-button, at an X position of 150 pixels and 20 pixels high (but automatically sized to the width of the text).

```
form.AddPushButton "Button1", "Select Me!", -1, 20, 150
```

7.2.1.2.18 AddStatic

AddStatic strName, strText [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a static text control to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Static (text) Controls for the properties and methods available to the static control once it has been added to the form.

Parameters

strName	The name of the static control. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
strText	The text to put on the static control.
sWidth	(*optional*) The width of the static control, in pixels. Use -1 to automatically size the static control to the width of the text.
sHeight	(*optional*) The height of the static control, in pixels.

	Use -1 to automatically size the static control to the height of the text.
sXPos	(*optional*) The X position of the static control, in pixels from the left-hand side of the form. Use -1 to automatically position the static control to the right of the previous control.
sYPos	(*optional*) The Y position of the static control, in pixels from the left-hand side of the form. Use -1 to automatically position the static control at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a static control, at an X position of 100 pixels and 14 pixels high (but automatically sized to the width of the text).

```
form.AddStatic "static2", "This is some text", -1, 14, 100
```

7.2.1.2.19 AddEdit

AddEdit strName, strText [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add an edit control to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Edit Controls for the properties and methods available to the edit control once it has been added to the form.

Parameters

strName	The name of the edit control. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
strText	The text to put on the edit control.
sWidth	(*optional*) The width of the edit control, in pixels. Use -1 to automatically size the edit control to the width of the text.
sHeight	(*optional*) The height of the edit control, in pixels. Use -1 to automatically size the edit control to the height of the text.
sXPos	(*optional*) The X position of the edit control, in pixels from the left-hand side of the form. Use -1 to automatically position the edit control to the right of the previous control.
sYPos	(*optional*) The Y position of the edit control, in pixels from the left-hand side of the form. Use -1 to automatically position the edit control at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add an edit control, at an X position of 30 pixels and 50 pixels high (but automatically sized to the width of the text).

```
form.AddEdit "edit3", "Type some text here...", -1, 50, 30
```

7.2.1.2.20 AddCheckBox

AddCheckBox strName, strText [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a check box to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Check boxes for the properties and methods available to the check box once it has been added to the form.

Parameters

strName	The name of the check box. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
strText	The text to put on the check box.
sWidth	(*optional*) The width of the check box, in pixels. Use -1 to automatically size the control to the width of the text and the check box.
sHeight	(*optional*) The height of the check box, in pixels. Use -1 to automatically size the control to the height of the text.
sXPos	(*optional*) The X position of the check box, in pixels from the left-hand side of the form. Use -1 to automatically position the check box to the right of the previous control.
sYPos	(*optional*) The Y position of the check box, in pixels from the left-hand side of the form. Use -1 to automatically position the check box at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a check box, at an X position of 5 pixels, but automatically sized to the width and height of the text.

```
form.AddCheckBox "CheckBox5", "Click to turn on/off!",
    -1, -1, 5
```

7.2.1.2.21 AddRadioButton

AddRadioButton strName, strText [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a radio button to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Radio buttons for the properties and methods available to the radio button once it has been added to the form.

Radio buttons can be grouped. When a radio button is selected, all other radio buttons in the same group will automatically be un-selected. For details of how to group radio buttons, see StartButtonGroup.

Parameters

strName	The name of the radio button. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
strText	The text to put on the radio button.
sWidth	(*optional*) The width of the radio button, in pixels. Use -1 to automatically size the control to the width of the text and the radio button.
sHeight	(*optional*) The height of the radio button, in pixels. Use -1 to automatically size the control to the height of the text.
sXPos	(*optional*) The X position of the radio button, in pixels from the left-hand side of the form. Use -1 to automatically position the radio button to the right of the previous control.
sYPos	(*optional*) The Y position of the radio button, in pixels from the left-hand side of the form. Use -1 to automatically position the radio button at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add two radio buttons, at X positions of 35 pixels, but automatically sized to the width and height of the text.

```
form.AddRadioButton "Radio1", "This one's for kids",
    -1, -1, 35
form.AddRadioButton "Radio2", "What Radio1 used to be",
    -1, -1, 35
```

7.2.1.2.22 AddSlider

AddSlider strName, [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a slider control to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Slider Controls for the properties and methods available to the slider control once it has been added to the form.

Parameters

strName	The name of the slider control. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
sWidth	(*optional*) The width of the slider control, in pixels. Use -1 to automatically size the control to a width based on the number of ticks in the slider control.
sHeight	(*optional*) The height of the slider control, in pixels. Use -1 to automatically size the control's height.
sXPos	(*optional*) The X position of the slider control, in pixels from the left-hand side of the form. Use -1 to automatically position the slider control to the right of the previous control.
sYPos	(*optional*) The Y position of the slider control, in pixels from the left-hand side of the form. Use -1 to automatically position the slider control at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a slider control, at an X position of 40 pixels and a Y position of 20, but automatically sized.

```
form.AddSlider "Slider1", -1, -1, 40, 20
```

7.2.1.2.23 AddDropList

AddDropList strName, [, sWidth [, sXPos [, sYPos]]]

This method can be called to add a drop-list control to the form.

The ***sWidth***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Drop-list controls for the properties and methods available to the drop-list control once it has been added to the form.

NB: **The height of the drop-list will always be set automatically to the height of one line of text, in the drop-list's font.**

Parameters

strName	The name of the drop-list control. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
sWidth	(*optional*) The width of the drop-list control, in pixels. Use -1 to automatically size the control to a width based on the width of the longest text in the drop-list.
sXPos	(*optional*) The X position of the drop-list control, in pixels from the left-hand side of the form. Use -1 to automatically position the drop-list control to the right of the previous control.
sYPos	(*optional*) The Y position of the drop-list control, in pixels from the left-hand side of the form. Use -1 to automatically position the drop-list control at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a drop-list control, 250 pixels wide, at an X position of 40 pixels and a Y position of 20, but automatically sized.

```
form.AddDropList "DropList1", 250, -1, 40, 20
```

7.2.1.2.24 AddListBox

AddListBox strName, [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a list box control to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See List box controls for the properties and methods available to the list box control once it has been added to the form.

Parameters

strName	The name of the list box control. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
sWidth	(*optional*) The width of the list box control, in pixels. Use -1 to automatically size the control to an arbitrary width (around 200 pixels)
sHeight	(*optional*) The height of the list box control, in pixels. Use -1 to automatically size the control to an arbitrary height (enough for 10 items)
sXPos	(*optional*) The X position of the list box control, in pixels from the left-hand side of the form. Use -1 to automatically position the list box control to the right of the previous control.
sYPos	(*optional*) The Y position of the list box control, in pixels from the left-hand side of the form. Use -1 to automatically position the list box control at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a drop-list control, 200 pixels wide and 150 pixels high, at an X position of 10 pixels and a Y position of 20.

```
form.AddListBox "MyListBox", 200, 150, 10, 20
```

7.2.1.2.25 AddBitmap

AddBitmap strName, strFileName, [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a bitmap control to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Bitmap controls for the properties and methods available to the bitmap control once it has been added to the form.

Parameters

strName	The name of the bitmap control. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.

strFileName	The file name of the bitmap to insert into this control.
sWidth	(*optional*) The width of the bitmap control, in pixels. Use -1 to automatically size the control to the width of the bitmap specified with ***strFileName***.
sHeight	(*optional*) The height of the bitmap control, in pixels. Use -1 to automatically size the control to the height of the bitmap specified with ***strFileName***.
sXPos	(*optional*) The X position of the bitmap control, in pixels from the left-hand side of the form. Use -1 to automatically position the bitmap control to the right of the previous control.
sYPos	(*optional*) The Y position of the bitmap control, in pixels from the left-hand side of the form. Use -1 to automatically position the bitmap control at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a bitmap control, at an X position of 40 pixels and a Y position of 20, automatically sized to the size of the bitmap. The bitmap to display in the control is stored in "C:\My Documents\Bitmap1.bmp".

```
form.AddBitmap "Bitmap1",
    "C:\My Documents\Bitmap1.bmp",
    -1, -1, 40, 20
```

7.2.1.2.26 AddProgress

AddProgress strName, [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a progress control to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See Progress controls for the properties and methods available to the progress control once it has been added to the form.

Parameters

strName	The name of the progress control. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
sWidth	(*optional*) The width of the progress control, in pixels. Use -1 to automatically size the control to a nominal width.
sHeight	(*optional*) The height of the progress control, in pixels. Use -1 to automatically size the progress to a nominal height.
sXPos	(*optional*) The X position of the progress control, in pixels from the left-hand side of the form. Use -1 to automatically position the progress control to the right of the

	previous control.
sYPos	(*optional*) The Y position of the progress control, in pixels from the left-hand side of the form. Use -1 to automatically position the progress control at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a progress control, automatically positioned after the previous control, with a width of 150 pixels and an automatic height.

```
form.AddProgress "Progress", 150
```

7.2.1.2.27 AddScrollBar

AddScrollBar strName, [, sWidth [, sHeight [, sXPos [, sYPos]]]]

This method can be called to add a ScrollBar control to the form.

The ***sWidth***, ***sHeight***, ***sXPos*** and ***sYPos*** are optional parameters - however if any are specified, then all the parameters to the left must also be specified (see Example section below).
Any parameters not specified will be passed as -1, to denote "automatic".

See ScrollBar Controls for the properties and methods available to the ScrollBar control once it has been added to the form.

Parameters

strName	The name of the ScrollBar control. This name does not appear on the form, but is used to refer to this control to set further properties of the control itself.
sWidth	(*optional*) The width of the ScrollBar control, in pixels. Use -1 to automatically size the control to a width based on the number of ticks in the ScrollBar control.
sHeight	(*optional*) The height of the ScrollBar control, in pixels. Use -1 to automatically size the control's height.
sXPos	(*optional*) The X position of the ScrollBar control, in pixels from the left-hand side of the form. Use -1 to automatically position the control to the right of the previous control.
sYPos	(*optional*) The Y position of the ScrollBar control, in pixels from the left-hand side of the form. Use -1 to automatically position the control at the same height as the previous control (or the row below, if NewRow has been called in-between).

Return value

This method has no return value.

Example

Use the following line to add a ScrollBar control, at an X position of 40 pixels and a Y position of 20, but automatically sized.

```
form.AddScrollBar "VScroll", -1, -1, 40, 20
```

7.2.1.3 Events

7.2.1.3.1 OnCreate event

The OnCreate event will be fired when the form is created. This will be after all controls have been added to the form, but after it is actually displayed.

To tell the form which event function to call on creation, use the SetEvent_OnCreate method.

The OnCreate event function may have any name, but must have the following format:

```
Sub form_OnCreate()
    ' Do your processing here
End Sub
```

Parameters

This event function has no parameters.

Return value

This event function has no return value.

7.2.1.3.2 OnClose event

The OnClose event will be fired when the form is closed. This may be from the Close method, or simply by the user clicking on the "X" in the top-right corner of the form.

To tell the form which event function to call on closing the form, use the SetEvent_OnClose method.

The OnClose event function may have any name, but must have the following format:

```
Sub form_OnClose (sValue)
    ' Do your processing here
End Sub
```

NB: **A parameter (e.g. *sValue*, as shown above) must be included in this event, or it will not get called.**

Parameters

sValue	Indicates the value passed to the Close function. This can be used by the script to determine, for example, which button was used to close the form.

Return value

This event function has no return value.

7.2.1.3.3 OnHelp event

The OnHelp event will be fired when the F1 key is pressed while the form is active.

To tell the form which event function to call, use the SetEvent_OnHelp method.

The OnHelp event function may have any name, but must have the following format:

```
Sub form_OnHelp()
     ' Do your processing here
End Sub
```

Parameters

This event function has no parameters.

Return value

This event function has no return value.

7.2.2 Push buttons

Push buttons allow the user to perform some action by clicking on the button.

Creation

A push button can be created using the form's AddPushButton method. This method takes as one of its parameters the name of the button, which is then used throughout the script when referring to this button.

For example

```
form.AddPushButton "Button1", "Click me!", 100, 12, 5, 5
```

would mean that all properties and methods of this button would be called by simply prefixing them with

```
form.Button1.
```

Properties

Visible
Enabled
XPos
YPos
Width
Height
Text
TabStop
Alignment

Methods

SetTextColour
SetBackgroundColour
SetFont
SetDefault
SetEvent_OnClick

Events

OnClick

7.2.2.1 Properties

7.2.2.1.1 Visible

Description

This property specifies whether the button should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A button is visible by default.

NB: **An invisible button cannot fire events.**

Values

True	Sets the button to be visible.
False	Sets the button to be invisible.

7.2.2.1.2 Enabled

Description

This property specifies whether the button should be enabled or disabled.

A button is enabled by default.

NB: **A disabled button cannot fire any events.**

Values

True	Sets the button to be enabled.
False	Sets the button to be disabled.

7.2.2.1.3 XPos

Description

This property represents the X position of the button, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddPushButton is used, then the button will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.2.1.4 YPos

Description

This property represents the Y position of the button, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddPushButton is used, then the button will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.2.1.5 Width

Description

This property represents the width of the button, in pixels.

If this property is not explicitly specified, either using this property or when AddPushButton is used, then the button will be sized to fit the current text (specified using Text, or originally with AddPushButton). In this case, any changes to the text may result in the button changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.2.1.6 Height

Description

This property represents the height of the button, in pixels.

If this property is not explicitly specified, either using this property or when AddPushButton is used, then the button will be sized to fit the current text (specified using Text, or originally with AddPushButton). In this case, any changes to the text may result in the button changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.2.1.7 Text

Description

This property represents the text on the button.

Values

Any string can be used as the button's text.

7.2.2.1.8 TabStop

Description

This property specifies whether the button should have the "TabStop" property.

A button with the TabStop property will get the focus as the TAB key is used to move through the controls on the form.

By default, buttons have the TabStop property set to **True**.

Values

True	Sets the button to have the TabStop property.
False	Specifies that the button should not have the TabStop property.

7.2.2.1.9 Alignment

Description

This property represents the alignment of the text within the button control.

It can be any of the alignments detailed below. Note that if more than one alignment is specified, the first one from the list below will be used.

The default alignment for the text on a button is in the centre.

Values

alignLeft	**1**	Sets the text on the button to be horizontally aligned to the left of the control.
alignHCentre	**2**	Sets the text on the button to be horizontally aligned in the centre of the control.
alignRight	**4**	Sets the text on the button to be horizontally aligned to the right of the control.

7.2.2.1.10 TooltipText

Description

This property represents the Tooltip text for the button. This is text displayed in a small pop-up window when the mouse pointer is held over the button. It is usually used to contain a brief description of the purpose of the button.

Values

Any string can be used as the Tooltip text, up to a maximum of 80 characters.

7.2.2.2 Methods

7.2.2.2.1 SetTextColour

SetTextColour (sRed, sGreen, sBlue)

This method is used to set the text colour of this button.

Parameters

sRed	The red component of the colour to set the text to (0 - 255).
sGreen	The green component of the colour to set the text to (0 - 255).
sBlue	The blue component of the colour to set the text to (0 - 255).

NB:

To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".

Return value

This method has no return value.

Example

The following code will set two buttons' text colours to blue and green, respectively.

```
form1.button1.SetTextColour 0, 0, 255
form1.button2.SetTextColour 0, 255, 0
```

7.2.2.2.2 SetBackgroundColour

SetBackgroundColour (sRed, sGreen, sBlue)

This method is used to set the background colour of this button.

Parameters

sRed	The red component of the colour to set the background to (0 - 255).
sGreen	The green component of the colour to set the background to (0 - 255).
sBlue	The blue component of the colour to set the background to (0 - 255).

NB:

To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".

Return value

This method has no return value.

Example

The following code will set two buttons' background colours to black and white, respectively.

```
form1.button1.SetBackgroundColour 0, 0, 0
form1.button2.SetBackgroundColour 255, 255, 255
```

7.2.2.2.3 SetFont

SetFont (strFontName, sSize, sStyle)

This method can be called to set the font of the text on a button.

Parameters

strFontName	The name of the font. This can be any one of the standard Windows font names, such as "Arial", "Courier", etc.
sSize	The size of the font, in points. The default font size is the same as the system font (usually 10-points).
sStyle	The style of the font to set. It can have any combination of the styles listed under Font Styles below.

Return value

This method has no return value.

Font Styles

The following font styles, or combinations of styles, are allowed for the ***sStyle*** parameter:

fontBold	**1**	Sets the font to be **Bold**.
fontItalic	**2**	Sets the font to be *italic*.

Example

Use the following line to set the font of a button to a 10-point, bold and italic, "MS Sans Serif" font.

```
form.button3.SetFont "MS sans Serif", 10, 3
```

7.2.2.2.4 SetFocus

SetFocus ()

This method can be called to set the focus to this button (i.e. make it the control that currently handles keyboard input). This means that pressing the Enter key, or the space bar, will act as a click on the button.

NB: **Giving the focus to a control will remove the focus from any control which has previously been given it.**

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

Use the following line to set the focus to a previously-created push button.

```
form.MyButton1.SetFocus()
```

7.2.2.2.5 HasFocus

HasFocus ()

This method can be called to determine whether a control currently has the focus.

Parameters

This method has no parameters.

Return value

This method returns **True** .

Example

Use the following line to determine whether a previously-created push button has the focus. This can be used in an OnClick event handler, for example, to determine which button was actually clicked.

```
If form.MyButton1.HasFocus() Then
    ' Do something here
End If
```

7.2.2.2.6 SetDefault

SetDefault ()

This method is used to set this button to be the default selected button. This means that if the <Enter> key is pressed, it will behave like a click on this button.

Parameters

This method has no parameters.

NB: **If a button is set as the default button, then all other buttons on the form will have their default property set to False.**

Return value

This method has no return value.

Example

The following code will set a button to be the default for a form.

```
form1.OKbutton.SetDefault()
```

7.2.2.2.7 SetEvent_OnClick

SetEvent_OnClick (eventref)

This method is used to specify the function that will be called when a button is clicked.

See OnClick event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the button is clicked. This must be passed in the form **GetRef ("button3_OnClick")**.

Return value

This method has no return value.

Example

The following code snippet will create a form, initialize it so that events will be fired, create a button called "button1", and set an event that is fired when the button is clicked.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.AddPushButton "button1", "Click me!"
form1.button1.SetEvent_OnClick GetRef("button1_OnClick")

form1.Display()

Sub button1_OnClick()
MsgBox "You clicked the button!"
End Sub
```

7.2.2.3 Events

7.2.2.3.1 OnClick event

The OnClick event will be fired when a button is clicked.

To tell the button which event function to call when it is clicked, use the SetEvent_OnClick method.

The OnClick event function may have any name, but must have the following format:

```
Sub button_OnClick()
    ' Do your processing here
End Sub
```

Parameters

This event function has no parameters.

Return value

This event function has no return value.

7.2.3 Edit controls

Edit controls allow the user to enter text on the form. They can be single or multiple lines.

Creation

An edit control can be created using the form's AddEdit method. This method takes as one of its parameters the name of the edit control, which is then used throughout the script when referring to this edit control.

For example

```
form.AddEdit "Edit1", "Enter text here...", 200, 12, 5, 5
```

would mean that all properties and methods of this edit control would be called by simply prefixing them with

```
form.Edit1.
```

Properties

Visible
Enabled
XPos
YPos
Width
Height
Text
TabStop
Alignment
PasswordStyle
ReadOnly
TooltipText

Methods

SetTextColour
SetBackgroundColour
SetFont

Events

Edit controls do not fire events.

7.2.3.1 Properties

7.2.3.1.1 Visible

Description

This property specifies whether the edit control should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

An edit control is visible by default.

Values

True	Sets the edit control to be visible.
False	Sets the edit control to be invisible.

7.2.3.1.2 Enabled

Description

This property specifies whether the edit control should be enabled or disabled.

An edit control is enabled by default.

NB: **A disabled edit control stops the user from entering text.**

Values

True	Sets the edit control to be enabled.
False	Sets the edit control to be disabled.

7.2.3.1.3 XPos

Description

This property represents the X position of the edit control, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddEdit is used, then the edit control will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.3.1.4 YPos

Description

This property represents the Y position of the edit control, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddEdit is used, then the edit control will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.3.1.5 Width

Description

This property represents the width of the edit control, in pixels.

If this property is not explicitly specified, either using this property or when AddEdit is used, then the edit control will be sized to fit the current text (specified using Text, or originally with AddEdit). In this case, any changes to the text may result in the edit control changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.3.1.6 Height

Description

This property represents the height of the edit control, in pixels.

If this property is not explicitly specified, either using this property or when AddEdit is used, then the edit control will be sized to fit the current text (specified using Text, or originally with AddEdit). In this case, any changes to the text may result in the edit control changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.3.1.7 Text

Description

This property represents the text in the edit control.

Values

Any string can be used as the edit control's text.

7.2.3.1.8 TabStop

Description

This property specifies whether the edit control should have the "TabStop" property.

An edit control with the TabStop property will get the focus as the TAB key is used to move through the controls on the form.

By default, edit controls have the TabStop property set to **True**.

Values

True	Sets the edit control to have the TabStop property.
False	Specifies that the edit control should not have the TabStop property.

7.2.3.1.9 Alignment

Description

This property represents the alignment of the text within the button control.

It can be any of the alignments detailed below. Note that if more than one alignment is specified, the first one from the list below will be used.

The default alignment for the text on a button is in the centre.

Values

alignLeft	**1**	Sets the text in the edit control to be horizontally aligned to the left of the control.
alignHCentre	**2**	Sets the text in the edit control to be horizontally aligned in the centre of the control.
alignRight	**4**	Sets the text in the edit control to be horizontally aligned to the right of the control.

7.2.3.1.10 PasswordStyle

Description

This property specifies whether the edit control should be a password-style control. This means that any character typed into the control will appear as an asterisk (*).

An edit control does not have the password style by default.

Values

True	Sets the edit control to be a password-style control.
False	Sets the edit control to be a non-password-style control.

7.2.3.1.11 ReadOnly

Description

This property specifies whether the edit control should be read-only. This means that the edit control will not accept any keyboard input. Note that the contents of a read-only edit control can still be selected using the mouse, and copied to the clipboard.

An edit control allows editing (i.e. it is not read-only) by default.

Values

True	Sets the edit control to be read-only.
False	Sets the edit control to allow editing.

7.2.3.1.12 TooltipText

Description

This property represents the Tooltip text for the edit control. This is text displayed in a small pop-up window when the mouse pointer is held over the control. It is usually used to contain a brief description of the purpose of the control.

Values

Any string can be used as the Tooltip text, up to a maximum of 80 characters.

7.2.3.1.13 MultiLine

Description

This property specifies whether the edit control should be a multi-line edit control. If the MultiLine property is set to **True**, then the user can enter multiple lines in the control, and can advance on to the next line using the Enter key. If it is set to **False**, then the Enter key is ignored and only a single line of text can be entered.

An edit control is multi-line by default.

NB: **If an edit control is multi-line, and it currently has the Focus (i.e. the keyboard input), then the Enter key will move to the next line in the control. Setting the MultiLine property to False will mean that the parent Form will handle the Enter key instead, allowing it to be used for (for example) the default button (See Push Button's SetDefault function).**

Values

True	Sets the edit control to be a multi-line control.
False	Sets the edit control to be a multi-line control.

7.2.3.2 Methods

7.2.3.2.1 SetTextColour

SetTextColour (sRed, sGreen, sBlue)

This method is used to set the text colour of this edit control.

Parameters

sRed	The red component of the colour to set the text to (0 - 255).
sGreen	The green component of the colour to set the text to (0 - 255).
sBlue	The blue component of the colour to set the text to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set two edit controls' text colours to red and green, respectively.

```
form1.edit1.SetTextColour 255, 0, 0
form1.edit2.SetTextColour 0, 255, 0
```

7.2.3.2.2 SetBackgroundColour

SetBackgroundColour (sRed, sGreen, sBlue)

This method is used to set the background colour of this edit control.

Parameters

sRed	The red component of the colour to set the background to (0 - 255).
sGreen	The green component of the colour to set the background to (0 - 255).
sBlue	The blue component of the colour to set the background to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set two edit controls' background colours to black and white, respectively.

```
form1.edit1.SetBackgroundColour 0, 0, 0
form1.edit2.SetBackgroundColour 255, 255, 255
```

7.2.3.2.3 SetFont

SetFont (strFontName, sSize, sStyle)

This method can be called to set the font of the text in an edit control.

Parameters

strFontName	The name of the font. This can be any one of the standard Windows font names, such as "Arial", "Courier", etc.
sSize	The size of the font, in points. The default font size is the same as the system font (usually 10-points).
sStyle	The style of the font to set. It can have any combination of the styles listed under Font Styles below.

Return value

This method has no return value.

Font Styles

The following font styles, or combinations of styles, are allowed for the ***sStyle*** parameter:

fontBold	**1**	Sets the font to be **Bold**.
fontItalic	**2**	Sets the font to be *italic*.

Example

Use the following line to set the font of an edit control to a 10-point, italic, "MS Sans Serif" font.

```
form.edit2.SetFont "MS sans Serif", 10, 2
```

7.2.3.2.4 SetFocus

SetFocus ()

This method can be called to set the focus to this edit control (i.e. make it the control that currently handles keyboard input).

NB: **Giving the focus to a control will remove the focus from any control which has previously been given it.**

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

Use the following line to set the focus to a previously-created edit control.

```
form.Edit1.SetFocus()
```

7.2.3.2.5 HasFocus

HasFocus ()

This method can be called to determine whether a control currently has the focus.

Parameters

This method has no parameters.

Return value

This method returns **True** .

Example

Use the following line to determine whether a previously-created edit control has the focus.

```
If form.MyEdit1.HasFocus() Then
     ' Do something here
End If
```

7.2.4 Static (text) controls

Static controls are simple controls that display single or multiple lines of text.

Creation

A static control can be created using the form's AddStatic method. This method takes as one of its parameters the name of the static, which is then used throughout the script when referring to this control.

For example

```
form.AddStatic "static1",
     "This is a line of informative text"
```

would mean that all properties and methods of this edit control would be called by simply prefixing them with

```
form.static1.
```

Properties

Visible
Enabled
XPos
YPos
Width
Height

Methods

SetTextColour
SetBackgroundColour
SetFont

Events

Static controls do not fire events.

7.2.4.1 Properties

7.2.4.1.1 Visible

Description

This property specifies whether the static control should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A static control is visible by default.

Values

True	Sets the static control to be visible.
False	Sets the static control to be invisible.

7.2.4.1.2 Enabled

Description

This property specifies whether the static control should be enabled or disabled.

A static control is enabled by default.

Values

True	Sets the static control to be enabled.
False	Sets the static control to be disabled.

7.2.4.1.3 XPos

Description

This property represents the X position of the static control, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddStatic is used, then the static control will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the X position is less than zero, the control's X position will start beyond the

left-hand side of the form.

7.2.4.1.4 YPos

Description

This property represents the Y position of the static control, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddStatic is used, then the static control will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.4.1.5 Width

Description

This property represents the width of the static control, in pixels.

If this property is not explicitly specified, either using this property or when AddStatic is used, then the static control will be sized to fit the current text (specified using Text, or originally with AddStatic). In this case, any changes to the text may result in the static control changing size.

Values

Any numerical value greater than 0 can be used.

NB:

If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

7.2.4.1.6 Height

Description

This property represents the height of the static control, in pixels.

If this property is not explicitly specified, either using this property or when AddStatic is used, then the static control will be sized to fit the current text (specified using Text, or originally with AddStatic). In this case, any changes to the text may result in the static control changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.4.1.7 Text

Description

This property represents the text of the static control.

Values

Any string can be used as the static control's text.

7.2.4.1.8 Alignment

Description

This property represents the alignment of the text within the static control.

It can be any combination of horizontal alignments, and vertical alignments. Note that if more than one horizontal alignment is specified, the first one from the list below will be used. Similarly, if more than one vertical alignment is specified, the first one will be used.

The default alignment for the text in a static control is at the top left.

Values

alignLeft	**1**	Sets the text in the static control to be horizontally aligned to the left of the control.
alignHCentre	**2**	Sets the text in the static control to be horizontally aligned in the centre of the control.
alignRight	**4**	Sets the text in the static control to be horizontally aligned to the right of the control.
alignTop	**8**	Sets the text in the static control to be vertically aligned at the top of the control.
alignVCentre	**16**	Sets the text in the static control to be vertically aligned in the centre of the control.
alignBottom	**32**	Sets the text in the static control to be vertically aligned at the bottom of the control.

7.2.4.2 Methods

7.2.4.2.1 SetTextColour

SetTextColour (sRed, sGreen, sBlue)

This method is used to set the text colour of this static control.

Parameters

sRed	The red component of the colour to set the text to (0 - 255).
sGreen	The green component of the colour to set the text to (0 - 255).
sBlue	The blue component of the colour to set the text to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set two static controls' text colours to red and green, respectively.

```
form1.static1.SetTextColour 255, 0, 0
form1.static2.SetTextColour 0, 255, 0
```

7.2.4.2.2 SetBackgroundColour

SetBackgroundColour (sRed, sGreen, sBlue)

This method is used to set the background colour of this static control.

Parameters

sRed	The red component of the colour to set the background to (0 - 255).
sGreen	The green component of the colour to set the background to (0 - 255).
sBlue	The blue component of the colour to set the background to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set two static controls' background colours to blue and red, respectively.

```
form1.static1.SetBackgroundColour 0, 0, 255
form1.static2.SetBackgroundColour 255, 0, 0
```

7.2.4.2.3 SetFont

SetFont (strFontName, sSize, sStyle)

This method can be called to set the font of the text in an edit control.

Parameters

strFontName	The name of the font. This can be any one of the standard Windows font names, such as "Arial", "Courier", etc.
sSize	The size of the font, in points. The default font size is the same as the system font (usually 10-points).
sStyle	The style of the font to set. It can have any combination of the styles listed under Font Styles below.

Return value

This method has no return value.

Font Styles

The following font styles, or combinations of styles, are allowed for the ***sStyle*** parameter:

fontBold	**1**	Sets the font to be **Bold**.
fontItalic	**2**	Sets the font to be *italic*.

Example

Use the following line to set the font of an edit control to a 10-point, italic, "MS Sans Serif" font.

```
form.edit2.SetFont "MS sans Serif", 10, 2
```

7.2.5 Check boxes

Check boxes give the user an On/Off or True/False type of selection.

Creation

A check box can be created using the form's AddCheckBox method. This method takes as one of its parameters the name of the check box, which is then used throughout the script when referring to this check box.

For example

```
form.AddCheckBox "checkbox2", "On/Off"
```

would mean that all properties and methods of this check box would be called by simply prefixing them with

```
form.checkbox2.
```

Properties

Visible
Enabled
XPos
YPos
Width
Height
Text
TabStop
Checked

Methods

SetTextColour
SetBackgroundColour
SetFont
SetEvent_OnClick

Events

OnClick

7.2.5.1 Properties

7.2.5.1.1 Visible

Description

This property specifies whether the check box should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A check box is visible by default.

NB: **An invisible check box cannot fire events.**

Values

True	Sets the check box to be visible.
False	Sets the check box to be invisible.

7.2.5.1.2 Enabled

Description

This property specifies whether the check box should be enabled or disabled.

A check box is enabled by default.

NB: **A disabled check box cannot fire any events.**

Values

True	Sets the check box to be enabled.
False	Sets the check box to be disabled.

7.2.5.1.3 XPos

Description

This property represents the X position of the check box, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddCheckBox is used, then the check box will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.5.1.4 YPos

Description

This property represents the Y position of the check box, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddCheckBox is used, then the check box will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.5.1.5 Width

Description

This property represents the width of the check box, in pixels.

If this property is not explicitly specified, either using this property or when AddCheckBox is used, then the control will be sized to fit the current text and check box (specified using Text, or originally with AddCheckBox).

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.5.1.6 Height

Description

This property represents the height of the check box, in pixels.

If this property is not explicitly specified, either using this property or when AddCheckBox is used, then the check box will be sized to fit the current text (specified using Text, or originally with AddCheckBox).

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.5.1.7 Text

Description

This property represents the text on the check box.

Values

Any string can be used as the check box's text.

7.2.5.1.8 TabStop

Description

This property specifies whether the check box should have the "TabStop" property.

A check box with the TabStop property set to **True** will get the focus as the TAB key is used to move through the controls on the form.

By default, check boxes have the TabStop property set to **True**.

Values

True	Sets the check box to have the TabStop property.
False	Specifies that the check box should not have the TabStop property.

7.2.5.1.9 Checked

Description

This property sets or returns the current checked state of the check box.

Values

True	The check box is enabled.
False	The check box is disabled.

7.2.5.1.10 TooltipText

Description

This property represents the Tooltip text for the check box. This is text displayed in a small pop-up window when the mouse pointer is held over the control. It is usually used to contain a brief description of the purpose of the control.

Values

Any string can be used as the Tooltip text, up to a maximum of 80 characters.

7.2.5.2 Methods

7.2.5.2.1 SetTextColour

SetTextColour (sRed, sGreen, sBlue)

This method is used to set the text colour of this check box.

Parameters

sRed	The red component of the colour to set the text to (0 - 255).
sGreen	The green component of the colour to set the text to (0 - 255).
sBlue	The blue component of the colour to set the text to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set two check boxes' text colours to blue and green, respectively.

```
form1.checkbox1.SetTextColour 0, 0, 255
form1.checkbox2.SetTextColour 0, 255, 0
```

7.2.5.2.2 SetBackgroundColour

SetBackgroundColour (sRed, sGreen, sBlue)

This method is used to set the background colour of this check box.

Parameters

sRed	The red component of the colour to set the background to (0 - 255).
sGreen	The green component of the colour to set the background to (0 - 255).
sBlue	The blue component of the colour to set the background to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set two check boxes' background colours to white and black, respectively.

```
form1.checkbox1.SetBackgroundColour 255, 255, 255
form1.checkbox2.SetBackgroundColour 0, 0, 0
```

7.2.5.2.3 SetFont

SetFont (strFontName, sSize, sStyle)

This method can be called to set the font of the text on a check box.

Parameters

strFontName	The name of the font. This can be any one of the standard Windows font names, such as "Arial", "Courier", etc.
sSize	The size of the font, in points. The default font size is the same as the system font (usually 10-points).
sStyle	The style of the font to set. It can have any combination of the styles listed under Font Styles below.

Return value

This method has no return value.

Font Styles

The following font styles, or combinations of styles, are allowed for the ***sStyle*** parameter:

fontBold	**1**	Sets the font to be **Bold**.
fontItalic	**2**	Sets the font to be *italic*.

Example

Use the following line to set the font of a check box to a 10-point, italic, "MS Sans Serif" font.

```
form.checkbox3.SetFont "MS sans Serif", 12, 3
```

7.2.5.2.4 SetFocus

SetFocus ()

This method can be called to set the focus to this check box (i.e. make it the control that currently handles keyboard input). This means that pressing the space bar will toggle whether the check box is currently checked.

NB: **Giving the focus to a control will remove the focus from any control which has previously been given it.**

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

Use the following line to set the focus to a previously-created check box.

```
form.SelectedCheckBox.SetFocus()
```

7.2.5.2.5 HasFocus

HasFocus ()

This method can be called to determine whether a control currently has the focus.

Parameters

This method has no parameters.

Return value

This method returns **True** .

Example

Use the following line to determine whether a previously-created check box has the focus.

```
If form.MyCheckBox.HasFocus() Then
     ' Do something here
End If
```

7.2.5.2.6 SetEvent_OnClick

SetEvent_OnClick (eventref)

This method is used to specify the function that will be called when a check box is clicked.

See OnClick event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the check box is clicked.

This must be passed in the form **GetRef ("button3_OnClick")**.

Return value

This method has no return value.

Example

The following code snippet will create a form, initialize it so that events will be fired, create a check box, and set an event that is fired when the check box is clicked.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.AddCheckBox "checkbox1", "Checked or not?"
form1.checkbox1.SetEvent_OnClick GetRef("checkbox1_OnClick")

form1.Display()

Sub checkbox1_OnClick()
  If form1.checkbox1.Checked Then
     MsgBox "The check box is now checked!"
  Else
     MsgBox "The check box is now unchecked!"
  End If
End Sub
```

7.2.5.3 Events

7.2.5.3.1 OnClick event

The OnClick event will be fired when a check box is clicked.

To tell the check box which event function to call when it is clicked, use the SetEvent_OnClick method.

The OnClick event function may have any name, but must have the following format:

```
Sub checkbox_OnClick()
  ' Do your processing here
End Sub
```

Parameters

This event function has no parameters.

Return value

This event function has no return value.

7.2.6 Radio buttons

Radio buttons give the user the ability to select one of a number of choices.

All the choices in a group of radio buttons are mutually exclusive.

By default, all the radio buttons on a form are in the same group. To find out how to specify more groups, see StartButtonGroup.

Creation

A radio button can be created using the form's AddRadioButton method. This method takes as one of its parameters the name of the control, which is then used throughout the script when referring to this radio button.

For example

```
form.AddRadioButton "radio2", "On/Off"
```

would mean that all properties and methods of this radio button would be called by simply prefixing them with

```
form.radio2.
```

Properties

Visible
Enabled
XPos
YPos
Width
Height
Text
TabStop
Checked

Methods

SetTextColour
SetBackgroundColour
SetFont
SetEvent_OnClick

Events

OnClick

7.2.6.1 Properties

7.2.6.1.1 Visible

Description

This property specifies whether the radio button should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A radio button is visible by default.

NB: **An invisible radio button cannot fire events.**

Values

True	Sets the radio button to be visible.
False	Sets the radio button to be invisible.

7.2.6.1.2 Enabled

Description

This property specifies whether the radio button should be enabled or disabled.

A radio button is enabled by default.

NB: **A disabled radio button cannot fire any events.**

Values

True	Sets the radio button to be enabled.
False	Sets the radio button to be disabled.

7.2.6.1.3 XPos

Description

This property represents the X position of the radio button, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddRadioButton is used, then the radio button will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.6.1.4 YPos

Description

This property represents the Y position of the radio button, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddRadioButton is used, then the radio button will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.6.1.5 Width

Description

This property represents the width of the radio button, in pixels.

If this property is not explicitly specified, either using this property or when AddRadioButton is used, then the control will be sized to fit the current text and radio button part (specified using Text, or originally with AddCheckBox). In this case, any changes to the text may result in the radio button changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.6.1.6 Height

Description

This property represents the height of the radio button, in pixels.

If this property is not explicitly specified, either using this property or when AddRadioButton is used, then the radio button will be sized to fit the current text (specified using Text, or originally with AddRadioButton). In this case, any changes to the text may result in the radio button changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.6.1.7 Text

Description

This property represents the text on the radio button.

Values

Any string can be used as the check box's text.

7.2.6.1.8 TabStop

Description

This property specifies whether the radio button should have the "TabStop" property.

A radio button with the TabStop property will get the focus as the TAB key is used to move through the controls on the form.

By default, radio buttons have the TabStop property set to **True**.

Values

True	Sets the radio button to have the TabStop property.
False	Specifies that the radio button should not have the TabStop property.

7.2.6.1.9 Checked

Description

This property sets or returns the current checked state of the radio button.

NB: **Only one radio button in a group acn be checked at once. To find out how to group radio buttons, see StartButtonGroup.**

Values

True	The radio button is enabled.
False	The radio button is disabled.

7.2.6.1.10 TooltipText

Description

This property represents the Tooltip text for the radio button. This is text displayed in a small pop-up window when the mouse pointer is held over the control. It is usually used to contain a brief description of the purpose of the control.

Values

Any string can be used as the Tooltip text, up to a maximum of 80 characters.

7.2.6.2 Methods

7.2.6.2.1 SetTextColour

SetTextColour (sRed, sGreen, sBlue)

This method is used to set the text colour of this radio button.

Parameters

sRed	The red component of the colour to set the text to (0 - 255).
sGreen	The green component of the colour to set the text to (0 - 255).
sBlue	The blue component of the colour to set the text to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set two radio buttons' text colours to blue and green, respectively.

```
form1.radio1.SetTextColour 0, 0, 255
form1.radio2.SetTextColour 0, 255, 0
```

7.2.6.2.2 SetBackgroundColour

SetBackgroundColour (sRed, sGreen, sBlue)

This method is used to set the background colour of this radio button.

Parameters

sRed	The red component of the colour to set the background to (0 - 255).
sGreen	The green component of the colour to set the background to (0 - 255).
sBlue	The blue component of the colour to set the background to (0 - 255).

NB: **To get the red/green/blue components of colours, you can use any program that allows colour changing. For example, open the Paint program that comes with Windows, double-click on a colour from the palette at the bottom of the screen, and select "Define Custom Colors".**

Return value

This method has no return value.

Example

The following code will set two radio buttons' background colours to white and black, respectively.

```
form1.radio1.SetBackgroundColour 255, 255, 255
form1.radio2.SetBackgroundColour 0, 0, 0
```

7.2.6.2.3 SetFont

SetFont (strFontName, sSize, sStyle)

This method can be called to set the font of the text on a radio button.

Parameters

strFontName	The name of the font. This can be any one of the standard Windows font names, such as "Arial", "Courier", etc.
sSize	The size of the font, in points. The default font size is the same as the system font (usually 10-points).
sStyle	The style of the font to set. It can have any combination of the styles listed under Font Styles below.

Return value

This method has no return value.

Font Styles

The following font styles, or combinations of styles, are allowed for the ***sStyle*** parameter:

fontBold	**1**	Sets the font to be **Bold**.
fontItalic	**2**	Sets the font to be *italic*.

Example

Use the following line to set the font of a radio button to a 12-point, bold and italic, "Arial" font.

```
form.radio3.SetFont "Arial", 12, 3
```

7.2.6.2.4 SetFocus

SetFocus ()

This method can be called to set the focus to this radio button (i.e. make it the control that currently handles keyboard input). This means that pressing the space bar will toggle whether the radio button is currently checked.

NB: **Giving the focus to a control will remove the focus from any control which has previously been given it.**

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

Use the following line to set the focus to a previously-created radio button.

```
form.Radio2.SetFocus()
```

7.2.6.2.5 HasFocus

HasFocus ()

This method can be called to determine whether a control currently has the focus.

Parameters

This method has no parameters.

Return value

This method returns **True** .

Example

Use the following line to determine whether a previously-created radio button has the focus. This can be used in an OnClick event handler, for example, to determine which radio button was actually clicked.

```
If form.MyRadioButton.HasFocus() Then
    ' Do something here
End If
```

7.2.6.2.6 SetEvent_OnClick

SetEvent_OnClick (eventref)

This method is used to specify the function that will be called when a radio button is clicked.

See OnClick event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the check box is clicked. This must be passed in the form **GetRef ("radio3_OnClick")**.

Return value

This method has no return value.

Example

The following code snippet will create a form, initialize it so that events will be fired, create two radio b uttons, and set an event that is fired when the first radio button is clicked.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.AddRadioButton "radio1", "Square"
form1.AddRadioButton "radio2", "Circle"
form1.radio1.SetEvent_OnClick GetRef("radio1_OnClick")

form1.Display()

Sub radio1_OnClick()
    MsgBox "All squared up!"
End Sub
```

7.2.6.3 Events

7.2.6.3.1 OnClick event

The OnClick event will be fired when a radio button is clicked.

To tell the radio button which event function to call when it is clicked, use the SetEvent_OnClick method.

The OnClick event function may have any name, but must have the following format:

```
Sub radiobutton_OnClick()
    ' Do your processing here
End Sub
```

Parameters

This event function has no parameters.

Return value

This event function has no return value.

7.2.7 Slider controls

Slider controls allow the user to make a selection by sliding a pointer along a scale.

Creation

A slider control can be created using the form's AddSlider method. This method takes as one of its parameters the name of the slider, which is then used throughout the script when referring to this control.

For example

```
form.AddSlider "slider", 100, 20
```

would mean that all properties and methods of this slider control would be called by simply prefixing them with

```
form.slider.
```

Properties

Visible
Enabled
XPos
YPos
Width
Height
TabStop
Vertical
CurPos
NumTicks
TooltipText

Methods

SetRange
GetRange
SetFocus
SetEvent_OnPosChanged

Events

OnPosChanged

7.2.7.1 Properties

7.2.7.1.1 Visible

Description

This property specifies whether the slider control should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A slider control is visible by default.

NB: **An invisible slider control cannot fire events.**

Values

True	Sets the slider control to be visible.
False	Sets the slider control to be invisible.

7.2.7.1.2 Enabled

Description

This property specifies whether the slider control should be enabled or disabled.

A slider control is enabled by default.

NB: **A disabled slider control cannot fire any events.**

Values

True	Sets the slider control to be enabled.
False	Sets the slider control to be disabled.

7.2.7.1.3 XPos

Description

This property represents the X position of the slider control, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddSlider is used, then the slider will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.7.1.4 YPos

Description

This property represents the Y position of the slider control, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddSlider is used, then the slider will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.7.1.5 Width

Description

This property represents the width of the slider control, in pixels.

If this property is not explicitly specified, either using this property or when AddSlider is used, then the slider control will be sized automatically. If the slider is horizontal (see Vertical property), the size will depend on the current range (see SetRange) and the number of ticks.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.7.1.6 Height

Description

This property represents the height of the slider control, in pixels.

If this property is not explicitly specified, either using this property or when AddSlider is used, then the slider control will be sized automatically. If the slider is vertical (see Vertical property), the size will depend on the current range (see SetRange) and the number of ticks.

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.7.1.7 TabStop

Description

This property specifies whether the slider control should have the "TabStop" property.

A slider control with the TabStop property will get the focus as the TAB key is used to move through the controls on the form.

By default, slider controls have the TabStop property set to **True**.

Values

True	Sets the slider control to have the TabStop property.
False	Specifies that the slider control should not have the TabStop property.

7.2.7.1.8 Vertical

Description

This property specifies whether the slider control should be vertical or horizontal.

A slider is horizontal by default.

Values

True	Sets the slider to be vertical
False	Sets the slider to be horizontal.

7.2.7.1.9 CurPos

Description

This property sets or returns the current position of the slider control.

Values

Any whole number within the limits specified by SetRange is allowed.

7.2.7.1.10 NumTicks

Description

This property sets or returns the number of ticks on the slider control.

Values

Any whole number between 2 and 1000 is allowed.

7.2.7.1.11 TooltipText

Description

This property represents the Tooltip text for the slider control. This is text displayed in a small pop-up window when the mouse pointer is held over the control. It is usually used to contain a brief description of the purpose of the control.

Values

Any string can be used as the Tooltip text, up to a maximum of 80 characters.

7.2.7.2 Methods

7.2.7.2.1 SetRange

SetRange (sMin, sMax)

This method is used to set the range of values for a slider control.

Parameters

sMin	The start of the range to set
sMax	The end of the range to set.

Return value

This method has no return value.

7.2.7.2.2 GetRange

GetRange (sMin, sMax)

This method is used to get the current range of values of a slider control.

Parameters

sMin	After this method is called, this parameter will hold the slider control's current minimum value.
sMax	After this method is called, this parameter will hold the slider control's current maximum value.

Return value

This method has no return value.

7.2.7.2.3 SetFocus

SetFocus ()

This method can be called to set the focus to this slider (i.e. make it the control that currently handles keyboard input). This means that using the arrow keys, or Page Up and Page Down, will cause the slider control's position to change.

NB: **Giving the focus to a control will remove the focus from any control which has previously been given it.**

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

Use the following line to set the focus to a previously-created slider control.

```
form.FrequencySlider.SetFocus()
```

7.2.7.2.4 SetEvent_OnPosChanged

SetEvent_OnPosChanged (eventref)

This method is used to specify the function that will be called when the slider's position is changed..

See OnPosChanged event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the slider's position changes. This must be passed in the form **GetRef ("button3_OnClick")**.

Return value

This method has no return value.

Example

The following code snippet will create a form, initialize it so that events will be fired, create a slider, and set an event that is fired when the slider's position changes.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.AddSlider "slider1", 300, 40, 5, 5
form1.slider1.SetEvent_OnPosChanged
     GetRef("slider1_OnPosChanged")
form1.slider1.SetRange 0, 50
form1.slider1.CurPos = 25

form1.Display()

Sub slider1_OnPosChanged()
  MsgBox "Current slider position is " &
     form1.slider1.CurPos
End Sub
```

7.2.7.3 Events

7.2.7.3.1 OnPosChanged event

The OnPosChanged event will be fired when a slider's position changes.

To tell the slider control which event function to call when its position changes, use the

SetEvent_OnPosChanged method.

The OnPosChanged event function may have any name, but must have the following format:

```
Sub slider_OnPosChanged(nPos)
    ' Do your processing here
End Sub
```

Parameters

This event function has a single parameter which represents the type of position change:

0	Left / Up (the slider's position was moved by one place left (horizontal controls) or one place up (vertical controls)
1	Right / Down (the slider's position was moved by one place right (horizontal controls) or one place down (vertical controls)
2	Page Left / Page Up (the slider's position was moved by one page left (horizontal controls) or one page up (vertical controls)
3	Page Right / Page Down (the slider's position was moved by one page right (horizontal controls) or one page down (vertical controls)
4	The slider has been moved using the mouse, and has now finished moving. Use the CurPos property to get the current position.
5	The slider is being moved using the mouse. Use the CurPos property to get the current position.

Return value

This event function has no return value.

7.2.8 Drop-list controls

Drop-list controls allow the user to make a selection by choosing an item from a list that drops down to reveal the choices available.

Creation

A drop-list control can be created using the form's AddDropList method. This method takes as one of its parameters the name of the drop-list, which is then used throughout the script when referring to this control.

For example

```
form.AddDropList "Selection1", 100, -1
```

would mean that all properties and methods of this drop-list control would be called by simply prefixing them with

```
form.Selection1.
```

Properties

Visible
Enabled
XPos
YPos
Width

TabStop
Sorted
NumStrings
CurSel
TooltipText

Methods

SetEvent_OnSelChanged
AddString
GetString
RemoveAllStrings
DeleteString
SetItemData
GetItemData

Events

OnSelChanged

7.2.8.1 Properties

7.2.8.1.1 Visible

Description

This property specifies whether the drop-list control should be visible or invisible.

This can be useful if different selections in other controls (for example, other drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A drop-list control is visible by default.

Values

True	Sets the drop-list control to be visible.
False	Sets the drop-list control to be invisible.

7.2.8.1.2 Enabled

Description

This property specifies whether the drop-list control should be enabled or disabled.

A drop-list control is enabled by default.

Values

True	Sets the drop-list control to be enabled.
False	Sets the drop-list control to be disabled.

7.2.8.1.3 XPos

Description

This property represents the X position of the drop-list control, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddDropList is used, then the drop-list control will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.8.1.4 YPos

Description

This property represents the Y position of the drop-list control, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddDropList is used, then the drop-list control will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.8.1.5 Width

Description

This property represents the width of the drop-list control, in pixels.

If this property is not explicitly specified, either using this property or when AddDropList is used, then the drop-list control will be sized to fit the widest text entered in the drop-list. In this case, adding or deleting strings (using AddString, DeleteString or RemoveAllStrings) may result in the drop-list control changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.8.1.6 TabStop

Description

This property specifies whether the drop-list control should have the "TabStop" property.

A drop-list control with the TabStop property will get the focus as the TAB key is used to move through the controls on the form.

By default, drop-list controls have the TabStop property set to **True**.

Values

True	Sets the drop-list control to have the TabStop property.
False	Specifies that the drop-list control should not have the TabStop property.

7.2.8.1.7 Sorted

Description

This property specifies whether the drop-list control should be sorted.

If the drop-list control is sorted, then strings added to the control will be displayed in alphabetical order in the drop-list, regardless of the order in which they are added. Otherwise, strings will be displayed in the order that they were added.

By default, drop-list controls are not sorted..

Values

True	Sets the drop-list control to be sorted.
False	Specifies that the drop-list control should not be sorted.

7.2.8.1.8 NumStrings

Description

This **read-only** property can be used to find out the number of strings in the drop-list control.

NB: **The index of the last string in the drop-list will be (NumStrings-1).**

7.2.8.1.9 CurSel

Description

This property represents the zero-based index of the currently selected string in a drop-list.

Values

Any numerical value is valid, between 0 and NumStrings-1 inclusive..

7.2.8.1.10 TooltipText

Description

This property represents the Tooltip text for the drop-list control. This is text displayed in a small pop-up window when the mouse pointer is held over the control. It is usually used to contain a brief description of the purpose of the control.

Values

Any string can be used as the Tooltip text, up to a maximum of 80 characters.

7.2.8.2 Methods

7.2.8.2.1 AddString

AddString (strString [, ItemData])

This method is used to add a string to a drop-list.

NB: **If the drop-list is sorted, this string will be inserted into the list in alphabetical order once the control has been created. Otherwise, it will be added at the end of the list.**

Parameters

strString	The string to add to the drop-list.
lItemData	Optional parameter specifying a long integer which can be associated with this string in the list. For example, in a list of sample rates, you may wish to associate the sample rate itself with the string containing its description (see example below).

Return value

This method has no return value.

Example

The following example creates a drop-list and adds a series of sample rate selections to it. Each entry contains the actual sample rate as its item data.

```
form1.AddDropList "SampleRate", 120
form1.SampleRate.SetEvent_OnSelChanged GetRef ("SampleRate_OnSelChanged")

form1.SampleRate.AddString "32kHz",   32000
form1.SampleRate.AddString "44.1kHz", 44100
form1.SampleRate.AddString "48kHz",   48000
form1.SampleRate.AddString "88.2kHz", 88200
form1.SampleRate.AddString "96kHz",   96000

Sub SampleRate_OnSelChanged
   sCurSel = form1.SampleRate.CurSel
   form1.SampleRate.GetItemData sCurSel, lSampleRate
   MsgBox "You selected a sample rate of " & lSampleRate
End Sub
```

7.2.8.2.2 GetString

GetString (sIndex, strString)

This method is used to get the string at the given index in a drop-list.

NB: **If the drop-list is sorted, this string will be inserted into the list in alphabetical order once the control has been created. Otherwise, it will be added at the end of the list.**

Parameters

sIndex	The zero-based index into the drop-list to get the string at. This must be a number between 0 and NumStrings-1.
strString	After this method is called, this parameter will hold the string at the given index in the drop-list.

Return value

This method has no return value.

7.2.8.2.3 DeleteString

DeleteString (sIndex)

This method is used to delete the string at the given index in the drop-list.

NB: **If the string deleted is the currently selected string, the next string in the drop-list will be selected automatically (or the previous string, if it was the last one in the list).**

Parameters

sIndex	The zero-based index into the drop-list to delete the string at. This must be a number between 0 and NumStrings-1.

Return value

This method has no return value.

7.2.8.2.4 RemoveAllStrings

RemoveAllStrings ()

This method is used to delete all strings from a drop-list.

Parameters

This method has no parameters.

Return value

This method has no return value.

7.2.8.2.5 SetItemData

SetItemData (sIndex, lItemData)

This method is used to set the item data for a given entry in the drop-list.

Item data is a four-byte long integer value that can be associated with a string in the drop-list. You may wish to use it to store further details about the entry referred to by the string.

Parameters

sIndex	The zero-based index into the drop-list to set the item data for. This must be a number between 0 and NumStrings-1.
lItemData	After this method is called, this parameter will hold the string at the given index in the drop-list.

Return value

This method has no return value.

7.2.8.2.6 GetItemData

GetItemData (sIndex, lItemData)

This method is used to get the item data for a given entry in the drop-list.

Parameters

sIndex	The zero-based index into the drop-list to set the item data for. This must be a number between 0 and NumStrings-1.
lItemData	After this method is called, this parameter will hold the item data for the string at the given index in the drop-list.

Return value

This method has no return value.

7.2.8.2.7 SetFocus

SetFocus ()

This method can be called to set the focus to this drop-list (i.e. make it the control that currently handles keyboard input). This means that using the arrow keys, or Page Up and Page Down, will cause the current selection in the drop-list to change.

NB: **Giving the focus to a control will remove the focus from any control which has previously been given it.**

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

Use the following line to set the focus to a previously-created drop-list control.

```
form.MyCombo1.SetFocus()
```

7.2.8.2.8 HasFocus

HasFocus ()

This method can be called to determine whether a control currently has the focus.

Parameters

This method has no parameters.

Return value

This method returns **True** .

Example

Use the following line to determine whether a previously-created drop-list has the focus.

```
If form.MyDropList.HasFocus() Then
    ' Do something here
End If
```

7.2.8.2.9 SetEvent_OnSelChanged

SetEvent_OnSelChanged (eventref)

This method is used to specify the function that will be called when the selection in the drop-list is changed.

See OnSelChanged event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the drop-list's selection is changed. This must be passed in the form **GetRef ("DropList1_OnSelChange")**.

Return value

This method has no return value.

Example

The following code snippet will create a form, initialize it so that events will be fired, create a drop-list, add some strings to it, and set an event that is fired when the current selection changes.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.AddDropList "DropList1", 200, 5, 5
form1.DropList1.SetEvent_OnSelChanged
    GetRef("DropList1_OnSelChanged")
form1.DropList1.AddString "item 1"
form1.DropList1.AddString "item 2"
form1.DropList1.AddString "item 3"

form1.Display()

Sub DropList1_OnPosChanged()
  MsgBox "Current selection is " &
    form1.DropList1.CurSel
End Sub
```


7.2.8.3 Events

7.2.8.3.1 OnSelChanged event

The OnSelChanged event will be fired when the user changes the selection in a drop-list control. The item selected can be found using the CurSel property

To tell the drop-list control which event function to call when its position changes, use the SetEvent_OnSelChanged method.

The OnSelChanged event function may have any name, but must have the following format:

```
Sub droplist_OnSelChanged()
     ' Do your processing here
End Sub
```

Parameters

This event function has no parameters.

Return value

This event function has no return value.

7.2.9 List box controls

List box controls allow the user to make a selection by choosing an item from a list.

Creation

A list box control can be created using the form's AddListBox method. This method takes as one of its parameters the name of the list box, which is then used throughout the script when referring to this control.

For example

```
form.AddListBox "MyListBox", 200, 100
```

would mean that all properties and methods of this list box control would be called by simply prefixing them with

```
form.MyListBox.
```

Properties

Visible
Enabled
XPos
YPos
Width
TabStop
Sorted
NumStrings
CurSel
TooltipText

Methods

SetEvent_OnSelChanged
SetEvent_OnDoubleClick
AddString
GetString
RemoveAllStrings
DeleteString
SetItemData
GetItemData

Events

OnSelChanged
OnDoubleClick

7.2.9.1 Properties

7.2.9.1.1 Visible

Description

This property specifies whether the list box control should be visible or invisible.

This can be useful if different selections in other controls (for example, other list boxes or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A list box control is visible by default.

Values

True	Sets the list box control to be visible.
False	Sets the list box control to be invisible.

7.2.9.1.2 Enabled

Description

This property specifies whether the list box control should be enabled or disabled.

A list box control is enabled by default.

Values

True	Sets the list box control to be enabled.
False	Sets the list box control to be disabled.

7.2.9.1.3 XPos

Description

This property represents the X position of the list box control, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddListBox is used, then the list box control will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.9.1.4 YPos

Description

This property represents the Y position of the list box control, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddListBox is used, then the list box control will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.9.1.5 Width

Description

This property represents the width of the list box control, in pixels.

If this property is not explicitly specified, either using this property or when AddListBox is used, then the list box control will be sized to fit the widest text entered in the list box. In this case, adding or deleting strings (using AddString, DeleteString or RemoveAllStrings) may result in the list box control changing size.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.9.1.6 TabStop

Description

This property specifies whether the list box control should have the "TabStop" property.

A list box control with the TabStop property will get the focus as the TAB key is used to move through the controls on the form.

By default, list box controls have the TabStop property set to **True**.

Values

True	Sets the list box control to have the TabStop property.
False	Specifies that the list box control should not have the TabStop property.

7.2.9.1.7 Sorted

Description

This property specifies whether the list box control should be sorted.

If the list box control is sorted, then strings added to the control will be displayed in alphabetical order in the list box, regardless of the order in which they are added. Otherwise, strings will be displayed in the order that they were added.

By default, list box controls are not sorted..

Values

True	Sets the list box control to be sorted.
False	Specifies that the list box control should not be sorted.

7.2.9.1.8 NumStrings

Description

This **read-only** property can be used to find out the number of strings in the list box control.

NB: **The index of the last string in the list box will be (NumStrings-1).**

7.2.9.1.9 CurSel

Description

This property represents the zero-based index of the currently selected string in a list box.

Values

Any numerical value is valid, between 0 and NumStrings-1 inclusive..

7.2.9.1.10 TooltipText

Description

This property represents the Tooltip text for the list box control. This is text displayed in a small pop-up window when the mouse pointer is held over the control. It is usually used to contain a brief description of the purpose of the control.

Values

Any string can be used as the Tooltip text, up to a maximum of 80 characters.

7.2.9.2 Methods

7.2.9.2.1 SetEvent_OnSelChanged

SetEvent_OnSelChanged (eventref)

This method is used to specify the function that will be called when the selection in the list box is changed.

See OnSelChanged event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the list box's selection is changed. This must be passed in the form **GetRef ("ListBox1_OnSelChange")**.

Return value

This method has no return value.

Example

The following code snippet will create a form, initialize it so that events will be fired, create a list box, add some strings to it, and set an event that is fired when the current selection changes.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.AddListBox "MyList1", 200, 5, 5
form1.MyList1.SetEvent_OnSelChanged
     GetRef("MyList1_OnSelChanged")
form1.MyList1.AddString "item 1"
form1.MyList1.AddString "item 2"
form1.MyList1.AddString "item 3"

form1.Display()

Sub MyList1_OnPosChanged()
  MsgBox "Current selection is " &
     form1.MyList1.CurSel
End Sub
```

7.2.9.2.2 SetEvent_OnDoubleClick

SetEvent_OnDoubleClick (eventref)

This method is used to specify the function that will be called when an item in the list box is double-clicked.

See OnSelChanged event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the list box's selection is changed. This must be passed in the form **GetRef ("MyList1_OnSelChange")**.

Return value

This method has no return value.

Example

The following code snippet will create a form, initialize it so that events will be fired, create a list box, add some strings to it, and set an event that is fired when an item is double-clicked.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.AddDropList "MyList1", 200, 5, 5
form1.MyList1.SetEvent_OnDoubleClick
     GetRef("MyList1_OnDoubleClick")
```

```
form1.MyList1.AddString "item 1"
form1.MyList1.AddString "item 2"
form1.MyList1.AddString "item 3"

form1.Display()

Sub MyList1_OnPosChanged()
  MsgBox "Current selection is " &
     form1.MyList1.CurSel
End Sub
```

7.2.9.2.3 AddString

AddString (strString [, ItemData])

This method is used to add a string to a list box.

NB: **If the list box is sorted, this string will be inserted into the list in alphabetical order once the control has been created. Otherwise, it will be added at the end of the list.**

Parameters

strString	The string to add to the list box.
lItemData	Optional parameter specifying a long integer which can be associated with this string in the list. For example, in a list of sample rates, you may wish to associate the sample rate itself with the string containing its description (see example below).

Return value

This method has no return value.

Example

The following example creates a list box and adds a series of sample rate selections to it. Each entry contains the actual sample rate as its item data.

```
form1.AddListBox "SampleRate", 120
form1.SampleRate.SetEvent_OnSelChanged GetRef ("SampleRate_OnSelChanged")

form1.SampleRate.AddString "32kHz",   32000
form1.SampleRate.AddString "44.1kHz", 44100
form1.SampleRate.AddString "48kHz",   48000
form1.SampleRate.AddString "88.2kHz", 88200
form1.SampleRate.AddString "96kHz",   96000

Sub SampleRate_OnSelChanged
   sCurSel = form1.SampleRate.CurSel
   form1.SampleRate.GetItemData sCurSel, lSampleRate
   MsgBox "You selected a sample rate of " & lSampleRate
End Sub
```


7.2.9.2.4 GetString

GetString (sIndex, strString)

This method is used to get the string at the given index in a list box.

NB: **If the list box is sorted, this string will be inserted into the list in alphabetical order once the control has been created. Otherwise, it will be added at the end of the list.**

Parameters

sIndex	The zero-based index into the list box to get the string at. This must be a number between 0 and NumStrings-1.
strString	After this method is called, this parameter will hold the string at the given index in the list box.

Return value

This method has no return value.

7.2.9.2.5 DeleteString

DeleteString (sIndex)

This method is used to delete the string at the given index in the list box.

NB: **If the string deleted is the currently selected string, the next string in the list box will be selected automatically (or the previous string, if it was the last one in the list).**

Parameters

sIndex	The zero-based index into the list box to delete the string at. This must be a number between 0 and NumStrings-1.

Return value

This method has no return value.

7.2.9.2.6 RemoveAllStrings

RemoveAllStrings ()

This method is used to delete all strings from a list box.

Parameters

This method has no parameters.

Return value

This method has no return value.

7.2.9.2.7 SetItemData

SetItemData (sIndex, lItemData)

This method is used to set the item data for a given entry in the list box.

Item data is a four-byte long integer value that can be associated with a string in the list box. You may wish to use it to store further details about the entry referred to by the string.

Parameters

sIndex	The zero-based index into the list box to set the item data for. This must be a number between 0 and NumStrings-1.
lItemData	After this method is called, this parameter will hold the string at the given index in the list box.

Return value

This method has no return value.

7.2.9.2.8 GetItemData

GetItemData (sIndex, lItemData)

This method is used to get the item data for a given entry in the list box.

Parameters

sIndex	The zero-based index into the list box to set the item data for. This must be a number between 0 and NumStrings-1.
lItemData	After this method is called, this parameter will hold the item data for the string at the given index in the list box.

Return value

This method has no return value.

7.2.9.2.9 SetFocus

SetFocus ()

This method can be called to set the focus to this list box (i.e. make it the control that currently handles keyboard input). This means that using the arrow keys, or Page Up and Page Down, will cause the current selection in the list box to change.

NB: **Giving the focus to a control will remove the focus from any control which has previously been given it.**

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

Use the following line to set the focus to a previously-created list box control.

```
form.List1.SetFocus()
```

7.2.9.2.10 HasFocus

HasFocus ()

This method can be called to determine whether a control currently has the focus.

Parameters

This method has no parameters.

Return value

This method returns **True** .

Example

Use the following line to determine whether a previously-created list box has the focus.

```
If form.MyListBox.HasFocus() Then
     ' Do something here
End If
```

7.2.9.3 Events

7.2.9.3.1 OnSelChanged event

The OnSelChanged event will be fired when the user changes the selection in a list box control. The item selected can be found using the CurSel property.

To tell the drop-list control which event function to call when its position changes, use the SetEvent_OnSelChanged method.

The OnSelChanged event function may have any name, but must have the following format:

```
Sub listBox_OnSelChanged()
    ' Do your processing here
End Sub
```

Parameters

This event function has no parameters.

Return value

This event function has no return value.

7.2.9.3.2 OnDoubleClick event

The OnDoubleClick event will be fired when the user double-clicks on an item in a list box control. The item that is clicked on will be the current selection and can be found using the CurSel property.

To tell the list box control which event function to call when its position changes, use the SetEvent_OnDoubleClick method.

The OnDoubleClick event function may have any name, but must have the following format:

```
Sub list_OnDoubleClick()
    ' Do your processing here
End Sub
```

Parameters

This event function has no parameters.

Return value

This event function has no return value.

7.2.10 Bitmap controls

Bitmap controls are simple controls that display a picture specified by a bitmap (*.bmp) file.

Creation

A bitmap control can be created using the form's AddBitmap method. This method takes as one of its parameters the name of the bitmap control, which is then used throughout the script when referring to this control.

For example

```
form.AddBitmap "bitmap1",
    "C:\My Documents\MyPicture.bmp"
```

would mean that all properties and methods of this bitmap control would be called by simply prefixing them with

```
form.bitmap1.
```

Properties

Visible
XPos
YPos
Width
Height

Methods

SetBitmap

Events

Bitmap controls do not fire events.

7.2.10.1 Properties

7.2.10.1.1 Visible

Description

This property specifies whether the bitmap control should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A bitmap control is visible by default.

Values

True	Sets the bitmap control to be visible.
False	Sets the bitmap control to be invisible.

7.2.10.1.2 XPos

Description

This property represents the X position of the bitmap control, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddBitmap is used, then the bitmap control will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.10.1.3 YPos

Description

This property represents the Y position of the bitmap control, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddBitmap is used, then the bitmap control will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.10.1.4 Width

Description

This property represents the width of the bitmap control, in pixels.

If this property is not explicitly specified, either using this property or when AddBitmap is used, then the bitmap control will be sized to fit the bitmap. In this case, selecting a different bitmap in this control may result in the bitmap control changing size.

If this property is specified, then the bitmap will be stretched or shrunk to fit into the width specified.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.10.1.5 Height

Description

This property represents the height of the bitmap control, in pixels.

If this property is not explicitly specified, either using this property or when AddBitmap is used, then the bitmap control will be sized to fit the bitmap. In this case, selecting a different bitmap in this control may result in the bitmap control changing size.

If this property is specified, then the bitmap will be stretched or shrunk to fit into the height specified.

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.10.2 Methods

7.2.10.2.1 SetBitmap

SetBitmap (strBitmapFile)

This method is used to change the bitmap in this control.

Parameters

strBitmapFile	The bitmap file to show in this control.

Return value

This method has no return value.

7.2.11 Progress controls

Progress controls are simple controls that show the progress of an operation.

Creation

A progress control can be created using the form's AddProgress method. This method takes as one of its parameters the name of the progress control, which is then used throughout the script when referring to this control.

For example

```
form.AddProgress "Progress1"
```

would mean that all properties and methods of this progress control would be called by simply prefixing them with

```
form.Progress1.
```

Properties

Visible
XPos
YPos
Width
Height
Vertical
Range
CurPos

Methods

Step

Events

Progress controls do not fire events.

7.2.11.1 Properties

7.2.11.1.1 Visible

Description

This property specifies whether the progress control should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A progress control is visible by default.

Values

True	Sets the progress control to be visible.
False	Sets the progress control to be invisible.

7.2.11.1.2 XPos

Description

This property represents the X position of the progress control, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddProgress is used, then the progress control will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.11.1.3 YPos

Description

This property represents the Y position of the progress button, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddProgress is used, then the progress control will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB:

If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.11.1.4 Width

Description

This property represents the width of the progress control, in pixels.

If this property is not explicitly specified, either using this property or when AddProgress is used, then the button will be set to a nominal size.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.11.1.5 Height

Description

This property represents the height of the progress control, in pixels.

If this property is not explicitly specified, either using this property or when AddProgress is used, then the progress control will be set to a nominal size.

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.11.1.6 Vertical

Description

This property specifies whether the progress control should be vertical or horizontal.

A progress control is horizontal by default.

Values

True	Sets the progress control to be vertical
False	Sets the progress control to be horizontal.

7.2.11.1.7 Range

Description

This property sets or returns the range of the progress control. The current position of the control (see CurPos) can be any number between 0 and this value.

The default range of the progress control is 10.

Values

Any whole number between 1 and 1000 is allowed.

7.2.11.1.8 CurPos

Description

This property sets or returns the current position of the progress control. It can be any number between 0 and the range of the control (see Range).

The default position of the progress control is 0.

Values

Any whole number between 0 and the range of the control is allowed.

7.2.11.2 Methods

7.2.11.2.1 Step

Step()

This method is used to step the progress control by a single step. If this would take it beyond the range specified by the Range property, the current position (See CurPos) is set to 0.

Parameters

This method has no parameters.

Return value

This method has no return value.

Example

The following code will step the progress of the progress control to the next step.

```
form1.Progress1.Step()
```

7.2.12 ScrollBar controls

ScrollBar controls allow the user to scroll along a ScriptDlg, either to change a value according to the position of the ScrollBar, or to allow a ScriptDlg to display controls that cannot all be fitted onto the form at the same time.

Creation

A ScrollBar control can be created using the form's AddScrollBar method. This method takes as one of its parameters the name of the ScrollBar, which is then used throughout the script when referring to this control.

For example

```
form.AddScrollBar "HScroll", 100, 20
```

would mean that all properties and methods of this ScrollBar control would be called by simply prefixing them with

```
form.HScroll.
```

Properties

Visible
Enabled
XPos
YPos
Width
Height
TabStop
Vertical
CurPos
TooltipText
PageSize

Methods

SetRange
GetRange
SetEvent_OnPosChanged

Events

OnPosChanged

7.2.12.1 Properties

7.2.12.1.1 Visible

Description

This property specifies whether the ScrollBar control should be visible or invisible.

This can be useful if different selections in other controls (for example, drop-lists or radio buttons) require a different set of controls - the unwanted controls can be hidden, and the required controls shown, depending on the selection.

A ScrollBar control is visible by default.

NB: **An invisible ScrollBar control cannot fire events.**

Values

True	Sets the ScrollBar control to be visible.
False	Sets the ScrollBar control to be invisible.

7.2.12.1.2 Enabled

Description

This property specifies whether the ScrollBar control should be enabled or disabled.

A ScrollBar control is enabled by default.

NB: **A disabled ScrollBar control cannot fire any events.**

Values

True	Sets the ScrollBar control to be enabled.
False	Sets the ScrollBar control to be disabled.

7.2.12.1.3 XPos

Description

This property represents the X position of the ScrollBar control, in pixels from the left hand side of the form.

If this property is not explicitly set, either using this property or when AddScrollBar is used, then the ScrollBar will be positioned to the right of the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the X position specified is greater than the current width of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the X position is less than zero, the control's X position will start beyond the left-hand side of the form.

7.2.12.1.4 YPos

Description

This property represents the Y position of the ScrollBar control, in pixels from the top of the form (not including the title bar of the form).

If this property is not explicitly set, either using this property or when AddScrollBar is used, then the ScrollBar will be positioned at the same Y position as the previous control, unless the NewRow method has been called since the previous control was added.

Values

Any numerical value can be used.

NB: **If the Y position specified is greater than the current height of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

If the Y position is less than zero, the control's X position will start above the top of the form.

7.2.12.1.5 Width

Description

This property represents the width of the ScrollBar control, in pixels.

If this property is not explicitly specified, either using this property or when AddScrollBar is used, then the ScrollBar control will be sized automatically. If the ScrollBar is horizontal (see Vertical property), the size will depend on the current range (see SetRange) and the number of ticks.

Values

Any numerical value greater than 0 can be used.

NB: **If the width specified will take this control beyond the right hand side of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.12.1.6 Height

Description

This property represents the height of the ScrollBar control, in pixels.

If this property is not explicitly specified, either using this property or when AddScrollBar is used, then the ScrollBar control will be sized automatically. If the ScrollBar is vertical (see Vertical property), the size will depend on the current range (see SetRange) and the number of ticks.

Values

Any numerical value greater than 0 can be used.

NB: **If the height specified will take this control beyond the bottom of the form, and the form is set to be automatically sizeable, then the form will be resized to include this control.**

7.2.12.1.7 TabStop

Description

This property specifies whether the ScrollBar control should have the "TabStop" property.

A ScrollBar control with the TabStop property will get the focus as the TAB key is used to move through the controls on the form.

By default, ScrollBar controls have the TabStop property set to **True**.

Values

True	Sets the ScrollBar control to have the TabStop property.
False	Specifies that the ScrollBar control should not have the TabStop property.

7.2.12.1.8 Vertical

Description

This property specifies whether the ScrollBar control should be vertical or horizontal.

A ScrollBar is horizontal by default.

Values

True	Sets the ScrollBar to be vertical
False	Sets the ScrollBar to be horizontal.

7.2.12.1.9 CurPos

Description

This property sets or returns the current position of the ScrollBar control.

Values

Any whole number within the limits specified by SetRange is allowed.

7.2.12.1.10 TooltipText

Description

This property represents the Tooltip text for the ScrollBar control. This is text displayed in a small pop-up window when the mouse pointer is held over the control. It is usually used to contain a brief description of the purpose of the control.

Values

Any string can be used as the Tooltip text, up to a maximum of 80 characters.

7.2.12.1.11 PageSize

Description

This property represents the Page Size of the ScrollBar control. This is how many items will be scrolled when the user clicks within the background of the ScrollBar control.

Values

Any numerical value greater than 0 can be used.

NB: **If the page size is greater than the range set by SetRange, the page size will be adjusted to be the entire range of the ScrollBar.**

7.2.12.2 Methods

7.2.12.2.1 SetRange

SetRange (sMin, sMax)

This method is used to set the range of values for a ScrollBar control.

Parameters

sMin	The start of the range to set
sMax	The end of the range to set.

Return value

This method has no return value.

7.2.12.2.2 GetRange

GetRange (sMin, sMax)

This method is used to get the current range of values of a ScrollBar control.

Parameters

sMin	After this method is called, this parameter will hold the ScrollBar control's current minimum value.
sMax	After this method is called, this parameter will hold the ScrollBar control's current maximum value.

Return value

This method has no return value.

7.2.12.2.3 SetEvent_OnPosChanged

SetEvent_OnPosChanged (eventref)

This method is used to specify the function that will be called when the ScrollBar's position is changed..

See OnPosChanged event for details of the event function.

NB: **The form's event handler MUST have been initialised using InitEventHandler before any events will fire.**

Parameters

eventref	A reference to the event function to be called when the ScrollBar's position changes. This must be passed in the form **GetRef ("button3_OnClick")**.

Return value

This method has no return value.

Example

The following code snippet will create a form, initialize it so that events will be fired, create a ScrollBar, and set an event that is fired when the ScrollBar's position changes.

```
Set form1 = CreateObject ("ScriptDlg.Form")
form1.InitEventHandler(form1)

form1.AddScrollBar "HScroll1", 300, 40, 5, 5
form1.HScroll1.SetEvent_OnPosChanged
     GetRef("HScroll1_OnPosChanged")
form1.HScroll1.SetRange 0, 50
form1.HScroll1.CurPos = 25

form1.Display()

Sub HScroll1_OnPosChanged(nPos)
  MsgBox "Current ScrollBar position is " &
     form1.HScroll1.CurPos
End Sub
```

7.2.12.3 Events

7.2.12.3.1 OnPosChanged event

The OnPosChanged event will be fired when a ScrollBar's position changes.

To tell the ScrollBar control which event function to call when its position changes, use the SetEvent_OnPosChanged method.

The OnPosChanged event function may have any name, but must have the following format:

```
Sub scrollbar_OnPosChanged()
     ' Do your processing here
End Sub
```

Parameters

This event function has a single parameter which represents the type of position change:

0	Left / Up (the ScrollBar's position was moved by one place left (horizontal controls) or one place up (vertical controls)
1	Right / Down (the ScrollBar's position was moved by one place right (horizontal controls) or one place down (vertical controls)
2	Page Left / Page Up (the ScrollBar's position was moved by one page left (horizontal controls) or one page up (vertical controls)
3	Page Right / Page Down (the ScrollBar's position was moved by one page right (horizontal controls) or one page down (vertical controls)
4	The ScrollBar has been moved using the mouse, and has now finished moving. Use the CurPos property to get the current position.
5	The ScrollBar is being moved using the mouse. Use the CurPos property to get the current position.

d

Return value

This event function has no return value.

Part 8

Glossary

8 Glossary

A

ActiveX - ActiveX describes a group of technologies (incorporating OLE, automation etc) by which different Windows applications can comminucate with each other and use each others' capabilities.

Array - an array of variables is a series of variables that are all of the same type and size. Individual elements in the array are accessed using the index of the element (zero-based);

ASIO (Audio Stream Input/Output) - A digital audio device driver protocol for Windows computers, created by Steinberg, that bypasses the Windows audio processing, allowing audio to be passed unaltered and with low latency between different applications and devices.

Automation - Automation is the process by which one application can control another using its ActiveX interface.

B

C

D

E

Element - an element is a single item in an array.

Event-driven - a script is said to be event-driven if its main body has finished running, and the only code that subsequently runs is triggered by a certain event happening (for example, a Limit Line being breached).

Event handler - an event handler is a subroutine that is inserted into a script to handle a certain event. When that event is triggered, the event handler subroutine will be called to take the appropriate action.

Expression - a combination of keywords, operators, variables, and constants that yield a value. This value may be a string, number, or object. An expression can perform a calculation, manipulate characters, or test data.

F

G

H

Hungarian notation - This is a system used by some programmers, in which the type of a variable is specified by inserting one or more letters at the beginning of the variable name.
For example: `sReturnValue` means that the variable indicating the return value is of type "short integer", shown by the s at the start.

I

J

K

L

M

Method - an object's method is a function that has been made available via its OLE interface, allowing the function to be called by another application.

Multi-threaded - if an application is multi-threaded, it means that is has two or more threads all running concurrently, and this enables more than one task to be performed at a time.
In fact the tasks do not run at *exactly* the same time, but the Windows operating system manages switching between them to make it look as if they are happening at the same time.

N

O

OLE - OLE stands for Object Linking and Embedding. It is a term used for a number of ways in which Windows allows different applications to interact with each other.

P

Pixel - a pixel is one dot on the screen. If your screen resolution is set up to be 800 x 600, this means that there are 800 pixels across the screen, and 600 down.

Property - an object's property is a setting that has been made available via its OLE interface, allowing the property to be set and/or read by another application.

Q

R

Read-only - A property is said to be read-only when its value can be read, but not written.

S

Scope - the scope of a variable defines the context in which it can be seen by a script. A variable declared outside any functions or subroutines has 'global' scope, that is, it can be seen and used by anything in the script. A variable declared within a function or subroutine has 'local' scope, and can only be used from within that function or subroutine.

Script engine - A script engine is the entity that processes a script - that is, checks for syntax errors and runs the code in the script.

Scripting host - A scripting host is a program that contains a script engine, thus allowing it to run a script to control itself or other applications.

T

Type library - A Type library must be defined by any program that supports OLE Automation. The Type library is simply a definition of all the methods and properties that can be externally controlled.

Thread - a multi-threaded application has several threads running. Each thread performs a separate task or group of tasks, each one effectively running at the same time as the others.

Title bar - A window's Title Bar is the bar at the top of the window, containing the window name, and usually buttons to minimize, maximize and close the window.

U

V

Variant - Variables in VBScript are all stored internally as Variants. This means that the value can be of any type - e.g. integer, string or just an array of bytes. They assume a type when they are first assigned a value.

W

WDM (Windows Driver Model) - A digital audio device driver protocol for Windows computers.

X

Y

Z

Index

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- Z -