cadence™

Virtuoso Visualization and Analysis XL SKILL Reference

Product Version 6.1.6
November 2014

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

5
Results Browser Functions

Preface

This manual describes the SKILL functions that you can use with Virtuoso Visualization and Analysis XL. You can use these functions to modify Virtuoso Visualization and Analysis XL to better suit your needs.

This manual assumes you are familiar with the Cadence SKILL™ language.

The preface discusses the following:

Scope of this Manual

The SKILL functions described in this manual can be used in either IC6.1.6, ICADV12.1, or both of these releases. Functions that are supported only in a particular release are identified using the **(ICADV12.1 ONLY)** or **(IC6.1.6 ONLY)** text at the beginning of the function description. All other functions are supported in both releases.

Related Documents for Virtuoso Visualization and Analysis XL SKILL Functions

The SKILL programming language is often used with other Virtuoso products or requires knowledge of a special language. The following documents give you more information about these tools and languages.

- If you want to use the SKILL language functions, the Virtuoso SKILL++™ functions, and the SKILL++ object system (for object-oriented programming), you need to read the *Cadence SKILL Language User Guide*.
- If you want to see descriptions, syntax, and examples for the SKILL and SKILL++ functions, you need to read the *Cadence SKILL Language Reference*.
- If you want to see descriptions, syntax, and examples for the object system functions, you need to read the *Cadence SKILL++ Object System Reference*.
- *Virtuoso Design Environment SKILL Functions Reference* provides detailed information about the SKILL functions that interface to applications in the Virtuoso Design Environment.
- If you want to design and simulate analog and mixed-signal circuits, you need to read the *VirtuosoAnalog Design Environment L User Guide*.
- If you want to set up, simulate, and analyze circuit data without starting the Virtuoso analog design environment, you need to read the *OCEAN Reference*.
- If you want to use the Virtuoso®schematic editor to enter your design, you need to read the *Virtuoso Schematic Editor L User Guide*.
- If you want to use the Virtuoso®layout editor to enter your design, you need to read the *Virtuoso Layout Suite L User Guide*.

Additional Learning Resources

Cadence provides various Rapid Adoption Kits that you can use to learn how to employ Virtuoso applications in your design flows. These kits contain workshop databases, designs, and instructions to run the design flow.

Cadence offers the following training course on the Virtuoso Visualization and Analysis XL flow:

- Virtuoso Analog Design Environment
- Virtuoso Schematic Editor

- Analog Modeling with Verilog-A
- Behavioral Modeling with Verilog-AMS
- Real Modeling with Verilog-AMS
- Spectre Simulations Using Virtuoso ADE
- Virtuoso UltraSim Full-Chip Simulator
- Virtuoso Simulation for Advanced Node

For further information on the training courses available in your region, visit the Cadence Training portal. You can also write to training_enroll@cadence.com.

The links in this section open in a new browser. The course links initially display the requested training information for North America, but if required, you can navigate to the courses available in other regions.

Typographic and Syntax Conventions

This list describes the syntax conventions used for the Virtuoso Visualization and Analysis XL SKILL functions.

`literal`	Nonitalic words indicate keywords that you must enter literally. These keywords represent command (function, routine) or option names.
`argument` (*`z_argument`*)	Words in italics indicate user-defined arguments for which you must substitute a name or a value. (The characters before the underscore (_) in the word indicate the data types that this argument can take. Names are case sensitive. Do not type the underscore (*`z_`*) before your arguments.)
|	Vertical bars (OR-bars) separate possible choices for a single argument. They take precedence over any other character.
`[ ]`	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices. You can choose one argument from the list.
`{ }`	Braces are used with OR-bars and enclose a list of choices. You must choose one argument from the list.

`...`	Three dots (`...`) indicate that you can repeat the previous argument. If you use them with brackets, you can specify zero or more arguments. If they are used without brackets, you must specify at least one argument, but you can specify more.
`argument...`	specify at least one, but more are possible
`[argument]...`	you can specify zero or more
`,...`	A comma and three dots together indicate that if you specify more than one argument, you must separate those arguments by commas.
`=>`	A right arrow points to the possible values that a SKILL function can return. This character is represented by an equal sign and a greater than sign.
`/`	A slash separates the possible values that can be returned by a SKILL function.
`<`*`yourSimulator`*`>`	Angle brackets are used to indicate places where you need to insert the name of your simulator. Do not include the angle brackets when you insert the simulator name.

Important

The language requires any characters not included in the list above. You must enter required characters literally.

SKILL Syntax Examples

The following examples show typical syntax characters used in the SKILL language.

Example 1

```
list( g_arg1 [g_arg2] ...
     )
     => l_result
```

Example 1 illustrates the following syntax characters.

`list`	Plain type indicates words that you must enter literally.

g_arg1	Words in italics indicate arguments for which you must substitute a name or a value.
( )	Parentheses separate names of functions from their arguments.
_	An underscore separates an argument type (left) from an argument name (right).
[ ]	Brackets indicate that the enclosed argument is optional.
=>	A right arrow points to the return values of the function. Also used in code examples in SKILL manuals.
. . .	Three dots indicate that the preceding item can appear any number of times.

Example 2

```
needNCells(
     s_cellType | st_userType
     x_cellCount
     )
     => t/nil
```

Example 2 illustrates two additional syntax characters.

|	Vertical bars separate a choice of required options.
/	Slashes separate possible return values.

Identifiers Used to Denote Data Types

The Cadence SKILL language supports different data types to identify the type of value you can assign to an argument.

Data types are identified by a single letter followed by an underscore; for example, *t* is the data type in *t_viewNames* and denotes that the argument in question accepts a character string. Data types and the underscore are used as identifiers only; they should not be typed.

Prefix	Internal Name	Data Type
a	array	array

Prefix	Internal Name	Data Type
A	amsobject	AMS Object
b	ddUserType	DDPI object
B	ddCatUserType	DDPI Category Object
C	opfcontext	OPF context
d	dbobject	Cadence database object (CDBA)
e	envobj	environment
f	flonum	floating-point number
F	opffile	OPF file ID
g	general	any data type
G	gdmSpecIIUserType	gdm spec
h	hdbobject	hierarchical database configuration object
K	mapiobject	MAPI object
l	list	linked list
L	tc	Technology file time stamp
m	nmpIIUserType	nmpII user type
M	cdsEvalObject	—
n	number	integer or floating-point number
o	userType	user-defined type (other)
p	port	I/O port
q	gdmspecListIIUserType	gdm spec list
r	defstruct	defstruct
R	rodObj	relative object design (ROD) object
s	symbol	symbol
S	stringSymbol	symbol or character string
t	string	character string (text)
T	txobject	Transient Object
u	function	function object, either the name of a function (symbol) or a lambda function body (list)

Prefix	Internal Name	Data Type
U	funobj	function object
v	hdbpath	—
w	wtype	window type
x	integer	integer number
y	binary	binary function
&	pointer	pointer type

1

Introduction

The Virtuoso Visualization and Analysis XL SKILL functions in this manual are organized into sections according to their purpose. Within each chapter, the functions are in alphabetical order.

Each Waveform Window can contain a number of subwindows, each identified by an integer index. Each subwindow can contain a number of curves, each identified by an integer index.

Many of the functions in this chapter describe operations you can perform on a subwindow. If you do not specify a particular subwindow, the current subwindow is affected. It is important to understand that even if you call a Waveform Window and do not add any subwindows, your waveform display is still considered a subwindow. You can think of it as subwindow number one of one. Any of the functions that deal with subwindows will work on this type of waveform display as well as on waveform displays with several subwindows.

Searching for SKILL Functions using Finder Assistant

To view the description and syntax of the Virtuoso Visualization and Analysis XL SKILL functions, use the *Finder* assistant that can be accessed either directly from the CIW or through the SKILL IDE tool. For information about using the Finder assistant, refer to the Working with the Finder Assistant section in the *Cadence SKILL IDE User Guide*.

Plotting Functions

By default, waveforms are plotted in the current subwindow, go to axis Y1, and to different strips (when the display mode is `strip`).

Setting Waveform Window Defaults

You can set the default values for some of the Waveform Window options with the `awvSetOptionValue` function. When you set the default of an option in the CIW, the new value takes effect when you open a new Waveform Window or add a subwindow to an existing

Waveform Window. The plotting options take effect as soon as you send an image to the plotter.

If you want the defaults to apply to every new session, you need to change the options in your `.cdsinit` file.

Option name	Description	Type	Default	Valid values
cursorSuppressed	Removes the tracking cursor display	Boolean	`nil`	`t or nil`
cursorPrecision	Controls the number of digits displayed (at the top of the window) as cursor output	integer	`4`	Any integer greater than 2 and less than 16
cursorAction	Controls whether the cursor snaps to waveforms or actual data points	string	`"line"`	`"data point"` `"line"`
cursorPhase	Controls the phase display (Smith)	string	`"degree"`	`"degree"` `"radian"`
cursorValue	Controls the value display (Smith)	string	`"normalized impedance"`	`"normalized admittance"` `"reflection coefficient"` `"normalized impedance"`
dateStamp	Adds the date to the top right corner of the window	Boolean	`nil`	`t or nil`
displayAxes	Displays the axes	Boolean	`t`	`t or nil`

Option name	Description	Type	Default	Valid values
displayAxesBy125	Displays the axis labels by increments of 1, 2, or 5	Boolean	`nil`	`t or nil`
displayAxesLabel	Displays the axes labels	Boolean	`t`	`t or nil`
displayGrids	Displays grid lines	Boolean	`nil`	`t or nil`
displayMajorTicks	Displays major tick marks	Boolean	`t`	`t or nil`
displayMinorTicks	Displays minor tick marks	Boolean	`t`	`t or nil`
xLog	Displays the X axis logarith-mically	Boolean	`nil`	`t or nil`
mode	Controls the mode type	string	`"composite"`	`"strip" "smith" "composite"`
style	Controls how waveforms plot in the window	string	`"auto"`	`"bar" "scatterPlot""joined" "auto"`
numIdentifier	Controls the number of identifiers per waveform	integer	`6`	Any positive integer or nil to show all the identifiers
hcCopyNum	Specifies the number of copies to plot	integer	`1`	Any positive integer
hcDisplay	Specifies the display name	string	`"display"`	Defined in the technology file
hcHeader	Specifies header with plot	boolean	t	t or nil

Option name	Description	Type	Default	Valid values
hcMailLogNames	Emails plot submission output to user	boolean	t	t or nil
hcOrientation	Specifies the plot orientation	string	`"automatic"`	`"portrait"` `"landscape"` `"automatic"`
hcOutputFile	Specifies to plot to the file only	general (string or nil)	`nil`	Name of the output file
hcPaperSize	Specifies the plot paper size	string	Specified in `.cdsplotinit`	
hcPlotterName	Sets the plotter name	string	Specified in `.cdsplotinit`	
hcTmpDir	Specifies the temporary scratch space	string	`"/usr/tmp"`	Name of a temporary directory

2

Waveform Window Functions

This chapter describes the following waveform window functions in detail:

- awvAddSubwindow
- awvAnalog2Digital
- awvAppendExpression
- awvAppendList
- awvAppendWaveform
- awvClearPlotWindow
- awvClearSubwindowHistory
- awvClearWindowHistory
- awvCloseCalculator
- awvCloseWindowMenuCB
- awvCreateBus
- awvCreateBusFromWaveList
- awvCreatePlotWindow
- awvDeleteAllWaveforms
- awvDeleteSubwindow
- awvDeleteWaveform
- awvDigital2Analog
- awvDisableRedraw
- awvDisplayDate
- awvDisplayGrid

- awvDisplaySubwindowTitle
- awvDisplayTitle
- awvEraseWindowMenuCB
- awvEval
- awvExitWindowFunctionAdd
- awvExitWindowFunctionDel
- awvExitWindowFunctionGet
- awvGetCurrentSubwindow
- awvGetCurrentWindow
- awvGetDisplayMode
- awvGetDrawStatus
- awvGetInitializationTimeout
- awvGetOnSubwindowList
- awvGetPlotStyle
- awvGetSavePromptNeeded
- awvGetSmithModeType
- awvGetSubwindowList
- awvGetUnusedEntityList
- awvGetWaveNameList
- awvGetWindowList
- awvGetXAxisLabel
- awvGetXMarkerNames
- awvGetYAxisLabel
- awvGetYMarkerNames
- awvInitWindowFunctionAdd
- awvInitWindowFunctionDel
- awvInitWindowFunctionGet

- awvIsPlotWindow
- awvLoadMenuCB
- awvLoadWindow
- awvLogYAxis
- awvLogXAxis
- awvPlaceWaveformLabel
- awvPlaceWindowLabel
- awvPlaceXMarker
- awvPlaceYMarker
- awvPlotExpression
- awvPrintWaveform
- awvPlotList
- awvPlotWaveform
- awvPlotWaveformOption
- awvRedisplaySubwindow
- awvRedrawWindowMenuCB
- awvRedisplayWindow
- awvRemoveDate
- awvResumeViVA
- awvRemoveLabel
- awvRemoveSubwindowTitle
- awvRemoveTitle
- awvResetAllWindows
- awvResetWindow
- awvRfLoadPull
- awvSaveWindow
- awvSaveWindowImage

- awvSaveMenuCB
- awvSetCurrentSubwindow
- awvSetCurrentWindow
- awvSetCursorPrompts
- awvSetDisplayMode
- awvSetDisplayStatus
- awvSetInitializationTimeout
- awvSetOptionDefault
- awvSetOptionValue
- awvSetOrigin
- awvSetPlotStyle
- awvSetSavePromptNeeded
- awvSetSmithModeType
- awvSetSmithXLimit
- awvSetSmithYLimit
- awvSmithAxisMenuCB
- awvSetUpdateStatus
- awvSetWaveformDisplayStatus
- awvSetWaveNameList
- awvSetXAxisLabel
- awvSetXLimit
- awvSetXScale
- awvSetYAxisLabel
- awvSetYLimit
- awvSetYRange
- awvSimplePlotExpression
- awvUpdateAllWindows

- awvUpdateWindow
- awvZoomFit
- awvZoomGraphX
- awvZoomGraphY
- awvZoomGraphXY

When you work with the Waveform Window using SKILL, you must use* only *the functions described in this chapter. Never use the functions that appear in the CIW when you use the menus in the Waveform Window because these functions interact with menus and forms directly. Unpredictable results might occur if you use these CIW functions in a SKILL procedure and their associated forms and menus are not instantiated.

awvAddSubwindow

```
awvAddSubwindow( w_windowId )
     => x_subwindow | nil
```

Description

Adds a subwindow to the Waveform window.

Arguments

w_windowId	Waveform window ID.

Value Returned

x_subwindow	Returns the number for the new subwindow (found in the upper right corner).
`nil`	Returns `nil` if the specified Waveform window does not exist.

Example

```
awvAddSubwindow( window(3) )
=> 2
```

Adds a new subwindow and returns its number.

Related Function

To delete a subwindow, see the awvDeleteSubwindow function on page-48.

awvAnalog2Digital

```
awvAnalog2Digital(
     o_wave
     n_vhi
     n_vlo
     n_vc
     n_timex
     t_thresholdType
     )
     => o_digWave | n_digval/nil
```

Description

Returns the digital form of the analog input, which may be a scalar, waveform, list or family of waveforms or a string representation of expression(s).

Arguments

o_wave	Input waveform.
o_vhi	High threshold value (used only when *t_thresholdType* is hilo).
o_vlo	Low threshold value (used only when *t_thresholdType* is hilo).
o_vc	Central threshold value (used only when *t_thresholdType* is centre).
n_timex	The value that determines logic X.
t_thresholdType	Can take the values hilo or centre. If *t_thresholdType* is centre, it is a high state (1) unless its value is less than n_vc, in which case it is a low state (0). If *t_thresholdType* is hilo, any value less than *n_vlo* is a low state (0), any value greater than *n_vhi* is a high state (1) and the rest is treated as unknown based on the value of *n_timex*.

Value Returned

o_digWave	A waveform (or a list of waveforms) is returned if the analog input specified was *o_wave*.

o_digVal	A scalar value is returned if the analog input specified was *o_val*.
nil	Returns nil if the specified Waveform window does not exist.

awvAppendExpression

```
awvAppendExpression(
     w_windowId
     t_expr
     l_context
     [?index        l_waveIndexList]
     [?color        l_colorList]
     [?lineType     l_styleList]
     [?dataSymbol   l_symbolList]
     [?subwindow    x_subwindow]
     [?showSymbols  l_showList]
     [?lineStyle    l_styleList]
     [?lineThickness l_thicknessList]
     )
     => t | nil
```

Description

Evaluates the *t_expr* expression and adds the resulting waveforms to a subwindow.

The new waveforms are plotted at the same strip and Y axis as their corresponding curve from *l_waveIndexList*. Also, the new waveforms are assigned the next lowest unassigned numbers. So, if *l_waveIndexList* contains curves 1 and 2, the curves resulting from the expression evaluation are numbered 3, 4, and so on.

Note: If you do not specify *l_waveIndexList*, the new waveforms are plotted at the Y-axis and strip of existing waveforms with the lowest numbers. These new waveforms are assigned the lowest *unused* numbers.

Arguments

w_windowId	Waveform window ID.
t_expr	String containing an expression that is evaluated once when the command is issued and re-evaluated at the completion of each simulation in the auto-update mode. *t_expr* can be `nil`.
l_context	Data context for a particular simulation. If evaluating the expressions requires data generated during a simulation, you must specify this argument. Otherwise, specify `nil`.
l_waveIndexList	List of integer identifiers for existing waveform curves that were plotted with one of the Waveform Window plot functions

l_colorList

List specifying the colors for the waveforms. If you do not supply this argument, the default colors are used. The colors that are available are defined in your technology file.
Valid Values: "y1" through "y66"

l_styleList

List specifying the line styles for the waveforms. If you do not supply this argument, the default line style is used.
Valid Values: `line`, `bar`, `scalarPlot` and `polezero`. Default Value: `Line`.

l_symbolList

List of symbols to use if you want the system to plot data points only (that is, not curves). The system places a symbol on each data point and does not use a line to connect the points. To use a symbol, specify an integer or a single character corresponding to the symbol, as listed below:

To use the symbol	Enter integers	Enter character
+ (plus)	0, 10 or 20	+
. (dot)	1	.
x	2, 9, 19	x or X
square	3, 5, 15	Not supported
circle	4, 6 or 16	o or O
box	5	Not supported

x_subwindow

Number for the subwindow (found in the upper right corner).
Default Value: Current subwindow

l_showList

Flag or list of flags that specifies if the symbols for the traces should be plotted or not. By default, the value is set as `nil` and none of the symbols is plotted. Set the value as `t` to show symbols for all the traces. To set the show status for each symbol, specify a list of flags, one flag for each symbol. The number of flags should match the number of symbols in `l_symbolList`. Each flag specifies if the corresponding symbol in the `l_symbolList` list will be shown or not.
Default Value: `nil`

l_styleList	Specifies the line-style of the signal. Valid values: `Solid`, `Dotted`, `Dashed`, `Dotdashed`
l_thicknessList	Specifies the thickness of the signal. Valid values: `Fine`, `Medium`, `Bold`

Value Returned

`t`	Returns `t` when the *t_expr* expression is evaluated and the resulting waveforms are added to the Waveform Window.
`nil`	Returns `nil` if there is an error.

Related Functions

To plot the Y values in a list against the X values in a list and add the resulting waveforms to a subwindow, see the awvAppendList function on page-32.

To add the waveforms in a list to a subwindow, see the awvAppendWaveform function on page-35.

To evaluate an expression and assign the numbers specified in a list to the waveforms resulting from the evaluation, see the awvPlotExpression function on page-100.

To plot the Y values in a list against the X values in a list and display the resulting waveforms in a subwindow, see the awvPlotList function on page-106.To plot the waveforms in a list in a subwindow, see the awvPlotWaveform function on page-112.

To evaluate an expression and specify whether to add the resulting waveforms to the set of existing waveforms, or overwrite the existing waveforms, see the awvSimplePlotExpression function on page-166.

awvAppendList

```
awvAppendList(
     w_windowId
     l_YListList
     l_XList
     [?index        l_waveIndexList]
     [?color        l_colorList]
     [?lineType     l_styleList]
     [?dataSymbol   l_symbolList]
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Plots the Y values in *l_YListList* against the X values in *l_XList* and adds the resulting waveforms to a subwindow.

The new waveforms are plotted at the same strip and Y axis as their corresponding curve in *l_waveIndex*. Also, the new waveforms are assigned the next lowest unassigned numbers. So, if *l_waveIndexList* contains curves 1 and 2, the curves resulting from plotting the X and Y values are numbered 3, 4, and so on.

If you do not specify *l_waveIndexList*, the new waveforms are plotted at the Y axis and strip of existing waveforms with the lowest numbers. These new waveforms are assigned the lowest *unused* numbers.

Arguments

w_windowId	Waveform window ID.
l_YListList	List of lists that specify the different Y values. Each list must have the same number of elements.
l_XList	List that specifies the different X values. The number of elements in this list must equal the number of elements in each list in *l_YListList*
l_waveIndexList	List of integers identifying existing waveform curves that were plotted with one of the Waveform window plot functions

l_colorList

List specifying the colors for the waveforms. If you do not supply this argument, the last color used in each of the waveform entities is used. The colors that are available are defined in your technology file.
Valid Values: "y1" through "y66"

l_styleList

List specifying the line styles for the waveforms. If you do not supply this argument, the default line style is used.
Valid Values: `line`, `bar`, `scalarPlot` and `polezero`. Default Value: `Line`.

l_symbolList

List of symbols to use if you want the system to plot data points only (that is, not curves). The system places a symbol on each data point and does not use a line to connect the points. To use a symbol, specify an integer or a single character corresponding to the symbol, as listed below:

To use the symbol	Enter integers	Enter character
+ (plus)	0, 10 or 20	+
. (dot)	1	.
x	2, 9, 19	x or X
square	3, 5, 15	Not supported
circle	4, 6 or 16	o or O
box	5	Not supported

x_subwindow

Number for the subwindow (found in the upper right corner). Default Value: Current subwindow.

Value Returned

`t`

Returns `t` when the waveforms created by plotting the specified Y values against the specified X values are added to the Waveform window.

`nil`

Returns `nil` if there is an error.

Related Functions

To evaluate an expression and add the resulting waveforms to a subwindow, see the awvAppendExpression function on page-29.

To add the waveforms in a list to a subwindow, see the awvAppendWaveform function on page-35.

To evaluate an expression and assign the numbers specified in a list to the waveforms resulting from the evaluation, see the awvPlotExpression function on page-100.

To plot the Y values in a list against the X values in a list and display the resulting waveforms in a subwindow, see the awvPlotList function on page-106.

To plot the waveforms in a list in a subwindow, see the awvPlotWaveform function on page-112.

To evaluate an expression and specify whether to add the resulting waveforms to the set of existing waveforms, or overwrite the existing waveforms, see the awvSimplePlotExpression function on page-166.

awvAppendWaveform

```
awvAppendWaveform(
     w_windowId
     l_waveform
     [?expr          l_exprList]
     [?index         l_waveIndexList]
     [?color         l_colorList]
     [?lineType      l_styleList]
     [?dataSymbol    l_symbolList]
     [?subwindow     x_subwindow]
     [?stripNumber   l_stripNumberList]
     [?showSymbols   l_showList]
     [?lineStyle     l_styleList]
     [?lineThickness l_thicknessList]
     )
     => t | nil
```

Description

Adds the waveforms in the *l_waveform* list to a subwindow.

The new waveforms are plotted at the same strip and Y axis as their corresponding curves from *l_waveIndexList*. Also, the new waveforms are assigned the next lowest unassigned numbers. So, if *l_waveIndexList* contains curves 1 and 2, the new curves are numbered 3, 4, and so on.

If you do not specify *l_waveIndexList*, the new waveforms are plotted at the Y axis and strip of existing waveforms with the lowest numbers. These new waveforms are assigned the lowest *unused* numbers.

Arguments

w_windowId	Waveform window ID.
l_waveform	List of waveforms to add.
l_exprList	Specifies the expressions to display next to the waveform identifiers. If you do not specify *l_exprList*, no expressions are displayed beside the waveform identifiers.
l_waveIndexList	List of integer identifiers for existing waveform curves that were plotted with one of the Waveform window plot functions.

l_colorList List specifying the colors for the waveforms. If you do not supply this argument, the default colors are used. The colors that are available are defined in your technology file.
Valid Values: "y1" through "y66"

l_styleList List specifying the line styles for the waveforms. If you do not supply this argument, the default line style is used.
Valid Values: line, bar, scalarPlot and polezero. Default Value: Line.

l_symbolList List of symbols to use if you want the system to plot data points only (that is, not curves). The system places a symbol on each data point and does not use a line to connect the points. To use a symbol, specify an integer or a single character corresponding to the symbol, as listed below:

To use the symbol	Enter integers	Enter character
+ (plus)	0, 10 or 20	+
. (dot)	1	.
x	2, 9, 19	x or X
square	3, 5, 15	Not supported
circle	4, 6 or 16	o or O
box	5	Not supported

x_subwindow Number for the subwindow (found in the upper right corner).
Default Value: Current subwindow

l_stripNumberList List of numbers identifying the strips.

l_showList Flag or list of flags that specifies if the symbols for the traces should be plotted or not. By default, the value is set as nil and none of the symbols is plotted. Set the value as t to show symbols for all the traces. To set the show status for each symbol, specify a list of flags, one flag for each symbol. The number of flags should match the number of symbols in l_symbolList. Each flag specifies if the corresponding symbol in the l_symbolList list will be shown or not.
Default Value: nil

l_styleList	Specifies the line-style of the signal. Valid values: `Solid`, `Dotted`, `Dashed`, `Dotdashed`
l_thicknessList	Specifies the thickness of the signal. Valid values: `Fine`, `Medium`, `Bold`

Value Returned

`t`	Returns t when the *l_waveform* waveforms are added to the Waveform window.
`nil`	Returns `nil` if there is an error.

Example

```
awvAppendWaveform( window( 2 ) list( w3 w4 ) ?color
    list( "y2" "y4" )
)
```

Adds waveforms *w3* and *w4* to the strip and Y axis locations of waveforms 1 and 2, respectively. *w3* is displayed in the *y2* layer color, and *w4* is displayed in the *y4* layer color.

Related Functions

To evaluate an expression and add the resulting waveforms to a subwindow, see the awvAppendExpression function on page-29.

To plot the Y values in a list against the X values in a list, see the awvAppendList function on page-32.

To evaluate an expression and assign the numbers specified in a list to the waveforms resulting from the evaluation, see the awvPlotExpression function on page-100.

To plot the Y values in a list against the X values in a list and display the resulting waveforms in a subwindow, see the awvPlotList function on page-106.

To plot the waveforms in a list in a subwindow, see the awvPlotWaveform function on page-112.

To evaluate an expression and specify whether to add the resulting waveforms to the set of existing waveforms, or overwrite the existing waveforms, see the awvSimplePlotExpression function on page-166.

awvClearPlotWindow

```
awvClearPlotWindow(w_windowId) => t/nil
```

Description

Clears the graphics shown in the waveform window. The History for the window and subwindows are maintained.

Arguments

w_windowId	window ID

Value Returned

t	Returns t when the contents of the window are erased.
nil	Returns nil if the Waveform window does not exist.

Example

```
awvClearPlotWindow(awvGetCurrentWindow()) => t
```

awvClearSubwindowHistory

```
awvClearSubwindowHistory(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => t/nil
```

Description

Erases the contents of a particular subwindow. This function deletes the waveforms, title, date stamp, and labels stored in internal memory. The other subwindows are not affected.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the contents of the subwindow are erased.
`nil`	Returns `nil` if the Waveform window or the subwindow does not exist.

Example

```
awvClearSubwindowHistory( window(3) ?subwindow 1 )
=> t
```

Erases the contents of subwindow 1 and deletes the information from internal memory.

Related Function

To erase the contents of a Waveform window, see the awvClearWindowHistory function on page-40.

awvClearWindowHistory

```
awvClearWindowHistory(
     w_windowId
     [?force   g_force]
     )
     => t/nil
```

Description

Erases the contents of a Waveform window and deletes the waveforms, title, date stamp, and labels stored in internal memory. This function operates on subwindows whose update statuses are *on*. To force this function to operate on all subwindows, set *g_force* to t.

Arguments

w_windowId	Waveform window ID.
g_force	Boolean flag that specifies whether the function operates on all subwindows, or only on subwindows whose update statuses are turned *on*. Valid Values: t to perform function on *all* subwindows, or nil to perform function only on updatable subwindows. Default value: nil.

Value Returned

t	Returns t when the waveform information is deleted.
nil	Returns nil if the specified Waveform window does not exist.

Example

```
awvClearWindowHistory( window(3) ?force t )
=> t
```

Clears all the subwindows from Waveform window 3, including subwindows whose update statuses are turned *off*.

Related Function

To erase the contents of a subwindow, see the awvClearSubwindowHistory.

awvCloseCalculator

```
awvCloseCalculator(
     adesession
     adexlSession)

) => t
```

Description

Closes calculator window of the current session or the session specified (optional).

Arguments

adeSession	Optional; closes the calculator window invoked from ADE session.
adexlSession	Optional; closes the calculator window invoked from ADEXL session

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns nil.

Example

```
awvCloseCalculator()
```

awvCloseWindow

```
awvCloseWindow(
     w_windowID
     )

) => t
```

Description

Closes the specified Waveform window.

Arguments

w_windowId	Waveform window ID.

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns nil.

Example

```
awvCloseWindow(window(6))
```

awvCloseWindowMenuCB

```
awvCloseWindowMenuCB() => t
```

Description

Closes the current window.

The function is defined in `dfII/etc/context/awv.cxt`.

Arguments

None

Value Returned

t	Returns `t` if the command was successful. Otherwise, throws error (if there are no windows to be deleted).

Example

```
awvCloseWindowMenuCB() => t
```

awvCreateBus

```
awvCreateBus(w_bus l_wavelist r_radix)=> o_bus/nil
```

Description

Creates a bus with the given digital signals and radix type.

Arguments

w_bus	Name of the digital waveform representing a bus.
l_wavelist	List of digital waves or expressions in the bus
r_radix	Radix of the bus.

Value Returned

o_bus	Returns an output digital bus.
nil	Returns `nil` if there is an error.

Example

Following are the examples to create a digital binary bus with the name *bus:*

```
awvCreateBus("bus" list( awvAnalog2Digital( v("/data<0> " ?result "tran-tran") nil
nil 0.5 nil "centre")
awvAnalog2Digital( v("/datab<1> " ?result "tran-tran")  nil nil 0.5 nil "centre")
awvAnalog2Digital( v("/data<1> " ?result "tran-tran") nil nil 0.5 nil "centre")
awvAnalog2Digital( v("/datab<0> " ?result "tran-tran")  nil nil 0.5 nil "centre")
) "Binary")
```

awvCreateBusFromWaveList

```
awvCreateBusFromWaveList( l_waveList )
     => o_bus | nil
```

Description

Creates a digital bus from a list of digital waves provided as input.

Arguments

l_wavelist	List of digital waves or expressions (in string format), which yield digital waves.

Value Returned

o_bus	A digital bus whose bits are the input digital waves. The first wave in the list corresponds to MSB and the last one corresponds to LSB.
`nil`	Returns `nil` if there is an error.

Example

```
awvCreateBusFromWaveList( dig1 dig2 dig3 dig4 )
=> srrWave: 170938824
```

Creates a digital bus from the digital waves `dig1`, `dig2`, `dig3`, and `dig4`.

awvCreatePlotWindow

```
awvCreatePlotWindow(
     ?parentWindow  w_windowId
     )
     => w_windowId | nil
```

Description

Creates a Waveform window and returns the window ID.

Arguments

w_windowId	Waveform window ID for the parent window. If the parent window is a graphics editor window, the new Waveform window uses pens from the parent widow's technology file.

Value Returned

w_windowId	Returns the ID for the new Waveform window.
nil	Returns nil if there is an error.

Example

```
awvCreatePlotWindow( ?bBox list( 0:0 500:500 ) )
=> window:2
```

Creates a Waveform window that is 500 by 500 pixels with the lower left corner of the window in the lower left corner of the screen.

Related Function

To delete a window, use the hiCloseWindow function or your window manager's close command. For more information about hiCloseWindow, see the *Cadence User Interface SKILL Reference*.

awvDeleteAllWaveforms

```
awvDeleteAllWaveforms(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Deletes all the waveforms in the specified subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the waveforms are deleted from the subwindow.
`nil`	Return `nil` if the Waveform window or the subwindow do not exist.

Example

```
awvDeleteAllWaveforms( window(2) ?subwindow 4 )
=> t
```

Deletes all the waveforms in subwindow 4.

Related Function

To delete a specific waveform curve from a subwindow, see the awvDeleteWaveform function on page-49.

awvDeleteSubwindow

```
awvDeleteSubwindow(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Deletes a subwindow from a Waveform window.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the subwindow is deleted.
`nil`	Returns `nil` if the Waveform window or the subwindow do not exist.

Example

```
awvDeleteSubwindow( window(3) ?subwindow 4 )
=> t
```

Removes subwindow 4 from the Waveform window.

Related Function

To add a subwindow to a Waveform window, see the awvAddSubwindow function on page-26.

awvDeleteWaveform

```
awvDeleteWaveform(
     w_windowId
     x_index
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Deletes a waveform curve from a subwindow.

Arguments

w_windowId	Waveform window ID.
x_index	Integer identifying the waveform curve.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the waveform is deleted.
`nil`	Returns `nil` if there is an error.

Example

```
awvDeleteWaveform( window(2) 1 ?subwindow 3 )
=> t
```

Deletes waveform curve 1 from subwindow 3 of a Waveform window.

Related Function

To delete all the waveforms in a subwindow, see the awvDeleteAllWaveforms function on page-47.

awvDigital2Analog

```
awvDigital2Analog(
     o_waveform
     n_vhi
     n_vlo
     s_VX
     @key
     s_mode
     s_outWaveType
     s_vprevSTART
     )
     => o_waveform | nil
```

Description

Computes the analog output of the provided digital waveform `o_waveform`.

Arguments

o_waveform	Represents the digital wave or bus that is to be converted to analog. This may be a simple wave or bus, a list of waves or buses, a string representing the expression yielding a wave or bus.
o_vhi	The high analog value to which the digital 1 (for single bit waveform) or maximum possible bus value is converted.
o_vlo	The low analog value to which the digital 0 (for single bit waveform) or minimum possible bus value is converted.
s_VX	The value to which state X of the digital wave is converted. It can be a number or a simple expression of `vhi`, `vlo` and `vprev` (that is, the previous value) in the form of a string.
s_mode	A string to be provided if `t_waveform` is a bus or it is a list with at least one bus. Valid values: `wavelist`, `busvalue` `wavelist`: the input bus will be converted into a list of analog waves, each representing a single bit digital waveform in the bus. `busvalue`: the output will be a single analog wave representing the value of the bus.

s_outWaveType — A string to be provided if t_waveform is a bus or it is a list with at least one bus.
Valid values: pwl, zeroT
pwl: the points in the outputted analog waveform will be joined by straight line segments.
zeroT: the output analog waveform will have voltage transitions in zero time.

s_vprevSTART — The initial value of vprev to be used. This is required if the provided digital waveform starts at state X and vprev is used in the expression supplied in s_VX. This value overrides the value specified by the .cdsenv variable vprevSTART for the tool calculator.d2a.

Value Returned

o_waveform — A waveform is returned if the input was a single-bit digital waveform or if the input was a bus and the specified s_mode was busvalue.

A list of waveforms is returned if the input was a list of single-bit digital waveforms or if the input was a bus and the specified s_mode was wavelist.

nil — Returns nil if the specified waveform window does not exist.

Example

```
awvDigital2Analog( bus1 5.0 0 "(vhi + vlo)/2.0" ?mode "busvalue" ?outWaveType
"zeroT")
```

Returns the analog, zero transition wave representing the specified digital bus bus1.

awvDisableRedraw

```
awvDisableRedraw( w_windowId g_disable )
     => t | nil
```

Description

Disables or enables redraw of the Waveform window based on the value of the *g_disable* flag. You might use this function to freeze the Waveform window display, send several plots to the window, then unfreeze the window to display all the plots at once.

Arguments

w_windowId	Waveform window ID.
g_disable	Flag that specifies whether the redraw feature of the window is disabled. Valid Values: `t` (redraw is disabled) or `nil` (redraw is enabled)

Value Returned

`t`	Returns `t` when redraw is successfully disabled or enabled.
`nil`	Returns `nil` if there is an error, such as the specified Waveform window does not exist.

awvDisplayDate

```
awvDisplayDate( w_windowId )
     => t | nil
```

Description

Displays the current date and time in the Waveform windowwindow.

Arguments

w_windowId	Waveform window ID.

Value Returned

`t`	Returns `t` when the date and time is displayed.
`nil`	Returns `nil` if the specified Waveform window does not exist.

Related Functions

To remove the date and time from a Waveform window, see the awvRemoveDate function on page-121.

To display a title in a subwindow, see the awvDisplaySubwindowTitle function on page-55.

To display a title on a Waveform window, see the awvDisplayTitle function on page-56.

awvDisplayGrid

```
awvDisplayGrid(
     w_windowId
     g_on
     [?subwindow x_subwindow]
     )
     => t
```

Description

Sets the display status for a grid on the indicated waveform subwindow.

Arguments

w_window	window ID
g_on	flag for turning the grid on (`t`) or off (`nil`)
x_subwindow	subwindow number, defaults to 1

Example

```
awvDisplayGrid(awvGetCurrentWindow() t) => t
```

awvDisplaySubwindowTitle

```
awvDisplaySubwindowTitle(
     w_windowId
     t_title
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Displays a title in a subwindow.

Arguments

w_windowId	Waveform window ID.
t_title	Title for the subwindow.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the title is displayed.
`nil`	Returns `nil` if there is an error.

Example

```
awvDisplaySubwindowTitle( window(2) "Transient Response" ?subwindow 1 )
```

Displays *Transient Response* as the title for subwindow 1.

Related Functions

To remove the title from a subwindow, see the awvRemoveSubwindowTitle function on page-124.

To display a title on a Waveform window, see the awvDisplayTitle function on page-56.

To remove the title from a Waveform window, see the awvRemoveTitle function on page-125.

awvDisplayTitle

```
awvDisplayTitle(
     w_windowId
     t_title
     )
     => t | nil
```

Description

Displays a title on a Waveform window.

Arguments

w_windowId	Waveform window ID.
t_title	Title for the Waveform window.

Value Returned

`t`	Returns `t` when the title is displayed.
`nil`	Returns `nil` if the specified Waveform window does not exist.

Example

```
awvDisplayTitle( window(2) "Transient Response" )
```

Displays *Transient Response* as the title for the Waveform window 2.

Related Functions

To remove the title from a Waveform window, see the awvRemoveTitle function on page-125.

To display a title in a subwindow, see the awvDisplaySubwindowTitle function on page-55.

awvEraseWindowMenuCB

```
awvEraseWindowMenuCB() => t | nil
```

Description

Deletes all the objects (e.g. waveforms, markers) from the waveform subwindow.

The function is defined in `dfII/etc/context/awv.cxt`.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awvEraseWindowMenuCB() => t
```

awvEval

```
awvEval( expr l_expr)
=> t | nil
```

Description

Returns the expression

Arguments

u_func	Function to add to the list. Callback parameter list: (*w_windowId*)

Value Returned

`t`	Returns `t` if the function is added to the list.
`nil`	Returns `nil` otherwise.

Example

```
w = VT("/out")

awvEval("Expr_1" w)

This function returns w, which is the second argument.
```

awvExitWindowFunctionAdd

```
awvExitWindowFunctionAdd( u_func )
     => t | nil
```

Description

Adds a function to the list of functions that are called when you close a Waveform window or exit the Cadence software while a Waveform Window is open.

Arguments

u_func	Function to add to the list. Callback parameter list: (*w_windowId*)

Value Returned

`t`	Returns `t` if the function is added to the list.
`nil`	Returns `nil` otherwise.

Example

```
procedure( myExitFunc( windowId ) awvSaveWindow( windowId
    sprintf( nil "~/wave%d" windowId->windowNum ) )
)
awvExitWindowFunctionAdd( 'myExitFunc )
```

Adds a procedure called `myExitFunc` to the list of functions that are called when a Waveform window is closed. The `myExitFunc` function automatically saves the Waveform window.

Related Functions

To delete a function from the list of *exit* functions, see the awvExitWindowFunctionDel function on page-60.

To get the list of *exit* functions, see the awvExitWindowFunctionGet function on page-61.

awvExitWindowFunctionDel

```
awvExitWindowFunctionDel( u_func )
     => t | nil
```

Description

Deletes a function from the list of functions that are called when you close a Waveform window or exit the Cadence software while a Waveform window is open.

Arguments

u_func	Function to delete from the list. Callback parameter list: (*w_windowId*)

Value Returned

`t`	Returns `t` if the function is deleted from the list.
`nil`	Returns `nil` otherwise.

Example

```
awvExitWindowFunctionDel( 'myExitFunc )
=> t
```

Removes `myExitFunc` from the list of exit functions.

Related Functions

To add an exit function, see the awvExitWindowFunctionAdd function on page-59.

To get the list of *exit* functions, see the awvExitWindowFunctionGet function on page-61.

awvExitWindowFunctionGet

```
awvExitWindowFunctionGet()
     => l_initFunctionList | nil
```

Description

Gets the list of functions that are called when you close a Waveform window or exit the Cadence software while a Waveform window is open.

Arguments

None.

Value Returned

l_initFunctionList	Returns the list of functions.
nil	Returns nil if the list is empty.

Example

```
awvExitWindowFunctionGet
=> (myExitFunc)
```

Returns a list containing the names of the exit functions for Waveform Windows. In this case, there is only one exit function.

Related Functions

To add an exit function, see the awvExitWindowFunctionAdd function on page-59.

To delete a function from the list of exit functions, see the awvExitWindowFunctionDel function on page-60.

awvGetCurrentSubwindow

```
awvGetCurrentSubwindow( w_windowId )
     => x_subwindow/nil
```

Description

Returns the current subwindow.

Arguments

w_windowId	Waveform window ID.

Value Returned

x_subwindow	Returns the identification number found in the upper right corner of the current subwindow.
nil	Returns nil if there is an error.

Example

```
awvGetCurrentSubwindow( window(2) )
=> 3
```

Returns 3 as the current subwindow for Waveform window 2.

Related Function

To specify a subwindow as the current subwindow, see the awvSetCurrentSubwindow function on page-134.

awvGetCurrentWindow

```
awvGetCurrentWindow()
     => w_windowId | nil
```

Description

Returns the window ID for the current Waveform window.

Arguments

None.

Value Returned

w_windowId	Returns the ID for the current Waveform window.
nil	Returns nil if there is no current Waveform window.

Example

```
awvGetCurrentWindow
=> window:4
```

Returns *window:4* as the current window.

Related Function

To specify a Waveform window as the current window, see the awvSetCurrentWindow function on page-135.

awvGetDisplayMode

```
awvGetDisplayMode(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => t_mode | nil
```

Description

Returns the display mode of a subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

t_mode	Returns the display mode of the subwindow (`strip`, `smith`, or `composite`).
`nil`	Returns `nil` if there is an error.

Related Functions

To set the display mode of a subwindow, see the awvSetDisplayMode function on page-138.

To return the Smith display type of a subwindow, see the awvGetSmithModeType function on page-71.

To set the Smith display mode for a subwindow, see the awvSetSmithModeType function on page-146.

awvGetDrawStatus

```
awvGetDrawStatus(w_windowId) => t | nil
```

Description

Returns the draw status of the waveform display window.

Arguments

w_windowId	window Id

Example

```
awvGetDrawStatus(awvGetCurrentWindow()) => t
```

awvGetInitializationTimeout

```
awvGetInitializationTimeout()
     => x_timeOut
```

Description

Retrieves the time-out period (in seconds) set for ADE to establish connection with Virtuoso Visualization and Analysis XL.

Arguments

None.

Value Returned

x_timeOut	Time-out period (in seconds).

Example

```
awvGetInitializationTimeout()
    => 240
```

This means that the time-out period for ADE to establish connection with Virtuoso Visualization and Analysis XL was set to 240 `seconds`.

awvGetOnSubwindowList

```
awvGetOnSubwindowList(
     w_windowId
     [?all     g_all]
     )
     => l_onSubwindows | nil
```

Description

Returns the list of subwindows that are being used in the specified Waveform window. This list includes only subwindows whose display and update statuses are turned *on*. To get a list of all subwindows whose displays are *on*, regardless of update status, set *g_all* to `t`.

Arguments

w_windowId	Waveform window ID.
g_all	Boolean flag that specifies whether the list shall contain every subwindow whose display is *on*, regardless of update status. Valid Values: `t` to include all subwindows whose displays are *on*, regardless of update status, or `nil` to include only subwindows whose update statuses and display are *on*.

Value Returned

l_onSubwindows	Returns the list of subwindows whose displays are *on*.
`nil`	Returns `nil` if the specified Waveform window does not exist.

Example

```
awvGetOnSubwindowList( window(5) ?all t )
=> (1 2 3)
```

Returns a list of three subwindows that are being used in Waveform window 5, including subwindows whose update statuses are *off*.

Related Function

To get a list of all the subwindows whose update statuses are turned *on* (regardless of whether their displays are *on* or *off*), see the awvGetSubwindowList function on page-72.

awvGetPlotStyle

```
awvGetPlotStyle(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => s_style
```

Description

Gets the plotting style for the waveforms in a subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

s_style	Returns the plotting style for the subwindow.

Example

```
awvGetPlotStyle( window(4) ?subwindow 2 )
=> scatterPlot
```

Returns `scatterPlot` as the plotting style for subwindow 2.

Related Function

To set the plotting style for all the waveforms in a subwindow, see the awvSetPlotStyle function on page-144.

awvGetSavePromptNeeded

```
awvGetSavePromptNeeded()
     => t | nil
```

Description

Returns the value of an option that toggles a prompt for saving Waveform Windows. If this function returns `t`, then the software is set up to prompt you to save unsaved Waveform Windows when you quit them or quit the Cadence software. If this function returns `nil`, you are *not* prompted to save Waveform Windows.

Arguments

None.

Value Returned

`t`	Returns `t` if the software is set to prompt you to save Waveform Windows.
`nil`	Returns `nil` otherwise.

Example

```
awvGetSavePromptNeeded
=> t
```

Returns `t`, which means that the software is set up to prompt you to save unsaved Waveform Windows.

Related Function

To set the option for the prompt for saving Waveform Windows, see the awvSetSavePromptNeeded function on page-145.

awvGetSmithModeType

```
awvGetSmithModeType(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => t_type | nil
```

Description

Returns the Smith display type of a subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

t_type	Returns the type of Smith display (`impedance`, `admittance`, or `polar`).
`nil`	Returns `nil` if there is an error.

Example

```
awvGetSmithModeType( window(2) ?subwindow 3)
=> "polar"
```

Returns polar as the Smith display type for subwindow 3.

Related Functions

To set the Smith display type of a subwindow, see the awvSetSmithModeType function on page-146.

To return the display mode of a subwindow, see the awvGetDisplayMode function on page-64.

To set the display mode of a subwindow, see the awvSetDisplayMode function on page-138.

awvGetSubwindowList

```
awvGetSubwindowList(
     w_windowId
     [?all     g_all]
     )
     => l_subwindows | nil
```

Description

Returns a list of all the subwindows whose update statuses are turned *on*, regardless of whether their displays are *on* or *off*. To get a list of all subwindows, including subwindows whose update statuses are *off*, set `g_all` to `t`.

Arguments

w_windowId	Waveform window ID.
g_all	Boolean flag that specifies whether the function returns all subwindows, or only subwindows whose update statuses are turned *on*. Valid Values: `t` to return all subwindows, `nil` to return only updatable subwindows.

Value Returned

l_subwindows	Returns the list of all the subwindows for a Waveform window.
`nil`	Returns `nil` if the specified Waveform window does not exist.

Example

```
awvGetSubwindowList( window(5) ?all t )
=> (1 2 3 4)
```

Returns a list of all the subwindows for Waveform window 5, including subwindows whose update statuses are turned *off*.

Related Function

To return a list of subwindows that are being used in a Waveform window (including *only* subwindows whose display and update statuses are turned *on*), see the awvGetOnSubwindowList function on page-67.

awvGetUnusedEntityList

```
awvGetUnusedEntityList(
     w_windowId
     [?subwindow    x_subwindow]
     [?total   x_total]
     )
     => l_waveformEntityIndices | nil
```

Description

Returns a list of integers that have not already been used to identify curves in a subwindow. You can specify the total number of integers to include in the return value with *x_total*.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow
x_total	Specifies the number of unused curves to include in the return value, which is a list. Default Value: 20

Value Returned

l_waveformEntityIndices	Returns a list of the next lowest integers that have not been used to identify existing curves.
nil	Returns nil if there is an error.

Example

```
awvGetUnusedEntityList( window(2) ?subwindow 3 ?total 10 )
=>
(6 7 8 9 10
     11 12 13 14 15
)
```

Returns numbers 6 through 15 as the next 10 unused numbers. This means that curves 1 through 5 already exist in the window.

awvGetWaveNameList

```
awvGetWaveNameList(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => l_infoList
```

Description

Returns a list that contains two elements. The first element is a list of numbers for the curves and the second is a list of the corresponding names.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

l_infoList	Returns a list with the numbers and another list with the names for the waveform curves.

Example

```
awvGetWaveNameList( window(3) ?subwindow 1)
=>
((2 3)
("/net30" "/net50")
)
```

Returns a list with the numbers and a list with the names for waveform curves 2 and 3.

awvGetWindowList

```
awvGetWindowList()
     => l_windows
```

Description

Returns a list of all the Waveform Windows associated with the current process.

Arguments

None.

Value Returned

l_windows	Returns a list of all the Waveform Windows.

Example

```
awvGetWindowList
=> ( window:3 window:4 )
```

Returns the identifiers for two Waveform Windows.

awvGetXAxisLabel

```
awvGetXAxisLabel(
     w_windowId
     [?subwindow    x_subwindow]
     [?computed     g_computed]
     )
     => t_label | nil
```

Description

Returns the user-specified X axis label if you set `computed` to `nil`. Returns the system-computed X axis label otherwise.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow
g_computed	Boolean flag that specifies whether to return the user-specified X axis label or the system-computed X axis label. Valid Values: `t`, which specifies that the system-computed X axis label is returned, or `nil`, which specifies that the user-defined X axis label is returned.

Value Returned

t_label	Returns the X axis label for the specified subwindow.
`nil`	Returns `nil` if there is an error.

Example

```
awvGetXAxisLabel( window(7) ?subwindow 1 ?computed nil)
=> "seconds"
```

Returns *seconds* as the user-defined X axis label for subwindow 1.

Related Functions

If you want to specify an X axis label to replace the automatically computed label, see the awvSetXAxisLabel function on page-157.

To return the Y axis label, see the awvGetYAxisLabel function on page-80.

If you want to specify a Y axis label to replace the automatically computed label, see the awvSetYAxisLabel function on page-160.

awvGetXMarkerNames

```
awvGetXMarkerNames(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => l_markerNames | nil
```

Description

Returns the names of all the X markers in a subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

l_markerNames	Returns a list of all the X markers in the subwindow.
`nil`	Returns `nil` if there is an error.

Example

```
awvGetXMarkerNames( window(2) ?subwindow 1 )
=> ( "M1" "M2" "M3" )
```

Returns the names of the X markers in subwindow 1.

Related Function

To return the names of all the Y markers in a subwindow, see the awvGetYMarkerNames function on page-82.

awvGetYAxisLabel

```
awvGetYAxisLabel(
     w_windowId
     x_yNumber
     [?subwindow    x_subwindow]
     [?computed     g_computed]
     [?stripNumber  x_stripNumber]
     )
     => t_label | nil
```

Description

Returns the user-specified Y axis label if you set `computed` to `nil`. Returns the system-computed Y axis label otherwise.

Arguments

w_windowId	Waveform window ID.
x_yNumber	Specifies the Y axis whose label you want to get. Valid Values: 1 through 4
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow
g_computed	Boolean flag that specifies whether to return the user-specified Y axis label or the system-computed Y axis label. Valid Values: `t`, which specifies that the system-computed Y axis label is returned, or `nil`, which specifies that the user-defined Y axis label is returned.
x_stripNumber	Specifies the strip containing the Y axis. Valid Values: 1 through 20

Value Returned

t_label	Returns the Y axis label for the specified subwindow.
`nil`	Returns `nil` if there is an error.

Example

```
awvGetYAxisLabel( window(4) 2 ?subwindow 3 ?computed nil)
=> "Voltage"
```

Returns `Voltage` as the user-defined Y axis label for Y axis 2 of subwindow 3.

Related Functions

If you want to specify a Y axis label to replace the automatically computed label, see the awvSetYAxisLabel function on page-160.

To return the X axis label, see the awvGetXAxisLabel function on page-77.

If you want to specify an X axis label to replace the automatically computed label, see the awvSetXAxisLabel function on page-157.

awvGetYMarkerNames

```
awvGetYMarkerNames(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => l_markerNames | nil
```

Description

Returns the names of all the Y markers in a subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

*l_markerName*s	Returns a list of all the Y markers in the subwindow.
nil	Returns nil if there is an error.

Example

```
awvGetYMarkerNames( window(2) ?subwindow 2 )
=> ( "M1" "M2" "M3" )
```

Returns the names of the Y markers in subwindow 2.

Related Function

To return the names of all the X markers in a subwindow, see the awvGetXMarkerNames function on page-79.

awvInitWindowFunctionAdd

```
awvInitWindowFunctionAdd( u_func )
     => t | nil
```

Description

Adds a function to the list of functions that are called when a new Waveform window is opened. The list of functions is empty by default. For example, you can use this function to add menus to every new Waveform window that is opened.

Arguments

u_func	Function to add to the list. Callback parameter list: (*w_windowId*)

Value Returned

`t`	Returns `t` if the function is added to the list.
`nil`	Returns `nil` otherwise.

Example

```
procedure( myCreateMenu( windowId )
    let( ( item1 item2 )
        item1 = hiCreateMenuItem(
            ?name 'item1
            ?itemText "Item1"
            ?callback "myItem1CB()"
        )

        item2 = hiCreateMenuItem(
            ?name 'item2
            ?itemText "Item2"
            ?callback "myItem2CB()"
        )

        hiCreatePulldownMenu(
            'myMenu
            "Test"
            list( item1 item2 )
        )

        hiInsertBannerMenu( window myMenu hiGetNumMenus(window))
    )
)
awvInitWindowFunctionAdd( 'myCreateMenu )
```

Adds a procedure called `myCreateMenu`, which adds a banner menu called `Test` with two menu choices called *Item1* and *Item2*.

Related Functions

To delete a function from the list of initialization functions, see the awvInitWindowFunctionDel function on page-85.

To get the list of initialization functions, see the awvInitWindowFunctionGet function on page-86.

awvInitWindowFunctionDel

```
awvInitWindowFunctionDel( u_func )
     => t | nil
```

Description

Deletes a function from the list of functions that are called when a new Waveform window is opened.

Arguments

u_func	Function to delete from the list.

Value Returned

`t`	Returns `t` if the function is deleted from the list.
`nil`	Returns `nil` otherwise.

Example

```
awvInitWindowFunctionDel( 'myCreateMenu )
=> t
```

Removes the `myCreateMenu` function from the list of initialization functions.

Related Functions

To add a function to the list of initialization functions, see the awvInitWindowFunctionAdd function on page-83.

To get the list of initialization functions, see the awvInitWindowFunctionGet function on page-86.

awvInitWindowFunctionGet

```
awvInitWindowFunctionGet()
     => l_initFunctionList
```

Description

Returns the current list of functions that are called when a new Waveform window is opened. This list is empty by default.

Arguments

None.

Value Returned

l_initFunctionList
Returns the list of functions that are called when a new Waveform window is opened.

Example

```
awvInitWindowFunctionGet
=> (myCreateMenu)
```

Returns a list containing the names of the initialization functions for Waveform Windows. In this case, there is only one initialization function.

Related Functions

To add a function to the list of initialization functions, see the awvInitWindowFunctionAdd function on page-83.

To delete a function from the list of initialization functions, see the awvInitWindowFunctionDel function on page-85.

awvIsPlotWindow

```
awvIsPlotWindow( w_windowId )
     => t | nil
```

Description

Returns `t` if the specified window is a Waveform window.

Arguments

w_windowId	Window ID.

Value Returned

`t`	Returns `t` if the specified window is a Waveform window.
`nil`	Returns `nil` otherwise.

Example

```
awvIsPlotWindow( window(8))
=> t
```

Returns `t`, indicating that *window(8)* is a Waveform window.

awvLoadMenuCB

```
awvLoadMenuCB() => t | nil
```

Description

Displays the *Load Menu (Windows -> Load ...)*.

The function is defined in `dfII/etc/context/awv.cxt`.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awvLoadMenuCB() => t
```

awvLoadWindow

```
awvLoadWindow(
     w_windowId
     t_fileName
     )
     => t | nil
```

Description

Initializes the state of a Waveform window from information saved in a file.

Arguments

w_windowId	Waveform window ID for the window to be affected.
t_fileName	Name of the file containing the state of the Waveform window.

Value Returned

`t`	Returns `t` when the Waveform window is initialized.
`nil`	Returns `nil` if there is an error.

Example

```
awvLoadWindow( window(3) "my_file")
=> t
```

Initializes Waveform window 3 using the information saved in *my_file* (in the current directory).

Related Function

To save the state of a Waveform window, see the awvSaveWindowImage function on page-132.

awvLogYAxis

```
awvLogYAxis(
     w_windowId
     x_yNumber
     g_state
     [?stripNumber  x_stripNumber]
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the Y axis for a strip in a subwindow to display logarithmically if the *g_state* flag is set to `t`. If *g_state* is not set to `t`, the display is set to linear. If you do not specify a strip, the limits are applied when the Waveform window is in the *composite* mode.

Arguments

w_windowId	Waveform window ID.
x_yNumber	Specifies the Y axis whose display is to be set. Valid Values: 1 through 4
g_state	Flag that specifies whether the display is logarithmic or linear. Valid Values: `t` (specifies that the display is logarithmic), or `nil` (specifies that the display is linear)
x_stripNumber	Specifies the strip in which the Y axis display is to be set. Valid Values: 1 through 20
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the display of the Y axis is set.
`nil`	Returns `nil` if there is an error.

Example

```
awvLogYAxis( window( 2 ) 1 t )
```

Sets the Y axis (Y1) to display logarithmically. This takes effect only in the *composite* mode.

```
awvLogYAxis( window( 2 ) 1 nil ?stripNumber 2 )
```

Sets the Y axis (Y1) in strip 2 to display linearly. This takes effect only in the *strip* mode.

Related Function

To set the display mode of the X axis in a subwindow, see the awvSetXScale function on page-159.

awvLogXAxis

```
awvLogXAxis(
     w_windowId
     g_state
     [?subwindow x_subwindowID]
     )
     => t | nil
```

Description

Sets the X axis for a strip in a subwindow to display logarithmically if the *g_state* flag is set to t. If *g_state* is not set to t, the display is set to linear.

Arguments

w_windowId	Waveform window identifier
g_state	Flag that specifies whether the display is logarithmic or linear. Specifies t for log, otherwise sets it to the linear axis. The state set will not take effect if the axis type is *smith*.
[?subwindow x_subwindowId]	Subwindow number

Values Returned

t	Returns t if the command was successful
nil	Returns nil if the command was successful

Example

```
awvLogXAxis( window(2) t ?subwindow 2) => t
```

where window(2) corresponds to an AWD window

awvPlaceWaveformLabel

```
awvPlaceWaveformLabel(
     w_windowId
     x_waveIndex
     l_location
     t_label
     t_expr
     [?textOffset   g_textOffset]
     [?color        t_color]
     [?justify      t_justify]
     [?fontStyle    t_fontStyle]
     [?height       t_height]
     [?orient       t_orient]
     [?subwindow    x_subwindow]
     )
     => s_labelId | nil
```

Description

Attaches a label to the specified waveform curve in a subwindow.

Arguments

w_windowId	Waveform window ID.
x_waveIndex	Integer identifying the waveform curve.
l_location	List of two waveform coordinates that describe the location for the label.
t_label	Label for the waveform.
t_expr	String containing an expression that is evaluated once when the command is issued, and re-evaluated at the completion of each simulation in the auto-update mode. *t_expr* can be `nil`.
g_textOffset	Boolean that specifies whether to place a marker or label. If set to `t`, a marker is placed. If set to `nil`, a label is placed. Default value: `t`
t_color	Color for the waveform label. The colors that are available are defined in your technology file.

	Valid Values: `y1` through `y66`
t_justify	Justification for the label text. Valid Values: `lowerLeft, centerLeft, upperLeft, lowerCenter, centerCenter, upperCenter, lowerRight, centerRight, upperRight`
t_fontStyle	Font style for the label text. Valid Values: `stick, fixed, euroStyle, gothic, math, roman, script, swedish, milSpec`
t_height	Height for the label. Valid Values: `small, medium, large`
t_orient	Orientation for the label. Valid Values: `R0`, which specifies horizontal display, and `R90`, which specifies vertical display.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

s_labelId	Returns an identification number for the waveform label.
`nil`	Returns `nil` if there is an error.

Example

```
awvPlaceWaveformLabel( window(2) 1 list(0 3.5) "Input" nil )
```

Attaches the label *Input* to waveform curve 1 at the location (0 3.5).

Additional Information

Note the following points:

- The valid label location ranges between absolute co-ordinates (0, 0) on X-axis and (1,1) on Y-axis (upper and lower bound inclusive).
- The valid marker location ranges between data co-ordinates defined by X-axis and Y-axis limits (upper and lower bound inclusive).

Case 1:

```
awvPlaceWaveformLabel(awvGetCurrentWindow() 1 list( -0.5 0.5 ) "Label1" nil
?textOffset nil)
```

The following error message appears when the specified label location (`-0.5 0.5`) is outside of the defined boundary limits of label.

```
The location specified for placing a label on the graph is invalid. Specify a valid
label location that ranges between absolute coordinates (0,0) on X-axis and (1,1)
on Y-axis (upper and lower bounds inclusive).
```

Case 2:

```
awvPlaceWaveformLabel(awvGetCurrentWindow() 1 list( 80MHz -0.5 ) "MARKER " nil)
```

The following error message appears when the specified marker location (`80MHz -0.5`) is outside of the X- and Y-axis limits of the graph to be plotted.

```
The location specified for placing a marker on the graph is invalid. Specify a valid
marker location that ranges between data coordinates '(0,-1)' on X-axis and
'(10000,1)' on Y-axis (upper and lower bounds inclusive).
```

Related Functions

To display a label in a subwindow, see the awvPlaceWindowLabel function on page-96.

To remove the label, or all the labels identified in a list, from a subwindow, see the awvRemoveLabel function on page-123.

awvPlaceWindowLabel

```
awvPlaceWindowLabel(
     w_windowId
     l_location
     t_label
     t_expr
     [?color        t_color]
     [?justify      t_justify]
     [?fontStyle    t_fontStyle]
     [?height       t_height]
     [?orient       t_orient]
     [?subwindow    x_subwindow]
     )
     => s_labelId | nil
```

Description

Displays a label in a subwindow.

Arguments

w_windowId	Waveform window ID.
l_location	List of two Waveform window coordinates.
t_label	Label for the subwindow.
t_expr	String containing an expression that is evaluated once when the command is issued and re-evaluated at the completion of each simulation in the auto-update mode. *t_expr* can be `nil`.
t_color	Color for the waveform label. The colors that are available are defined in your technology file. Valid Values: `y1` through `y66`
t_justify	Justification for the label text. Valid Values: `lowerLeft, centerLeft, upperLeft, lowerCenter, centerCenter, upperCenter, lowerRight, centerRight, upperRight`

t_fontStyle	Font style for the label text. Valid Values: `stick, fixed, euroStyle, gothic, math, roman, script, swedish, milSpec`
t_height	Height for the label. Valid Values: `small, medium, large`
t_orient	Orientation for the label. Valid Values: `R0`, which specifies horizontal display, and `R90`, which specifies vertical display.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

s_labelId	Returns an identification number for the subwindow label.
`nil`	Returns `nil` if there is an error.

Example

```
awvPlaceWindowLabel( window(2) list(0 0.5) "R5 = " "varR5")
```

If the value of `varR5` is `1K`, the label `R5 =1K` is displayed at the Waveform window location (0 0.5).

Related Functions

To remove the label, or all the labels identified in a list, from a subwindow, see the awvRemoveLabel function on page-123.

To attach a label to a particular waveform curve in a subwindow, see the awvPlaceWaveformLabel function on page-93.

awvPlaceXMarker

```
awvPlaceXMarker(w_windowId n_xLoc [?label t_label] [?subwindow x_subwindowId])
     => t_xLoc/ t | nil
```

Description

Places a vertical marker at a specific x-coordinate in the optionally specified subwindow of the specified window.

Arguments

w_windowId	Waveform window ID.
n_xLoc	The x-coordinate at which to place the marker.
t_label	Marker label you want to set.
x_subwindowId	Waveform subwindow ID.

Value Returned

t_xLoc	Returns a string of x-coordinates if the command is successful and the vertical marker info form is opened.
t	Returns this when the command is successful but the vertical marker info form is not opened.
nil	Returns nil or an error message.

Examples

```
awvPlaceXMarker( window 5) => "5"
```

Vertical marker info form is opened when the command is executed.

```
awvPlaceXMarker( window 6 ?subwindow 2) => t
```

Vertical marker info form is not opened.

awvPlaceYMarker

```
awvPlaceYMarker(w_windowId n_yLoc [?label t_label][?subwindow x_subwindowId]) =>
     t_yLoc/ t | nil
```

Description

Places a horizontal marker at a specific y-coordinate in the optionally specified subwindow of the specified window.

Arguments

w_windowId	Waveform window ID.
n_yLoc	The y-coordinate at which to place the marker.
t_label	Marker label you want to set.
x_subwindowId	Waveform subwindow ID.

Value Returned

t_yLoc	Returns a string of y-coordinates if the command is successful and the horizontal marker info form is opened.
t	Returns this when the command is successful but the horizontal marker info form is not opened.
nil	Returns nil or an error message.

Examples

```
awvPlaceYMarker( window 5) => "5"
```

Horizontal marker info form is opened when the command is executed.

```
awvPlaceYMarker( window 6 ?subwindow 2) => t
```

Horizontal marker info form is not opened.

awvPlotExpression

```
awvPlotExpression(
     w_windowId
     t_expr
     l_context
     [?expr    l_dispExprList]
     [?index   l_waveIndexList]
     [?color   l_colorList]
     [?lineType     l_styleList]
     [?dataSymbol   l_symbolList]
     [?subwindow    x_subwindow]
     [?yNumber      l_yNumberList]
     [?stripNumber  l_stripNumberList]
     [?showSymbols  l_showList]
     [?lineStyle    l_styleList]
     [?lineThickness l_thicknessList]
     )
     => t | nil
```

Description

Evaluates the *t_expr* expression and assigns the numbers specified in *l_waveIndexList* to the waveforms resulting from the evaluation.

Any existing waveforms with these numbers are overwritten. If you do not specify *l_waveIndexList*, the lowest numbers for the window are assigned. You can provide an optional list of strings, *l_dispExprList*, with this function call. When you supply this list, the strings are displayed in the Waveform window instead of the expressions.

Arguments

w_windowId	Waveform window ID.
t_expr	String containing an expression that is evaluated once when the command is issued and reevaluated at the completion of each simulation in the auto-update mode. *t_expr* can be nil.
l_context	Data context for a particular simulation. If evaluating the expressions requires data generated during a simulation, you must specify this argument. Otherwise, specify `nil`.

l_dispExprList

List of strings to display in the Waveform window instead of the expressions.

l_waveIndexList

List of integer identifiers to assign to the waveform curves.

l_colorList

List specifying the colors for the waveforms. If you do not supply this argument, the default colors are used. The colors that are available are defined in your technology file.
Valid Values: "`y1`" through "`y66`"

l_styleList

List specifying the line styles for the waveforms. If you do not supply this argument, the default line style is used.
Valid Values: `line`, `bar`, `scalarPlot` and `polezero`. Default Value: `Line`.

l_symbolList

List of symbols to use if you want the system to plot data points only (that is, not curves). The system places a symbol on each data point and does not use a line to connect the points. To use a symbol, specify an integer or a single character corresponding to the symbol, as listed below:

To use the symbol	Enter integers	Enter character
+ (plus)	0, 10 or 20	+
. (dot)	1	.
x	2, 9, 19	x or X
square	3, 5, 15	Not supported
circle	4, 6 or 16	o or O
box	5	Not supported

x_subwindow

Number for the subwindow (found in the upper right corner).
Default Value: Current subwindow

l_yNumberList

List of numbers identifying the Y axes.

l_stripNumberList

List of numbers identifying the strips.

l_showList	Flag or list of flags that specifies if the symbols for the traces should be plotted or not. By default, the value is set as `nil` and none of the symbols is plotted. Set the value as `t` to show symbols for all the traces. To set the show status for each symbol, specify a list of flags, one flag for each symbol. The number of flags should match the number of symbols in `l_symbolList`. Each flag specifies if the corresponding symbol in the `l_symbolList` list will be shown or not. Default Value: `nil`
l_styleList	Specifies the line-style of the signal. Valid values: `Solid`, `Dotted`, `Dashed`, `Dotdashed`
l_thicknessList	Specifies the thickness of the signal. Valid values: `Fine`, `Medium`, `Bold`

Value Returned

`t`	Returns `t` when *t_expr* expression is evaluated and the resulting waveforms are plotted.
`nil`	Returns `nil` if there is an error.

Example

```
awvPlotExpression( window( 2 )
    "expr( x sin( x ) linRg( 0 1.5 .5 ) )" nil
    ?expr list( "sine" ) ?index list( 3 ) ?color list( "y2" ) )
```

Displays `sin(x)` from *x* = 0.0 to *x* = 1.5. The expression is evaluated at *x* = 0.0, 0.5, 1.0 and 1.5. The waveform for `sin(x)` is assigned number 3. The waveform is labeled `sine` in the Waveform window, and `sin(x)` is displayed in the *y2* layer color.

awvPrintWaveform

```
awvPrintWaveform( [?output    t_filename | p_port]
     [?numSigDigits  x_sigDigits]
     [?format       s_format]
     [?numSpaces    x_numSpaces]
     [?width        x_width]
     [?from         x_from]
     [?to           x_to]
     [?step         x_step]
     o_waveform1    [o_waveform2 ...] )
     => t | nil
```

Description

Prints the text data of the waveforms specified in the list of waveforms in a result display window.

If you provide a filename as the `?output` argument, this function opens the file and writes the information to it. If you provide a port (the return value of the SKILL `outfile` function), the `awvPrintWaveform` function appends the information to the file that is represented by the port. If you do not provide any argument, it opens a Result Display window and displays the data there. There is a limitation of `awvPrintWaveform` function for precision. It works upto 30 digits for the Solaris port and 18 digits for HP and AIX. If the data is too lengthy to be displayed in the print window, a pop-up form appears indicating this and notifying you that the data will be sent to a default output file or to a filename you specify.

Arguments

t_filename	File in which to write the information. The function opens the file, writes to it, and closes it.
p_port	Port (previously opened with `outfile`) through which to append the information to a file. You are responsible for closing the port. See the `outfile` function for more information.
x_sigDigits	The number of significant digits to print. This value overrides any global precision value set with the `setup` command. Valid values: `1` through `16` Default value: `6`
s_format	The format represents the notation for printed information. This value overrides any global format value set with the `setup`

	command. Valid values: `'suffix`, `'engineering`, `'scientific`, `'none` Default value: `'suffix` The format for each value is `'suffix`: `1m`, `1u`, `1n`,... `'engineering`: `1e-3`, `1e-6`, `1e-9`, ...; `'scientific`: `1.0e-2`, `1.768e-5`, ...;`'none`. The value `'none` is provided so that you can turn off formatting and therefore greatly speed up printing for large data files. For the fastest printing, use the `'none` value and set the `?output` argument to a filename or a port, so that output does not go to the CIW.
x_numSpaces	The number of spaces between columns. Valid values: `1` or greater Default value: `4`
x_width	The width of each column. Valid values: `4` or greater Default value: `14`
x_from	The start value at x axis for the waveform to be printed.
x_to	The end value at x axis for the waveform to be printed.
?step	The step by which text data to be printed is incremented.
o_waveform1	Waveform object representing simulation results that can be displayed as a series of points on a grid. (A waveform object identifier looks like this: `srrWave:`*XXXXX*.)
o_waveform2	Additional waveform object.

Value Returned

`t`	Returns `t` if the text for the waveforms is printed.
`nil`	Returns `nil` and an error message if the text for the waveforms cannot be printed.

Examples

```
awvPrintWaveform( v( "/net56" ) )
=> t
```

Prints the text for the waveform for the voltage of `net56`.

```
awvPrintWaveform( vm( "/net56" ) vp( "/net56" ) )
=> t
```

Prints the text for the waveforms for the magnitude of the voltage of `net56` and the phase of the voltage of `net56`.

```
awvPrintWaveform( ?output "myFile" v( "net55" ) )
=> t
```

Prints the text for the specified waveform to a file named `myFile`.

```
awvPrintWaveform( ?output "./myOutputFile" v("net1") ?from 0 ?to 0.5n ?step 0.1n )
```

Prints the text for the specified waveform from `0` to `0.5n` on the x axis in the incremental steps of `0.1n`.

Related Functions

To plot the Y values in a list against the X values in a list and display the resulting waveforms in a subwindow, see the awvPlotList function on page-106.

To plot the waveforms in a list in a subwindow, see the awvPlotWaveform function on page-112.

To evaluate an expression and specify whether to add the resulting waveforms to the set of existing waveforms, or overwrite the existing waveforms, see the awvSimplePlotExpression function on page-166.

To evaluate an expression and add the resulting waveforms to a subwindow, see the awvAppendExpression function on page-29.

To plot the Y values in a list against the X values in a list and add the resulting waveforms to a subwindow, see the awvAppendList function on page-32.

To add the waveforms in a list to a subwindow, see the awvAppendWaveform function on page-35.

awvPlotList

```
awvPlotList(
     w_windowId
     l_YListList
     l_XList
     [?expr          l_exprList]
     [?index         l_waveIndexList]
     [?color         l_colorList]
     [?lineType      l_styleList]
     [?dataSymbol    l_symbolList]
     [?subwindow     x_subwindow]
     [?yNumber       l_yNumberList]
     [?stripNumber   l_stripNumberList]
     )
     => t | nil
```

Description

Plots the Y values in *l_YListList* against the X values in *l_XList* and displays the resulting waveforms in a subwindow. If you do not specify numbers for the waveforms in *l_waveIndexList*, the lowest *unused* numbers for the subwindow are assigned.

Arguments

w_windowId	Waveform Window ID.
l_YListList	List of lists that specify the different Y values. Each list must have the same number of elements.
l_XList	List that specifies the different X values. The number of elements in this list must equal the number of elements in each list in *l_YListList*
l_exprList	Specifies the expressions to display by the waveform identifiers. If you do not supply this argument, no expressions are displayed beside the waveform identifiers.
l_waveIndexList	List of integers identifiers for waveform curves.
l_colorList	List specifying the colors for the waveforms. If you do not supply this argument, the default colors are used. The colors that are available are defined in your technology file. Valid Values: "y1" through "y66"

l_styleList — List specifying the line styles for the waveforms. If you do not supply this argument, the default line style is used.
Valid Values: `line`, `bar`, `scalarPlot` and `polezero`. Default Value: `Line`.

l_symbolList — List of symbols to use if you want the system to plot data points only (that is, not curves). The system places a symbol on each data point and does not use a line to connect the points. To use a symbol, specify an integer or a single character corresponding to the symbol, as listed below:

To use the symbol	Enter integers	Enter character
+ (plus)	0, 10 or 20	+
. (dot)	1	.
x	2, 9, 19	x or X
square	3, 5, 15	Not supported
circle	4, 6 or 16	o or O
box	5	Not supported

x_subwindow — Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

l_yNumberList — List of numbers identifying the Y axes.

l_stripNumberList — List of numbers identifying the strips.

Value Returned

`t` — Returns `t` when the specified Y values are plotted against the specified X values and the waveforms are displayed in the subwindow.

`nil` — Returns `nil` if there is an error.

Example

```
awvPlotList( window( 2 ) list( list( 1 2 3 )
    list( 2 3 4 ) ) list( 1 2 3 ) ?index list( 1 3 ) )
```

Displays two linear waveforms as waveform 1 and 3.

Related Functions

To evaluate an expression and assign the numbers specified in a list to the waveforms resulting from the evaluation, see the awvPlotExpression function on page-100.

To plot the waveforms in a list in a subwindow, see the awvPlotWaveform function on page-112.

To evaluate an expression and specify whether to add the resulting waveforms to the set of existing waveforms, or overwrite the existing waveforms, see the awvSimplePlotExpression function on page-166.

To evaluate an expression and add the resulting waveforms to a subwindow, see the awvAppendExpression function on page-29.

To plot the Y values in a list against the X values in a list and append the resulting waveforms to a subwindow, see the awvAppendList function on page-32.

To add the waveforms in a list to a subwindow, see the awvAppendWaveform function on page-35.

awvPlotSignals

```
awvPlotSignals(signals ?plotStyle style ?graphType type [?graphModifier modifier]
     [waveType])
```

Description

Displays a signal in the graph window.

Arguments

`signals`	List of signals to be plotted specified in the following format: (*list* (*list resultsDir1* (*list* (*list result1* (*list signal1 signal2* ...) ...) ...))) Remember to put quotation marks before and after each signal name.
style	Plot destination. Valid values: `Append`, `Replace`, `New Window`, `New Subwindow` Remember to put quotation marks before and after the style.
type	Type of plot. Valid values: `Default`, `Rectangular`, `Histogram`, `Polar`, `Impedance` `Admittance`, `RealvsImag` Remember to put quotation marks before and after the graph type.
modifier	X axis of graph for rectangular graphs Valid values: `Magnitude`, `Phase`, `WPhase`, `Real`, `Imaginary`, `dB10`, `dB20`
waveType	Specifies whether the signal is Y versus Y or not. Valid values: `YvsY`, `nil` This is an optional argument.
lineStyle	Specifies the line style of the signal. Valid values: Solid, Dotted, Dashed, Dotdashed This is an optional argument.
lineThickness	Specifies the line thickness of the signal. Valid values: Fine, Medium, Thick, and ExtraThick. This is an optional argument.

Value Returned

Example

```
awvPlotSignals((("./ampsim.raw" (("ac-ac" ("net10"))))) "plotStyle" "Append"
"graphType" "Default")
```

awvPlotSimpleExpression

```
awvPlotSimpleExpression(expression @key [plotStyle style] [graphType type]
     [graphModifier modifier])
```

Description

Evaluates an expression and plots the resulting waveform.

Arguments

expression	Expression to be evaluated and plotted.
style	Plot destination. Valid values: `Append`, `Replace`, `New Window`, `New Subwindow` Remember to put quotation marks before and after the style.
type	Type of plot. Valid values: `Default`, `Rectangular`, `Histogram`, `Polar`, `Impedance Admittance`, `RealvsImag` Remember to put quotation marks before and after the graph type.
modifier	X axis of graph for rectangular graphs Valid values: `Magnitude`, `Phase`, `WPhase`, `Real`, `Imaginary`, `dB10`, `dB20`
lineStyle	Specifies the line style of the signal. Valid values: Solid, Dotted, Dashed, Dotdashed This is an optional argument.
lineThickness	Specifies the line thickness of the signal. Valid values: Fine, Medium, Thick, and ExtraThick. This is an optional argument.

Example

```
awvPlotSimpleExpression(getData(\net10\ ?result \tran\ ?resultsDir \./ampsim.raw\)
- getData(\net35\ ?result \tran\ ?resultsDir \./ampsim.raw\) ?plotStyle New Window
?graphType Default)
```

awvPlotWaveform

```
awvPlotWaveform(
     w_windowId
     l_waveform
     [?expr         l_exprList]
     [?index        l_waveIndexList]
     [?color        l_colorList]
     [?lineType     l_styleList]
     [?dataSymbol   l_symbolList]
     [?subwindow    x_subwindow]
     [?yNumber      l_yNumberList]
     [?stripNumber  l_stripNumberList]
     [?showSymbols  l_showList]
     [?lineStyle    l_styleList]
     [?lineThickness l_thicknessList]
     )
     => t | nil
```

Description

Plots the waveforms in the *l_waveform* list in a subwindow. If you do not specify numbers for the waveforms in *l_waveIndexList*, the lowest *unused* numbers for the subwindow are assigned.

Arguments

w_windowId	Waveform window ID.
l_waveform	List of waveform objects.
l_exprList	Specifies the expressions to display next to the waveform identifiers. If you do not specify *l_exprList*, no expressions are displayed beside the waveform identifiers.
l_waveIndexList	List of integer identifiers to assign to the waveform curves.
l_colorList	List specifying the colors for the waveforms. If you do not supply this argument, the default colors are used. The colors that are available are defined in your technology file. The color information can also be provided in the RGB format, such as `0X00FF50`. Valid Values: "`y1`" through "`y66`"

l_styleList List specifying the line styles for the waveforms. If you do not supply this argument, the default line style is used.
Valid Values: `line`, `bar`, `scalarPlot` and `polezero`. Default Value: `Line`.

l_symbolList List of symbols to use if you want the system to plot data points only (that is, not curves). The system places a symbol on each data point and does not use a line to connect the points. To use a symbol, specify an integer or a single character corresponding to the symbol, as listed below:

To use the symbol	Enter integers	Enter character
+ (plus)	0, 10 or 20	+
. (dot)	1	.
x	2, 9, 19	x or X
square	3, 5, 15	Not supported
circle	4, 6 or 16	o or O
box	5	Not supported

x_subwindow Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

l_yNumberList List of numbers identifying the Y axes.

l_stripNumberList List of numbers identifying the strips.

l_showList Flag or list of flags that specifies if the symbols for the traces should be plotted or not. By default, the value is set as `nil` and none of the symbols is plotted. Set the value as `t` to show symbols for all the traces. To set the show status for each symbol, specify a list of flags, one flag for each symbol. The number of flags should match the number of symbols in `l_symbolList`. Each flag specifies if the corresponding symbol in the `l_symbolList` list will be shown or not. Default Value: `nil`

l_styleList Specifies the line-style of the signal.
Valid values: `Solid`, `Dotted`, `Dashed`, `Dotdashed`

l_thicknessList	Specifies the thickness of the signal. Valid values: `Fine`, `Medium`, `Bold`

Value Returned

`t`	Returns `t` when the waveforms specified in the *l_waveform* list are plotted.
`nil`	Returns `nil` if there is an error.

Example

```
awvPlotWaveform( window( 2 ) list( w1 w2 ) ?expr
    list( "wave1" "wave2" ) )
```

Plots waveforms *w1* and *w2* in the Waveform window and assigns them numbers 1 and 2, respectively. The *wave1* and *wave2* expressions are displayed next to their waveform identifiers.

Related Functions

To delete all the waveforms in a subwindow, see the awvDeleteAllWaveforms function on page-47.

To evaluate an expression and assign the numbers specified in a list to the waveforms resulting from the evaluation, see the awvPlotExpression function on page-100.

To plot the Y values in a list against the X values in a list and display the resulting waveforms in a subwindow, see the awvPlotList function on page-106.

To evaluate an expression and specify whether to add the resulting waveforms to the set of existing waveforms, or overwrite the existing waveforms, see the awvSimplePlotExpression function on page-166.

To evaluate an expression and add the resulting waveforms to a subwindow, see the awvAppendExpression function on page-29.

To plot the Y values in a list against the X values in a list and add the resulting waveforms to a subwindow, see the awvAppendList function on page-32.

To add the waveforms in a list to a subwindow, see the awvAppendWaveform function on page-35.

awvPlotWaveformOption

```
awvPlotWaveformOption(
     w_windowId
     x_waveIndex
     t_component
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the plot option for a particular waveform in a subwindow.

Arguments

w_windowId	Waveform Window ID.
x_waveIndex	Integer identifying the waveform curve.
t_component	Plot option for the waveform. This option is ignored unless the display mode is set to *strip* or *composite*. (The imaginary part of the waveform is plotted against its real part when the display mode is Smith.) Valid values are
magnitude	Plots the magnitude of the waveform against an independent variable.
dB10	Calculates *10*log* of the magnitude of each point in the waveform and plots the results against an independent variable.
dB20	Calculates *20*log* of the magnitude of each point in the waveform and plots the results against an independent variable.
dBm	Adds 30 to the value calculated by dB10.
phaseDeg	Plots the phase, in degrees, of the waveform against an independent variable.
phaseRad	Plots the phase, in radians, of the waveform against an independent variable.

`wrappedPhaseDeg`	Plots the wrapped phase, in degrees, of the waveform against an independent variable.
`wrappedPhaseRad`	Plots the wrapped phase, in radians, of the waveform against an independent variable.
`real`	Plots the real part of the waveform against an independent variable.
`imaginary`	Plots the imaginary part of the waveform against an independent variable.

`x_subwindow` Number for the subwindow (found in the upper right corner).
Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the plot option is set for the waveform.
`nil`	Returns `nil` if there is an error.

Example

```
awvPlotWaveformOption( window( 2 ) 1 "phaseDeg" )
```

Displays the phase (in degrees) of the waveform whose index is one.

awvRedisplaySubwindow

```
awvRedisplaySubwindow(
     w_windowId
     [?subwindow    x_subwindow]
     readData)
     => t | nil
```

Description

Refreshes the display for a subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow
readData	Boolean parameter. When set to `t`, graph reads the simulation data, refreshes the subwindow with new data and updates the traces if necessary. If set to `nil`, a simple UI refresh takes place for the graph subwindow.

Value Returned

`t`	Returns `t` when the subwindow is redrawn.
`nil`	Returns `nil` if the Waveform window or the subwindow do not exist.

Example

```
awvRedisplaySubwindow( window(2) ?subwindow 3 )
=> t
```

Refreshes the display for subwindow 3.

Related Function

To refresh the display for a Waveform window, see the awvRedisplayWindow.

awvRedrawWindowMenuCB

```
awvRedisplaySubwindow()
     => t | nil
```

Description

Redraws all the objects (e.g. waveforms, markers) in the waveform window.

The function is defined in `dfII/etc/context/awv.cxt`.

Arguments

None.

Value Returned

`t`	Returns `t` if the command is successful.
`nil`	Otherwise, returns `nil`.

Example

```
awvRedrawWindowMenuCB() => t
```

awvRedisplayWindow

```
awvRedisplayWindow( w_windowId )
     => t | nil
```

Description

Refreshes the display for a Waveform window.

Arguments

w_windowId	Waveform window ID.

Value Returned

`t`	Returns `t` when the Waveform window is redrawn.
`nil`	Returns `nil` if the specified Waveform window does not exist.

Related Function

To refresh the display for a subwindow, see the awvRedisplaySubwindow function on page-117.

awvRemoveDate

```
awvRemoveDate( w_windowId )
     => t | nil
```

Description

Removes the date and time from the Waveform window.

Arguments

w_windowId	Waveform window ID.

Value Returned

`t`	Returns `t` when the date and time are removed.
`nil`	Returns `nil` if the specified Waveform window does not exist.

Related Function

To display the date and time in a Waveform window, see the awvDisplayDate function on page-53.

awvResumeViVA

```
awvResumeViVA()
```

Description

This function, when called from CIW, resumes ViVA (if suspended) provided all the licensing requirements are met.

Arguments

None

Value Returned

None

Example

```
awvResumeViVA()
```

awvRemoveLabel

```
awvRemoveLabel(
     w_windowId
     {s_id | l_id}
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Removes the label, or all the labels identified in a list, from a subwindow.

Arguments

w_windowId	Waveform window ID.
s_id	Label to remove.
l_id	List of labels to remove.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the label or labels are removed.
`nil`	Returns `nil` if there is an error.

Related Functions

To attach a label to a particular waveform curve in a subwindow, see the awvPlaceWaveformLabel function on page-93.

To display a label in a subwindow, see the awvPlaceWindowLabel function on page-96.

awvRemoveSubwindowTitle

```
awvRemoveSubwindowTitle(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Removes the title from a subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the title is removed.
`nil`	Returns `nil` if there is an error, such as the specified Waveform window does not exist or the specified subwindow does not have a title.

Related Functions

To display a title in a subwindow, see the awvDisplaySubwindowTitle function on page-55.

To display a title on a Waveform window, see the awvDisplayTitle function on page-56.

To remove a title from a Waveform window, see the awvRemoveTitle function on page-125.

awvRemoveTitle

```
awvRemoveTitle( w_windowId )
     => t | nil
```

Description

Removes the title from a Waveform window.

Arguments

w_windowId	Waveform window ID.

Value Returned

t	Returns nil when the title is removed.
nil	Returns nil if the specified Waveform window does not exist.

Related Functions

To display a title on a Waveform window, see the awvDisplayTitle function on page-56.

To display a title in a subwindow, see the awvDisplaySubwindowTitle function on page-55.

To remove a title from a subwindow, see the awvRemoveSubwindowTitle function on page-124.

awvResetAllWindows

```
awvResetAllWindows( [?force   g_force] )
     => t
```

Description

Resets all the windows returned by `awvGetWindowList`. The contents of the windows are erased, and any subwindows whose update statuses are *on* are deleted. To delete all subwindows, regardless of update status, set *`g_force`* to `t`.

Note: The Waveform Windows remain at their current sizes and locations.

Arguments

`g_force`	Boolean flag that specifies whether all the subwindows of the Waveform Windows are deleted, regardless of update status. Valid Values: `t` specifies that all the subwindows are deleted, `nil` specifies that only subwindows whose update statuses are *on* are deleted. Default value: `nil`.

Value Returned

`t`	Returns `t` when the Waveform Windows are reset.

Related Function

To reset a particular Waveform window, see the awvResetWindow function on page-127.

awvResetWindow

```
awvResetWindow(
     w_windowId
     [?force   g_force]
     )
     => t | nil
```

Description

Resets a Waveform window to the state of a new window. The contents of the window are erased, and any subwindows whose update statuses are *on* are deleted. To delete all subwindows, regardless of update status, set *g_force* to `t`.

Note: The Waveform window remains at its current size and location.

Arguments

w_windowId	Waveform window ID.
g_force	Boolean flag that specifies whether all the subwindows of the Waveform window are deleted, regardless of update status. Valid Values: `t` specifies that all subwindows are deleted, `nil` specifies that only subwindow whose update statuses are *on* are deleted. Default value: `nil`.

Value Returned

`t`	Returns `t` when the Waveform window is reset.
`nil`	Returns `nil` if the specified Waveform window does not exist.

Example

```
awvResetWindow( window(5) ?force t )
=> t
```

Erases the contents of Waveform window 5 and sets the window to the state of a new window. All the subwindows are deleted, regardless of update status.

Related Function

To reset all the windows returned by `awvGetWindowList`, see the awvResetAllWindows function on page-126.

awvRfLoadPull

```
awvRfLoadPull(
     w_wave
     [?maxValue    x_maxValue]
     [?minValue    x_minValue]
     [?numCont     x_numCont]
     [?closeCont   g_closeCont]
     [?name        t_name]
     )
     => t | nil
```

Description

Draws load pull contour for the given waveform of PSS analysis. This function works only for two-dimensional sweep PSS results. The inner sweep should be phase and the outer sweep should be mag.

Arguments

w_wave	Signal waveform.
x_maxValue	Largest value of the contour to be drawn. Default value: `nil`. Specifies that the largest value is to be taken from the results.
x_minValue	Smallest value of the contour to be drawn. Default value: `nil`. Specifies that the smallest value is to be taken from the results.
x_numCont	Number of points on the contour. Default value: `8`
g_closeCont	Boolean flag that specifies if a closed or open contour is to be drawn. Valid values: `t` specifies that a closed contour is to be drawn. `nil` specifies that an open contours is to be drawn.
t_name	Name of the contour. Default value: `p`

Value Returned

`t`	Returns `t` when the Load Pull contour is drawn.
`nil`	Returns `nil` if the specified waveform does not exist.

Example

```
awvRfLoadPull(ip3_wave nil nil 9 t "ip3")
=> t
```

awvSaveWindow

```
awvSaveWindow(
     w_windowId
     t_fileName
     )
     => t | nil
```

Description

Saves the state of a Waveform window to a file. You can specify a path name, or you can specify only the file name. If you provide only a file name, the file is placed in the directory in which you started the software.

Arguments

w_windowId	Waveform window ID.
t_fileName	Name of the file in which to store the state of the Waveform window.

Value Returned

t	Returns t when the window state is stored.
nil	Returns nil if there is an error.

Example

```
awvSaveWindow( window(4) "/tmp/my_file" )
=> t
```

Saves the state of Waveform window 4 in *my_file* in the /tmp directory.

Related Function

To initialize a Waveform window from information saved in a file, see the awvLoadWindow function on page-89

awvSaveWindowImage

```
awvSaveWindowImage(
     w_window
     t_path
     t_filePrefix
     g_cardLayout
     )
     => l_files
```

Description

Saves the image of a plot window in `.png` format.

Arguments

w_windowId	Waveform window ID.
t_fileName	Path where you want to save the image file.
t_filePrefix	Prefix string, which is to be prefixed in the name of file being saved.
g_cardLayout	Card layout. Valid value: `t` if the window is in card layout mode.

Value Returned

`l_files`	Returns the list of image files saved.

awvSaveMenuCB

```
awvSaveMenuCB() => t | nil
```

Description

Displays the *Save (window -> Save...)* Option Menu.

The function is defined in `dfII/etc/context/awv.cxt`.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awvSaveMenuCB() => t
```

awvSetCurrentSubwindow

```
awvSetCurrentSubwindow(
     w_windowId
     x_subwindow
     )
     => t | nil
```

Description

Specifies *x_subwindow* as the current subwindow.

Arguments

w_windowId	Waveform window ID.
x_subwindow	Number of the subwindow (found in the upper right corner) that is to become the current subwindow.

Value Returned

t	Returns t when the subwindow is set to *x_subwindow*.
nil	Returns nil if there is an error.

Example

```
awvSetCurrentSubwindow( window(2) 1 )
=> t
```

Specifies subwindow 1 as the current subwindow.

Related Function

To get the current subwindow, see the awvGetCurrentSubwindow function on page-62.

awvSetCurrentWindow

```
awvSetCurrentWindow( w_windowId )
     => t | nil
```

Description

Specifies *w_windowId* as the current Waveform window.

Arguments

w_windowId	Waveform window ID for the window to become the current window.

Value Returned

t	Returns t when the current window is set.
nil	Returns nil if there is an error.

Example

```
awvSetCurrentWindow( window(2) )
=> t
```

Specifies window number 2 as the current Waveform window.

Related Function

To return the window ID for the current Waveform window, see the awvGetCurrentWindow function on page-63.

awvSetCursorPrompts

```
awvSetCursorPrompts(
     w_windowId
     x_yNumber
     t_xPrompt
     t_yPrompt
     [?stripNumber  x_stripNumber]
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the tracking cursor prompts for the waveforms around a particular Y axis and a particular strip in a subwindow. If you specify `nil` for the prompts, the default prompts are used.

Arguments

w_windowId	Waveform window ID.
x_yNumber	Specifies the Y axis to have the tracking cursor prompts. Valid Values: 1 through 4
t_xPrompt	Prompt to put next to the X axis value.
t_yPrompt	Prompt to put next to the Y axis value.
x_stripNumber	Specifies the strip in which the tracking cursor prompts are to be set. Valid Values: 1 through 20
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the tracking cursor prompts are set.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetCursorPrompts( window( 2 ) 1 "Time" "Voltage" )
```

When the tracking cursor is on a waveform on the Y axis (Y1), the tracking cursor banner has the word `Time` next to the X axis value and the word `Voltage` next to the Y axis value.

awvSetDisplayMode

```
awvSetDisplayMode(
     w_windowId
     t_mode
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the display mode of a subwindow.

Arguments

w_windowId	Waveform window ID.
t_mode	String representing the display mode for the subwindow. Valid Values: `strip`, `composite`, or `smith`.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the display mode of the subwindow is set.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetDisplayMode( window(2) "strip" ?subwindow 3 )
=> t
```

Sets the display of subwindow 3 to strip mode.

Related Functions

To return the display mode of a subwindow, see the awvGetDisplayMode function on page-64.

awvSetDisplayStatus

```
awvSetDisplayStatus(
     w_windowId
     g_enable
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Turns the display of a subwindow *on* or *off* based on the value of the *g_enable* flag. You can use this function to hide a subwindow without deleting it.

Arguments

w_windowId	Waveform window ID.
g_enable	Boolean flag that specifies whether the display of a subwindow is turned *on* or *off*. If this argument is set to `nil` and the specified subwindow is displayed, the subwindow display is hidden. If this argument is set to non-`nil` and the specified subwindow is not displayed, the subwindow is displayed again.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the subwindow display is turned *off* or *on*.
`nil`	Returns `nil` if the Waveform window or the subwindow do not exist.

Example

```
awvSetDisplayStatus( window(2) nil ?subwindow 1 )
=> t
```

Turns *off* the display of subwindow 1.

awvSetInitializationTimeout

```
awvSetInitializationTimeout( x_timeOut )
     => x_timeOut
```

Description

Sets the time-out period (in seconds) for ADE to establish connection with Virtuoso Visualization and Analysis XL and returns the same value. The default value of the time-out period is `120` seconds.

Arguments

x_timeOut	Time-out period (in seconds).

Value Returned

x_timeOut	Time-out period (in seconds).

Example

```
awvSetInitializationTimeout( 240 )
=> 240
```

This will set the time-out period to `240` seconds, so that ADE will keep trying to establish a connection with Virtuoso Visualization and Analysis XL for up to `240` seconds.

awvSetOptionDefault

```
awvSetOptionDefault( S_name )
     => t | nil
```

Description

Restores a Waveform window option to its default value. The option takes effect for any Waveform Windows or subwindows that are opened after the option is set.

For a list of Waveform window default options that you can change, see the table at the beginning of this chapter.

Arguments

S_name	Name of the option to restore. The option name can be a string or a symbol.

Value Returned

`t`	Returns `t` when the option is restored to its default value.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetOptionDefault( "mode" )
=> t
```

Sets the *mode* option back to its default value, which is `composite`.

Related Functions

To set a Waveform window option to a particular value, see the awvSetOptionValue function on page-142.

awvSetOptionValue

```
awvSetOptionValue(
     S_name
     g_value
     )
     => g_value | nil
```

Description

Sets a Waveform window option. The option takes effect for any Waveform Windows or subwindows that are opened after the option is set.

For a list of Waveform window default options that you can change, see the table at the beginning of this chapter.

Arguments

S_name	Name of the Waveform window option. The option name can be a string or a symbol.
g_value	Value for the option.

Value Returned

g_value	Returns the new value of the option.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetOptionValue( "displayGrids" t )
=> t
```

Turns on the grid display option for a Waveform window.

Related Functions

To restore a Waveform window option to its default value, see the awvSetOptionDefault function on page-141.

awvSetOrigin

```
awvSetOrigin(
     w_windowId
     l_origin
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the axis origin of a subwindow to a new location. This function takes effect only when the waveform display is in the composite mode with only one Y axis displayed.

Arguments

w_windowId	Waveform window ID.
l_origin	List of two numbers in waveform coordinates that specify the new location for the axis origin. If *l_origin* is `nil`, the axis origin goes to the most logical position according to the data.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the axis origin is set.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetOrigin( window(2) list( 4.5 5.5 ) ?subwindow 3 )
```

For subwindow 3, draws the X axis at Y = 5.5 and the Y axis at X = 4.5.

awvSetPlotStyle

```
awvSetPlotStyle(
     w_windowId
     S_style
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the plotting style for all the waveforms in a subwindow. If the plotting style is *bar* and the display mode is *smith*, then the plotting style is ignored until the display mode is set to *strip* or *composite.*

Arguments

w_windowId	Waveform window ID.
S_style	Plotting style for the subwindow. Valid Values: `auto`, `scatterPlot`, `bar`, or `joined`
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the plotting style is set.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetPlotStyle( window(2) "joined" ?subwindow 1 )
```

In subwindow 1, joins the adjacent points of the waveforms plotted with straight lines.

Related Function

To get the plotting style for the waveforms in a subwindow, see the awvGetPlotStyle function on page-69.

awvSetSavePromptNeeded

```
awvSetSavePromptNeeded( g_savePromptNeeded )
     => t
```

Description

Specifies whether you get a dialog box that asks if you want to save unsaved Waveform Windows when you close them or exit the Cadence software. By default, this option is set to *not* prompt you for unsaved Waveform Windows.

Arguments

g_savePromptNeeded
: Boolean flag that specifies whether or not you are prompted to save Waveform Windows.
Valid Values: `t` specifies that you are prompted to save Waveform Windows, `nil` specifies that you are not prompted to save Waveform Windows.
Default Value: `nil`

Value Returned

`t`
: This function always returns `t`.

Example

```
awvSetSavePromptNeeded( t )
=> t
```

Specifies that the user is prompted to save unsaved Waveform Windows.

Related Function

To return the value of the Boolean flag that specifies whether or not you are prompted to save Waveform Windows, see the awvGetSavePromptNeeded function on page-70.

awvSetSmithModeType

```
awvSetSmithModeType(
     w_windowId
     t_type
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the Smith display mode type for a subwindow. The display mode type takes effect only when the subwindow is set to *fs*.

Arguments

w_windowId	Waveform window ID.
t_type	Type of Smith display. Valid Values: `impedance`, `admittance`, or `polar`
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the Smith display is set.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetSmithModeType( window(2) "polar" ?subwindow 3 )
=> t
```

Sets the Smith display of subwindow 3 to polar.

Related Functions

To return the Smith display type of a subwindow, see the awvGetSmithModeType function on page-71.

To get the display mode of a subwindow, see the awvGetDisplayMode function on page-64.

To set the display of a subwindow to a different type, see the awvSetDisplayMode function on page-138.

awvSetSmithXLimit

```
awvSetSmithXLimit(
     w_windowId
     l_minMax
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Set the X axis display limits for a subwindow with a Smith display mode. This command does not take effect if the display mode is set to *strip* or *composite*.

Arguments

w_windowId	Waveform window ID.
l_minMax	List of two numbers in waveform coordinates that describe the limits for the display. The first number is the minimum and the second is the maximum. If this argument is set to `nil`, the limit is set to *autoscale*.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the X axis display limits are set.
`nil`	Returns `nil` if the Waveform window or subwindow do not exist.

Example

```
awvSetSmithXLimit( window(3) list(0 20) ?subwindow 2)
=> t
```

For subwindow 2, sets the X axis to display from 0 to 20.

Related Functions

To set the Y axis display limits for a subwindow with a Smith display mode, see the awvSetSmithYLimit function on page-150.

To set the X axis display limits for a subwindow, see the awvSetXLimit function on page-158.

To set the Y axis display limits for a subwindow, see the awvSetYLimit function on page-162.

awvSetSmithYLimit

```
awvSetSmithYLimit(
     w_windowId
     l_minMax
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Set the Y axis display limits for a subwindow with a Smith display mode. This command does not take effect if the display mode is set to *strip* or *composite*.

Arguments

w_windowId	Waveform window ID.
l_minMax	List of two numbers in waveform coordinates that describe the limits for the display. The first number is the minimum and the second is the maximum. If this argument is set to `nil`, the limit is set to autoscale.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the Y axis display limits are set.
`nil`	Returns `nil` if the Waveform window or subwindow do not exist.

Example

```
awvSetSmithYLimit( window(7) list(0 15) ?subwindow 2)
=> t
```

For subwindow 2, sets the Y axis to display from 0 to 15.

Related Functions

To set the X axis display limits for a subwindow with a Smith display mode, see the awvSetSmithXLimit function on page-148.

To set the X axis display limits for a subwindow, see the awvSetXLimit function on page-158.

To set the Y axis display limits for a subwindow, see the awvSetYLimit function on page-162.

awvSmithAxisMenuCB

```
awvSmithAxisMenuCB() => t | nil
```

Description

Displays the *Axes (Smith Plot)* Option Menu.

The function is defined in `dfII/etc/context/awv.cxt`.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awvSmithAxisMenuCB() => t
```

awvSetUpdateStatus

```
awvSetUpdateStatus(
     w_windowId
     g_enable
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Turns the update status of a subwindow *on* or *off*. If the update status is *off*, the subwindow display does not change when you tell the Waveform window to update the results.

Arguments

w_windowId	Waveform window ID.
g_enable	Boolean flag that specifies whether the update status is *on* or *off*. If this argument is set to `nil` and the update status of the specified subwindow is *on*, the update status is turned *off*. If this argument is set to non-`nil` and the update status of the subwindow is *off*, the update status is turned *on*.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the update status is turned on or off.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetUpdateStatus( window(2) t ?subwindow 1 )
=> t
```

Turns on the update status for subwindow 1.

awvSetWaveformDisplayStatus

```
awvSetWaveformDisplayStatus(
     w_windowId
     x_waveIndex
     g_enable
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Turns the display of a curve *on* or *off* based on the value of the *g_enable* flag. You can use this function to hide a curve without deleting it.

Arguments

w_windowId	Waveform window ID.
x_waveIndex	Integer identifying the waveform curve. (This is the integer identifier that you supplied in the *l_waveIndexList* argument in a previous Waveform window SKILL function.)
g_enable	Boolean flag that specifies whether the curve display is turned *on* or *off*. If this argument is set to `nil` and the specified curve is displayed, the curve display is removed. If this argument is set to non-`nil` and the specified curve is not displayed, the curve display is turned *on*.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the curve display is turned on or off.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetWaveformDisplayStatus( window(2) 2 nil ?subwindow 1 )
=> t
```

Turns off the display of curve 2 in subwindow 1.

awvSetWaveNameList

```
awvSetWaveNameList(
     list( list(trace_id1 trace_id2 ...)
     list(name1 name2 ...)
     )
     => t | nil
```

Description

Sets the names of the list of waveform curves returned by the awvGetWaveNameList function.

Arguments

trace_id	The ID of a waveform curve returned by the awvGetWaveNameList function.
name	The name of the waveform curve to be used for a `trace_id`.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
awvSetWaveNameList( list( list(1 2) list("wave1" "wave2"))
```

Sets the name of the waveform curve with the IDs `1` and `2` to `wave1` and `wave2` respectively.

awvSetXAxisLabel

```
awvSetXAxisLabel(
     w_windowId
     g_label
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Lets you specify an X axis label to replace the automatically computed label.

Arguments

w_windowId	Waveform window ID.
g_label	Text for the X axis label. You can set this argument to `nil` to display the automatically computed label.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
awvSetXAxisLabel( window(4) "Time in Seconds" ?subwindow 3)
=> t
```

Sets the X axis label of subwindow 3 to *Time in Seconds*.

Related Functions

To return the current X axis label, see the awvGetXAxisLabel function on page-77.

To specify a Y axis label to replace the automatically computed label, see the awvSetYAxisLabel function on page-160.

awvSetXLimit

```
awvSetXLimit(
     w_windowId
     l_minMax
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the X axis display limits for a subwindow. This command does not take effect if the display mode is set to Smith.

Arguments

w_windowId	Waveform window ID.
l_minMax	List of two numbers in waveform coordinates that describe the limits for the display. The first number is the minimum and the second is the maximum. If this argument is set to `nil`, the limit is set to *autoscale.*
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the X axis display limits are set.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetXLimit( window(2) list( 1 100 ) ?subwindow 3 )
```

For subwindow 3, sets the X axis to display from 1 to 100.

Related Function

To set the Y axis display limits for the waveforms associated with a particular Y axis a subwindow, see the awvSetYLimit function on page-162.

awvSetXScale

```
awvSetXScale(
     w_windowId
     t_scale
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the display mode of the X axis in a subwindow. This command does not take effect if the display mode is set to Smith.

Arguments

w_windowId	Waveform window ID.
t_scale	Specifies the scale for the X axis. Valid Values: `auto`, `log`, or `linear`
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the scale for the X axis is set.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetXScale( window( 2 ) "log" ?subwindow 3 )
```

For subwindow 3, sets the X axis to display logarithmically.

Related Function

To set the Y axis of a subwindow to display logarithmically, see the awvLogYAxis function on page-90.

awvSetYAxisLabel

```
awvSetYAxisLabel(
     w_windowId
     x_yNumber
     g_label
     [?subwindow    x_subwindow]
     [?stripNumber  x_stripNumber]
     )
     => t | nil
```

Description

Lets you specify a Y axis label to replace the automatically computed label.

Arguments

w_windowId	Waveform window ID.
x_yNumber	Specifies the Y axis whose label you want to specify. Valid Values: 1 through 4
g_label	Text for the Y axis label. You can set this argument to `nil` to display the automatically computed label.
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow
x_stripNumber	Specifies the strip containing the Y axis whose label you want to specify. Valid Value: Any number greater than 0. Default Value: 1

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
awvSetYAxisLabel( window(8) 2 "Voltage" ?subwindow 3 )
=> t
```

Sets the label display for Y axis 2 in subwindow 3 to *Voltage*.

Related Functions

To return the current Y axis label, see the awvGetYAxisLabel function on page-80.

To specify an X axis label to replace the automatically computed one, see the awvSetXAxisLabel function on page-157.

awvSetYLimit

```
awvSetYLimit(
     w_windowId
     x_yNumber
     l_minMax
     [?stripNumber  x_stripNumber]
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Sets the Y axis display limits for the waveforms associated with a particular Y axis and strip in a subwindow. If you don't specify *x_stripNumber*, the limits are applied when the subwindow is in *composite* mode.

Arguments

w_windowId	Waveform window ID.
x_yNumber	Specifies the Y axis to have the limited display. Valid Values: 1 through 4
l_minMax	List of two numbers in waveform coordinates that describe the limits for the display. The first number is the minimum and the second is the maximum. If this argument is set to `nil`, the limit is set to *autoscale*.
x_stripNumber	Specifies the strip in which the Y display is to be limited. Valid Values: Integer value greater than or equal to 1
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the Y axis display limits are set.
`nil`	Returns `nil` if there is an error.

Example

```
awvSetYLimit( window(2) 1 list( 4.5 7.5 ) )
```

Sets the Y axis (Y1) to display from 4.5 to 7.5. This takes effect only in *composite* mode.

```
awvSetYLimit( window(2) 1 list( 4.5 7.5 ) ?stripNumber 2 )
```

Sets the Y axis (Y1) in strip 2 to display from 4.5 to 7.5. This takes effect only in *strip* mode.

Related Function

To set the X axis display limits for a subwindow, see the awvSetXLimit function on page-158.

awvSetYRange

```
awvSetYRange(
     w_windowId
     x_yNumber
     n_range
     [?subwindow    x_subwindow]
     [?stripNumber  x_stripNumber]
     )
     => t | nil
```

Description

Sets the range for the Y axis.

The range is the difference between the maximum Y value and the minimum Y value, and the range must be positive. The maximum value for the Y axis is automatically computed according to the waveforms. The minimum is calculated with the range argument.

Arguments

w_windowId	Waveform window ID.
x_yNumber	Specifies the Y axis whose range you want to set. Valid Values: 1 through 4
n_range	Amount to subtract from the maximum value to determine the minimum value (for the Y axis).
x_subwindow	Number for the subwindow (found in the upper right corner). Default Value: Current subwindow
x_stripNumber	Specifies the strip containing the Y axis whose range you want to set.

Value Returned

`t`	Returns `t` when the range is set.
`nil`	Returns `nil` otherwise.

Example

```
awvSetYRange( window(3) 2 10 ?subwindow 4 )
=> t
```

Specifies that the difference between the maximum Y axis value and the minimum Y axis for Y axis 2 is 10.

Related Function

To set the Y axis display limits for the waveforms associated with a particular Y axis of a subwindow, see the awvSetYLimit function on page-162.

awvSimplePlotExpression

```
awvSimplePlotExpression(
     w_windowId
     t_expr
     l_context
     g_replace
     [?expr         l_dispExprList]
     [?subwindow    x_subwindow]
     [?yNumber      l_yNumberList]
     [?stripNumber  l_stripNumberList]
     [?showSymbols  l_showList]
     [?lineStyle    l_styleList]
     [?color        l_colorlist]
     [?lineThickness l_thicknessList]
     )
     => t | nil
```

Description

If *g_replace* is set to `t`, this function evaluates the *t_expr* expression and gives the resulting waveforms the lowest numbers already assigned to existing curves.

The old curves are overwritten. Otherwise, the resulting curves are given new numbers, which are the next lowest and in sequence, and the old curves are not overwritten.

You can provide an optional list of strings, *l_dispExprList*, with this function call. When you supply this list, the strings are displayed in the Waveform window instead of the expressions.

Arguments

w_windowId	Waveform window ID.
t_expr	String containing an expression that is evaluated once when the command is issued and reevaluated at the completion of each simulation in the auto-update mode. *t_expr* can be `nil`.
l_context	Data context for a particular simulation. If evaluating the expressions requires data generated during a previously saved simulation or requires simulation data that is being generated in real time, you must specify this argument. Otherwise, specify `nil`.

g_replace Flag that specifies which numbers to give the resulting waveforms.
Valid Values:

`non-nil`	Specifies that the waveforms are given the lowest numbers, even if the numbers are already assigned to existing curves. In this case, the existing curves are overwritten.
`nil`	Specifies that the resulting curves are given new numbers, which are the next lowest and in sequence.

l_dispExprList
List of strings to display in the Waveform window instead of the expressions.

x_subwindow Number for the subwindow (found in the upper right corner).
Default Value: Current subwindow

l_yNumberList
List of numbers identifying the Y axes.

l_stripNumberList
List of numbers identifying the strips.

l_showList Flag or list of flags that specifies if the symbols for the traces should be plotted or not. By default, the value is set as `nil` and none of the symbols is plotted. Set the value as `t` to show symbols for all the traces. To set the show status for each symbol, specify a list of flags, one flag for each symbol. The number of flags should match the number of symbols in `l_symbolList`. Each flag specifies if the corresponding symbol in the `l_symbolList` list will be shown or not.
Default Value: `nil`

l_styleList Specifies the line-style of the signal.
Valid values: `Solid`, `Dotted`, `Dashed`, `Dotdashed`

l_colorList List specifying the colors for the waveforms. If you do not supply this argument, the default colors are used. The colors that are available are defined in your technology file.
Valid Values: "`y1`" through "`y66`"

l_thicknessList	Specifies the thickness of the signal. Valid values: `Fine`, `Medium`, `Bold`

Value Returned

`t`	Returns `t` when *t_expr* is evaluated the resulting waveform curves are plotted.
`nil`	Returns `nil` if there is an error.

Example

```
awvSimplePlotExpression( window( 2 )
    "expr( x sin( x ) linRg( 0 1.5 .5 ) )" nil t )
```

Displays *sin(x)* from $x = 0.0$ to $x = 1.5$. The expression is evaluated at $x = 0.0$, 0.5, 1.0 and 1.5. *g_replace* is set to `t`, so the resulting waveforms are given the lowest numbers already assigned to existing curves, and the existing curves are overwritten.

Related Functions

To evaluate an expression and assign the numbers specified in a list to the resulting waveforms, see the awvPlotExpression function on page-100.

To plot the Y values in a list against the X values in a list and display the resulting waveforms in a subwindow, see the awvPlotList function on page-106.

To plot the waveforms in a list in a subwindow, see the awvPlotWaveform function on page-112.

To evaluate an expression and add the resulting waveforms to a subwindow, see the awvAppendExpression function on page-29.

To plot the Y values in a list against the X values in a list and add the resulting waveforms to a subwindow, see the awvAppendList function on page-32.

To add the waveforms in a list to a subwindow, see the awvAppendWaveform function on page-35.

awvTableSignals

```
awvTableSignals(siglist@key [plotStyle style] [?graphModifier modifier])
```

Description

Displays a signal in the table window.

Arguments

`siglist`	List of signals to be plotted specified in the following format: (*list* (*list resultsDir1* (*list* (*list result1* (*list signal1 signal2* ...) ...) ...))) Remember to put quotation marks before and after each signal.
style	Plot destination. Valid values: `Table`, `Append`, `Replace`, `New Window`, `New Subwindow` Remember to put quotation marks before and after the style.
modifier	X axis of graph for rectangular graphs Valid values: `Magnitude`, `Phase`, `WPhase`, `Real`, `Imaginary`, `dB10`, `dB20`

Value Returned

`t`	If the signal is displayed in a table.
`nil`	Otherwise.

Example

```
awvTableSignals((("./ampsim.raw" (("ac-ac" ("net10"))))) "plotStyle" "Append")
```

awvUpdateAllWindows

```
awvUpdateAllWindows()
     => t | nil
```

Description

Updates the display of all the Waveform Windows.

Arguments

None.

Value Returned

`t`	Returns `t` when the Waveform Windows are updated.
`nil`	Returns `nil` if there are no Waveform Windows open.

Related Function

To update the display of only those subwindows whose update statuses are turned *on*, see the awvUpdateWindow function on page-171.

awvUpdateWindow

```
awvUpdateWindow( w_windowId )
     => t | nil
```

Description

Updates the display of all subwindows whose update-statuses are turned *on*.

Arguments

w_windowId	Waveform window ID.

Value Returned

`t`	Returns `t` when the subwindows are updated.
`nil`	Returns `nil` if the specified Waveform window does not exist.

Example

```
awvUpdateWindow( window(4) )
=> t
```

Updates all the subwindows for the Waveform window.

Related Functions

To turn the update status of a subwindow *on* or *off*, see the awvSetUpdateStatus function on page-153.

To return the list of subwindows (in a particular Waveform window) whose display and update statuses are on, see the awvGetOnSubwindowList function on page-67.

To update the display of all Waveform Windows, see the awvUpdateAllWindows function on page-170.

awvZoomFit

```
awvZoomFit(
     w_windowId
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Returns the traces to its actual size to fit in the window.

Arguments

w_windowId	Waveform window ID.

Value Returned

`t`	Returns `t` when the zoom fit operation is successful.
`nil`	Returns `nil` if the zoom fit operation fails.

Example

```
awvZoomFit(awvGetCurrentWindow())
awvZoomFit(awvGetCurrentWindow() ?subwindow 2)
```

awvZoomGraphX

```
awvZoomGraphX(
     w_windowId
     l_minMax
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Zooms in or out the graph according to the specified X-axis (dependent axis) coordinates.

Arguments

w_windowId	Waveform window ID.
l_minMax	List of two numbers in waveform coordinates that describe the minimum and maximum independent axis values for zoom. The first number is the minimum and the second is the maximum. The following notations are supported: list(2e-7 2.4e-7) 2e-7:2.4e-7 200n:240n list(200n 240n) list("200ns" "240ns")
x_subwindow	Number for the subwindow. Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the zoom in or out operation is successful.
`nil`	Returns `nil` if there is an error.

Example

```
awvZoomGraphX(awvGetCurrentWindow() 200n:240n ?subwindow 2)
awvZoomGraphX(awvGetCurrentWindow() list("20ns" "45ns") ?subwindow
awvGetCurrentSubWindow(awvGetCurrentWindow()))
```

awvZoomGraphY

```
awvZoomGraph(
     w_windowId
     l_minMax
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Zooms in or out the graph according to the specified Y-axis (independent axis) coordinates.

Arguments

w_windowId	Waveform window ID.
l_minMax	List of two numbers in waveform coordinates that describe the minimum and maximum dependent axis values for zoom. The first number is the minimum and the second is the maximum. The following notations are supported: list(2e-7 2.4e-7) 2e-7:2.4e-7 200n:240n list(200n 240n) list("200ns" "240ns")
x_subwindow	Number for the subwindow. Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the zoom in or out operation is successful.
`nil`	Returns `nil` if there is an error.

Example

```
awvZoomGraphY(awvGetCurrentWindow() 0:3 ?subwindow 2)
awvZoomGraphX(awvGetCurrentWindow() list("1mv" "4.5mv") ?subwindow
awvGetCurrentSubWindow(awvGetCurrentWindow()
```

awvZoomGraphXY

```
awvZoomGraphXY(
     w_windowId
     l_xminMax
     l_yminMax
     [?stripNumber  x_stripNumber]
     [?subwindow    x_subwindow]
     )
     => t | nil
```

Description

Zooms in or out the graph according to the specified X- and Y-axis coordinates.

Arguments

w_windowId	Waveform window ID.
l_xminMax	List of two numbers representing waveform X-axis coordinates that describe the limits for the zoom. The first number is the minimum and the second is the maximum.
l_yminMax	List of two numbers representing waveform Y-axis coordinates that describe the limits for the zoom. The first number is the minimum and the second is the maximum.
x_stripNumber	Specifies the strip to be zoomed. Valid Values: Integer value greater than or equal to 1
x_subwindow	Number for the subwindow. Default Value: Current subwindow

Value Returned

`t`	Returns `t` when the zoom in or out operation is successful.
`nil`	Returns `nil` if there is an error.

Example

```
awvZoomGraphXY(awvGetCurrentWindow() 0:3 nil ?subwindow 2)
```

```
awvZoomGraphXY(awvGetCurrentWindow() 200n:240n list("1mv" "4.5mv") ?subwindow
awvGetCurrentSubWindow(awvGetCurrentWindow())
?stripNumber 2)
```

awvGetSubwindowTitle

```
awvGetSubwindowTitle(
     w_windowId
     [?subwindow w_subwindow]
     )
     => t_subwindow/nil
```

Description

Returns the title of the subwindow.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
w_subwindow	Subwindow ID.

Value Returned

t_subwindow	Returns the title of the specified subwindow.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwindow = awvGetCurrentSubwindow(window)
awvDisplaySubwindowTitle(window "My_subWindow" ?subwindow subwindow)
awvGetSubwindowTitle(window ?subwindow subwindow)=> "My_subWindow"
```

This example returns `My_subWindow` as the title of the current subwindow.

awvGetWindowTitle

```
awvGetWindowTitle(
     w_windowId
     )
     => t_window/nil
```

Description

Returns the title of the waveform window.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.

Value Returned

t_window	Returns the title of the specified window.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
awvDisplayTitle(window "My_Window" )
awvGetWindowTitle(window)
```

This example returns `My_Window` as the title of current waveform window.

awvGetXAxisMajorDivisions

```
awvGetXAxisMajorDivisions(
     w_windowId
     [?subwindow w_subwindow]
     )
     => x_majdivs/nil
```

Description

Returns the number of major divisions that are set on the X-axis of a given graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
w_subwindow	Subwindow ID.

Value Returned

x_majdivs	Returns the number of major divisions on X-axis for the specified graph.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwindow = awvGetCurrentSubwindow(window)
awvGetXAxisMajorDivisions(window ?subwindow subwin)
```

This example returns the number of major divisions for the current subwindow.

awvGetXAxisMinorDivisions

```
awvGetXAxisMinorDivisions(
     w_windowId
     [?subwindow w_subwindow]
     )
     => x_mindivs/nil
```

Description

Returns the number of minor divisions on the X-axis of a given graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
w_subwindow	Subwindow ID.

Value Returned

x_majdivs	Returns the number of major divisions on X-axis for the specified graph.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwindow = awvGetCurrentSubwindow(window)
awvGetXAxisMinorDivisions(window ?subwindow subwin)
```

This example returns the number of minor divisions for the current subwindow.

awvGetXAxisStepValue

```
awvGetXAxisStepValue(
     w_windowId
     [?subwindow w_subwindow]
     )
     => x_stepSize/nil
```

Description

Returns the step size value for the X-axis for a given graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
w_subwindow	Subwindow ID.

Value Returned

x_stepSize	Returns the step size on X-axis for the specified graph.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwindow = awvGetCurrentSubwindow(window)
awvGetXAxisStepValue(window ?subwindow subwin)
```

This example returns the X-axis step size value for the current subwindow.

awvGetXAxisUseStepValue

```
awvGetXAxisUseStepValue(
     w_windowId
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Determines whether the X-axis scale uses the step value.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the step value has been used.
`nil`	Returns `nil` if there is an error.

Example

```
awvGetXAxisUseStepValue(window ?subwindow subwin)
```

awvSetXAxisMajorDivisions

```
awvSetXAxisMajorDivisions
     w_windowId
     x_numMajDiv
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Sets the number of major divisions on the X-axis for the specified graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_numMajDiv	Number of major divisions to be set.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the function runs successfully.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
numMajDiv = 8
awvSetXAxisMajorDivisions(window 8 ?subwindow subwin)
```

This function sets 8 number of major divisions for the current subwindow.

awvSetXAxisMinorDivisions

```
awvSetXAxisMinorDivisions(
     w_windowId
     x_numMinDiv
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Sets the number of major divisions on the X-axis for the specified graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_numMinDiv	Number of minor divisions to be set.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the function runs successfully.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwindow = awvGetCurrentSubwindow(window)
numMajDiv = 8
awvSetXAxisMinorDivisions(window 5 ?subwindow subwin)
```

This function sets 5 number of minor divisions for the current subwindow.

awvSetXAxisStepValue

```
awvSetXAxisStepValue(
     w_windowId
     x_stepValue
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Sets the step value for a X-axis for the specified graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_stepValue	Step size to be set. It must be a valid number.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the function runs successfully.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
xAxisStepSize = 25n
subwin = awvGetCurrentSubwindow(win)
awvSetXAxisStepValue(win xAxisStepSize ?subwindow subwin)
```

This example sets the step value as `25n` for the X-axis in the current subwindow.

awvSetXAxisUseStepValue

```
awvSetXAxisUseStepValue(
     w_windowId
     g_value
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Specifies whether the step size value is to be used on the X-axis scale.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
g_value	Specifies whether step size value is used or not. Valid Values: `t` or `nil`.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the function runs successfully.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvSetXAxisStepValue(window 25n ?subwindow subwin)
```

The above example sets the X-axis step size to `25n`.

```
awvSetXAxisUseStepValue(window t ?subwindow subwin)
```

The above example sets the specified step value for the X-axis pertaining to specified graph.

```
awvSetXAxisUseStepValue(window nil ?subwindow subwin
```

The above example sets the default step size computed for the X-axis pertaining to specified graph.

awvGetYAxisMajorDivisions

```
awvGetYAxisMajorDivisions(
     w_windowId
     x_yNumber
     ?subwindow w_subwindow
     )
     => x_majdivs/nil
```

Description

Returns the number of major divisions that are set on the Y-axis of a given graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_yNumber	Specifies the Y-axis whose minor divisions to be returned. Valid Values: 1 through 4
w_subwindow	Subwindow ID.

Value Returned

x_majdivs	Returns the number of major divisions on Y-axis for the specified graph.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvGetYAxisMajorDivisions(window 1 ?subwindow subwin ?stripNumber 1)
```

This example returns the number of major divisions for the first Y-axis in the current subwindow.

awvGetYAxisMinorDivisions

```
awvGetYAxisMinorDivisions(
     w_windowId
     x_yNumber
     [?stripNumber x_stripNumber]
     [?subwindow w_subwindow]
     )
     => x_mindivs/nil
```

Description

Returns the number of minor divisions on the specified Y-axis of a given graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_yNumber	Specifies the Y-axis whose minor divisions to be returned. Valid Values: 1 through 4
x_stripNumber	Specifies the strip number for which minor divisions are to be returned.
w_subwindow	Subwindow ID.

Value Returned

x_majdivs	Returns the number of major divisions on Y-axis for the specified graph.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvGetYAxisMinorDivisions(window 2 ?subwindow subwin ?stripNumber 1)
```

This example returns the number of minor divisions for the Y-axis number 2 in the current subwindow.

awvGetYAxisStepValue

```
awvGetYAxisStepValue(
     w_windowId
     x_yNumber
     [?stripNumber x_stripNumber]
     [?subwindow w_subwindow]
     )
     => x_stepSize/nil
```

Description

Returns the step size value for the specified Y-axis for a given graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_yNumber	Specifies the Y-axis whose for which the step size is to be returned. Valid Values: 1 through 4
x_stripNumber	Specifies the strip number for which step size is to be returned.
w_subwindow	Subwindow ID.

Value Returned

x_stepSize	Returns the step size on Y-axis for the specified graph.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvGetYAxisStepValue(window 2 ?subwindow subwin ?stripNumber 4))
```

This example returns the Y-axis step size value for the strip number 4 of the current subwindow.

awvGetYAxisUseStepValue

```
awvGetYAxisUseStepValue(
     w_windowId
     x_yNumber
     [?stripNumber x_stripNumber]
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Determines whether the Y-axis scale use the step value for the specified strip in a given graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_yNumber	Specifies the Y-axis for which step size is to be set. Valid Values: 1 through 4
x_stripNumber	Specifies the strip number for which the step size is to be set.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the step value has been used.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvGetYAxisUseStepValue(window 2 ?subwindow subwin ?stripNumber 3)
```

This example returns true if step size has been used for strip number 3 in the current subwindow.

awvSetYAxisMajorDivisions

```
awvSetYAxisMajorDivisions(
     w_windowId
     x_yNumber
     x_numMajDiv
     [?stripNumber x_stripNumber]
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Sets the number of major divisions on the X-axis for the specified graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_yNumber	Specifies the Y-axis for which major divisions are to be set. Valid Values: 1 through 4
x_numMajDiv	Number of major divisions to be set.
x_stripNumber	Specifies the strip number for which major divisions are to be set.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the function runs successfully.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvSetYAxisMajorDivisions(window 1 6 ?subwindow subwin ?stripNumber 6)
```

This function sets 6 number of major divisions for the strip number 6 in the current subwindow.

awvSetYAxisMinorDivisions

```
awvSetYAxisMinorDivisions(
     w_windowId
     x_yNumber
     x_numMajDiv
     [?stripNumber x_stripNumber]
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Sets the number of major divisions on the specified Y-axis for the specified graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_yNumber	Specifies Y-axis for which minor divisions are to be set. Valid Values: 1 through 4
x_numMajDiv	Number of minor divisions to be set.
x_stripNumber	Specifies the strip number for which minor divisions are to be set.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the function runs successfully.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvSetYAxisMinorDivisions(window 1 6 ?subwindow subwin ?stripNumber 1))
```

This function sets 6 number of minor divisions for the strip number 1 in the current subwindow.

awvSetYAxisStepValue

```
awvSetYAxisStepValue(
     w_windowId
     x_yNumber
     x_stepValue
     [?stripNumber x_stripNumber]
     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Sets the step value for the specified Y-axis for the specified graph.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_yNumber	Specifies Y-axis for which the step size is to be set. Valid Values: 1 through 4
x_stepValue	Step size to be set. It must be a valid number.
x_stripNumber	Specifies the strip number for which the step size is to be set.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the function runs successfully.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvSetYAxisStepValue(window 1 0.25)
```

This example sets the step value as `0.25n` for the Y-axis in the current subwindow.

awvSetYAxisUseStepValue

```
awvSetYAxisUseStepValue(
     w_windowId
     x_yNumber
     g_value
     [?stripNumber x_stripNumber]

     [?subwindow w_subwindow]
     )
     => t/nil
```

Description

Specifies whether the step size value is to be used on the Y-axis scale.

Arguments

w_windowId	Waveform window ID. You can specify window ID as window(1) or specify the current window by using the `awvGetCurrentWindow` function.
x_yNumber	Specifies Y-axis for which the step size is to be used. Valid Values: 1 through 4
g_value	Specifies whether step size to be used or not. Valid Values: `t` or `nil`.
x_stripNumber	Specifies the strip number for which the step size is to be set.
w_subwindow	Subwindow ID.

Value Returned

`t`	Returns `t` if the function runs successfully.
`nil`	Returns `nil` if there is an error.

Example

```
window = awvGetCurrentWindow()
subwin = awvGetCurrentSubwindow(window)
awvGetYAxisStepValue(window 1 ?subwindow subwin ?stripNumber 1)
```

3

Callback Functions

This chapter describes the following callback functions in detail:

- awviEditMenuCB
- awviMakeActiveMenuCB
- awviPLoadMenuCB
- awviPSaveMenuCB
- awviPUpdateMenuCB
- awviShowOutputMenuCB

awviEditMenuCB

```
awviEditMenuCB() => t | nil
```

Description

Callback for the *Expressions->Edit* menu in the Results Display window. It brings up the Expressions Edit form.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awviEditMenuCB() => t
```

awviMakeActiveMenuCB

```
awviMakeActiveMenuCB => t | nil
```

Description

Callback for the menu *Window->Make Active* in the Results Display window. It makes current window the active print window.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awviMakeActiveMenuCB() => t
```

awviPLoadMenuCB

```
awviPLoadMenuCB => t | nil
```

Description

Callback for the menu *Window->Load State* in the Results Display window. It loads a saved state of print window.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awviPLoadMenuCB() => t
```

awviPSaveMenuCB

```
awviPSaveMenuCB => t | nil
```

Description

Callback for the menu *Window->Save State* in the Results Display window. It saves the current state of print window.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awviPSaveMenuCB() =>t
```

awviPUpdateMenuCB

```
awviPUpdateMenuCB => t | nil
```

Description

Callback for the *Window->Update Results* menu in the Results Display window. It updates the data using the current window setup.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awviPUpdateMenuCB() => t
```

awviShowOutputMenuCB

```
awviShowOutputMenuCB => t | nil
```

Description

Callback for the menu *Info->Show Output* in the Results Display window. It displays the expressions printed in the print window.

Arguments

None

Value Returned

t	Returns `t` if the command was successful.
nil	Otherwise, returns `nil`.

Example

```
awviShowOutputMenuCB() => t
```

appendWaves

```
appendWaves( o_wave1 o_wave2 ... [o_waveN] ) => o_waveform | nil
```

Description

Appends a series of input waveforms in the X vector direction into a single output waveform.It is the users responsibility to insure that the input waveforms X vectors are in the proper sequence.

Arguments

o_wave1	The first input waveform.
o_wave2	The second input waveform.
o_waveN	The optional Nth input waveform.

Values Returned

`o_waveform`	Creates an input waveform.
`nil`	Otherwise, returns `nil`

Example

```
appendWaves( VT("/out" "path_to_reference_directory")
VT("/out")) => o_waveform
```

waveVsWave

```
waveVsWave(?y o_wavey ?x o_wavex) => o_waveform | nil
```

Description

Creates an output waveform where its Y vector values are the Y vector values of the ?y input waveform and its X vector values are the Yvector values of the ?x input waveform.

Arguments

o_wavey	The Y vector values of this input waveform will be used for the Y vector values of the output waveform.
o_wavex	The Y vector values of this input waveform will be used for the X vector values of the output waveform.

Values Returned

`o_waveform`	Creates an output waveform.
`nil`	Otherwise, returns `nil`

Example

```
waveVsWave( VT("/out" "path_to_reference_directory")
VT("/out")) => o_waveform
```

4

Calculator Functions

This chapter lists the SKILL functions for Waveform Calculator.

armSetCalc

```
armSetCalc( s_name g_value )
     => g_value
```

Description

Sets the calculator resource of the specified property to the specified value.

Arguments

s_value	Name of any valid expression.
g_value	Value of a valid browser resource.

Value Returned

g_value	Value of the browser resource.

Example

```
armSetCalc( 'numStack 10 )
=> g_value
```

Sets the number of stacks to be displayed in the calculator to 10.

calCalculatorFormCB

```
calCalculatorFormCB() => t | nil
```

Description

Opens a new calculator window, if not already open. If the window is open, it activates the window and brings it in the focus.

Arguments

None

Value Returned

`t`	Returns `t` if the command was successful.
`nil`	Otherwise, returns nil

Example

```
calCalculatorFormCB()
```

calCalcInput

```
calCalcInput(isl_keyword @optional t_expression)
```

Description

The calCalcInput function manipulates the buffer and stack contents and enters arbitrary expressions into the buffer. The syntax of this function and a brief description of the arguments and valid keywords is given below.

Arguments

isl_keyword	Integer, symbol, or list of symbols indicating the function keywords to be performed on the buffer or stack.

Table of Keywords

The valid keywords you can pass to the `calCalcInput` function are those functions seen on the keypad and are summarized as follows.

Keyword	Function
0-9	Enter numerals in the buffer
eex	Puts the calculator in exponential mode and enters an *e* in the buffer
point	Enters a decimal point in the buffer
pi, twoPi, sqrt2, charge, degPerRad, boltzmann, epp0	Enter the constant names in the buffer
clear	Clears the buffer
lastx	Recalls the contents of the lastx register to the buffer
clst	Clears the buffer and all the stack registers
up, dwn, xchxy (x<>y on the keyboard)	Perform the up, down, and x-exchange-y operations on the buffer and stack contents
enter	Places a copy of the current buffer expression in the stack
append (app on the keyboard)	Appends the first stack element to the contents of the buffer

Keyword	Function
multiply, divide, add, subtract	Perform the specified operation on the buffer and the first stack element
chs	Changes the sign of the buffer expression or exponent
power	Raises 10 to the power of the buffer expression
square	Squares the buffer expression
exp	Calculates *e* to the power of the buffer expression
abs	Calculates the absolute value of the buffer expression
int	Truncates the integer portion of the buffer expression
db10	Calculates the dB magnitude of the buffer contents for a power expression
db20	Calculates the dB magnitude of the buffer contents for a voltage or current expression
sqrt	Calculates the square root of the buffer expression
mag	Calculates the magnitude of an AC buffer expression
phase	Calculates the phase of an AC buffer expression
imag	Calculates the imaginary part of an AC buffer expression
real	Calculates the real part of an AC buffer expression
log10	Calculates the base-10 logarithm of the buffer expression
ln	Calculates the base-*e* (natural) logarithm of the buffer expression
ytox	Raises the contents of the first stack element to the power of the contents of the buffer
expression	Specifies the expression string to place in the buffer (Provides an implicit enter for the current buffer expression.)
t_expression	Specifies the expression string to place in the buffer (Argument used only with the expression keyword)

calCreateSpecialFunction

```
calCreateSpecialFunction( ?formSym s_formSym ?formInitProc s_formInitProc
     ?formTitle t_formTitle ?formCallback t_formCallback ?envGetVal
     t_envGetVal) => t | nil
```

Description

Encapsulates the initialization and display of forms required for a special function.

Arguments

s_formSym	Form symbol for the form.
s_formInitProc	Symbol describing the procedure that creates the form entries and form.
t_formTitle	String describing the special function. This title is placed on the window border of the form.
t_formCallback	Callback for the form that processes the form fields and enters the expression into the calculator buffer.
t_envGetVal	Gives the name of the partition for the "calculator" tool. The function is called using this variable to retrieve the value for all the form fields (if set) from the default environment.

calCreateSpecialFunctionsForm

```
calCreateSpecialFunctionsForm( s_formSym l_fieldList ) => t
```

Description

Registers the form and calls an hiCreateForm to create the form. The form title and callback are specified through the call to calCreateSpecialFunction.

The form is also created with g_unmapAfterCB set to *t* (see hiCreateForm).

Arguments

s_formSym	Form symbol for the form.
l_fieldList	List of fields in the form.

calGetBuffer

```
calGetBuffer() => t_buffer
```

Description

Gets the expression constructed in the calculator buffer.

Arguments

None.

Value Returned

`t_buffer`	Returns the expression stored in the calculator buffer if the command was successful. It throws an error if the calculator form has not been initialized.

Example

```
calGetBuffer()
```

calSetBuffer

```
calSetBuffer(t_buffer) => nil
```

Description

Sets the contents of the calculator buffer.

Arguments

`t_buffer`	Expression to be stored in the calculator buffer.

Value Returned

```
nil
```

Example

```
calSetBuffer("VT('out')")
```

Sets the contents of the calculator buffer to `VT('out')`.

calSetCurrentTest

```
calSetCurrentTest(testName) => t | nil
```

Description

Informs the calculator of the current test. When an access function (such as vt) is used from the calculator, it will display the schematic associated with the current test.

The tests are displayed in a drop-down in the calculator, which allows you to change the current test.

Arguments

testName	Name of current test.

Value Returned

t	If the current test is displayed
nil	Otherwise.

Example

```
calSetCurrentTest(opamplib:amptest:1)
```

caliModeToggle

```
caliModeToggle() => t | nil
```

Description

Toggles the algebraic or RPN specific buttons on the calculator form based on the current value of the ‘calculator mode’.

Arguments

None.

Value Returned

`t`	Returns `t` if the command was successful.
`nil`	Otherwise, returns nil

Example

```
caliModeToggle()
```

caliRestoreDefaultWindowSize

```
caliRestoreDefaultWindowSize() => t | nil
```

Description

Restores the original size of the calculator form window while maintaining the same upper left coordinate of the current window position.

Arguments

None.

Value Returned

`t`	Returns `t` if the command was successful.
`nil`	Otherwise, returns nil

Example

```
caliRestoreDefaultWindowSize()
```

calRegisterSpecialFunction

```
calRegisterSpecialFunction( l_sfinfo ) => l_sfinfo | nil
```

Description

Registers the specified special function information if it is not already registered.

Arguments

l_sfinfo	Name of the form list (t_sfname s_sfcallback).
t_sfname	Name that appears in the Special Functions menu.
s_sfcallback	Symbol describing the name of the callback procedure for this special function.

calSpecialFunctionInput

```
calSpecialFunctionInput( t_sfname l_fields ) => t_expression | nil
```

Description

Checks the buffer and stack and processes the arguments defined under `l_fields` into the buffer expression.

Arguments

t_sfname	Name of the function.
l_fields	Ordered list of form field symbols that compose the special function argument list. If the special symbol 'STACK is encountered in the list of fields, the top stack expression is popped into the special function argument list. Currently, all argument fields specified must have associated values or an error message is generated. If the field type is cyclic or radio, double quotes are added around the field value in the special function argument list.

Value Returned

`t_expression`	Returns the expression in the Buffer with the specified argument values.
`nil`	Returns nil if there is an error.

Example1

```
calSpecialFunctionInput('test '(from STACK to))
```

This example defines a new function `test` with the following arguments:

- *from*—Obtains the value that you have specified in the `from` argument.
- `STACK`—Pops out the expression from the top of the stack.
- *to*—Obtains the value that you have specified in the `to` argument.

Suppose *from*=`10`, *to*=`20`, Top of stack=`v("out" ?result "tran")`

And, Buffer contains the following value:

```
Buffer=v("in" ?result "tran")
```

This function outputs the following expression in the Buffer:

```
test(Buffer from, STACK to)
```

```
test(v("in" ?result "tran") 10 v("out" ?result "tran")20)
```

Example2

```
calSpecialFunctionInput( 'test nil )
```

This example defines a new function `test` with no arguments. When you select this function, the function will be directly applied on the expression in the Buffer.

expr

```
expr( var expr l_values ) => o_waveform | nil
```

Description

Evaluates the expression by setting each of the values in the ‘l_values’ list to the ‘var’ variable in the ‘expr’.

A waveform object is created with ‘l_values’ forming the x-vector and the evaluated ‘expr’ forming the y-vector.

Arguments

var	variable
expr	expression containing the variable
l_values	list of x-values vectors

Value Returned

o_waveform	Returns a waveform object.
nil	Otherwise, returns nil

Example

```
expr( x  (x*100)  '(10 20 30))
```

This implies that a waveform object with x-vector values as (10 20 30) and y-vector values as (1000 2000 3000).

famEval

```
famEval( l_expression [[?va1 s_value1]...] ) => o_result
```

Description

Evaluates the expression with the sweep variables set as specified. A waveform or a number is returned.

Arguments

l_expression	expression
va1	sweep variable name
s_value1	value of the sweep variable

Value Returned

o_result	Returns a waveform object or a number depending upon the inputs.

Example

```
famEval( VT("out") ?len "100u" )
```

vvDisplayCalculator

```
vvDisplayCalculator [(expr)] => t | nil
```

Description

Invokes the calculator within a window. If an expression is specified, the expression is displayed in the buffer.

Arguments

expr	Expression to be displayed in the calculator buffer. This is an optional argument. Remember to put quotation marks before and after the expression.

Value Returned

`t`	If the calculator is invoked.
`nil`	Otherwise.

Example

```
vvDisplayCalculator(vt(\/net10\))
```

5

Results Browser Functions

This chapter describes the SKILL functions that apply to the Results Browser in the Virtuoso Visualization and Analysis XL tool.

rdbLoadResults

```
rdbLoadResults (sessionName resultsDir) => t | nil
```

Description

Loads the simulation results located at *resultsDir* into the browser associated with the specified *sessionName*.

Arguments

sessionName	Name of the session in which the results are to be loaded in the browser. Remember to put quotation marks before and after the session name.
resultsDir	Path to the location of the simulation results. Remember to put quotation marks before and after the path name.

Value Returned

t	If the simulation results are loaded.
nil	Otherwise.

Example

```
rdbLoadResults( "unbound" "./opamplib/ampTest/adexl/results/data/Interactive.267/
1/opamplib:ampTest:1/psf")
```

rdbReloadResults

```
rdbReloadResults (sessionName resultsDir) => t | nil
```

Description

Re-loads the simulation results located at *resultsDir* into the browser associated with the specified *sessionName*. It is desirable to reload results during the successive simulation runs.

Arguments

sessionName	ADE-L/XL sessionName or unbound if running the tool in standalone mode and the Results Browser window is open.
resultsDir	Path to the location of the simulation results. Remember to put quotation marks before and after the path name.

Value Returned

t	If the simulation results are re-loaded.
nil	Otherwise.

Example

```
rdbReloadResults( "unbound" list("/usr1/export/amptest/simulation/ampTest/
spectre/config/psf"))
```

rdbUnloadResults

```
rdbUnloadResults (sessionName resultsDir) => t | nil
```

Description

Unloads the simulation results located at *resultsDir* from the browser associated with the specified *sessionName*. It is desirable to unload results during the successive simulation runs to reduce resource consumption and remove clutter from the browser.

Arguments

sessionName	ADE-L/XL sessionName or unbound if running the tool in standalone mode and the Results Browser window is open.
resultsDir	Path to the location of the simulation results. Remember to put quotation marks before and after the path name.

Value Returned

t	If the simulation results are unloaded.
nil	Otherwise.

Example

```
rdbUnloadResults( "unbound" list("/usr1/export/amptest/simulation/ampTest/
spectre/config/psf"))
```

rdbSetCurrentDirectory

```
rdbSetCurrentDirectory (sessionName path) => t | nil
```

Description

The Results Browser associated with the specified *sessionName* navigates to the directory specified by *path*.

Arguments

sessionName	ADE-L/XL `sessionName` or `unbound` if running the tool in standalone mode and the Results Browser window is open.
path	Path to the simulation results hierarchy to be displayed. Remember to put quotation marks before and after the path name.

Value Returned

`t`	If the simulation results are displayed.
`nil`	Otherwise.

Example

```
rdbSetCurrentDirectory ("unbound" "./simulation/opampTest/spectre/schematic/psf/
tran/V11")
```

rdbWriteToFormat

```
rdbWriteToFormat (sessionName path format signals) => t | nil
```

Description

Outputs the specified signals from the browser associated with the specified *sessionName*.

Arguments

sessionName	ADE-L/XL `sessionName` or `unbound` if running the tool in standalone mode and the Results Browser window is open.
path	Path to the signals that are to be output. Remember to put quotation marks before and after the path name.
format	Format in which the signals are to be output. Valid values: `CSV` (Comma-Separated text), `VCSV` (Virtuoso Comma-Separated text), `PSF`, `SST2`, `Matlab`, `Spectre` Remember to put quotation marks before and after the format.
signals	List of signals to be output specified in the following format: `(`*list* `(`*list resultsDir1* `(`*list* `(`*list result1* `(`*list signal1 signal2* ...`)` ...`)` ...`)))`. If you do not specify a signal name in this argument, the command outputs all the signals that are selected in the Results Browser.

Value Returned

`t`	If the signals are displayed.
`nil`	Otherwise.

Example

```
rdbWriteToFormat( "unbound" "/tmp/output.vcsv" "VCSV" '(("./simulation/opampTest/
spectre/schematic/psf" (("tran" ("V11.p"))))))
```

rdbShowDialog

```
rdbShowDialog (sessionName path format signals) => t | nil
```

Description

Display and hides dialog boxes associated with the browser.

Arguments

sessionName	ADE-L/XL `sessionName` or `unbound` if running the tool in standalone mode and the Results Browser window is open.
path	Path to the signals that are to be output. Remember to put quotation marks before and after the path name.
format	Format in which the signals are to be output. Valid values: `CSV` (Comma-Separated text), `VCSV` (Virtuoso Comma-Separated text), `PSF`, `SST2`, `Matlab`, `Spectre` Remember to put quotation marks before and after the format.
signals	

Value Returned

`t`	If the specified dialog boxes are displayed or hidden.
`nil`	Otherwise.

Example

```
rdbShowDialog(unbound browser findResults show ((pathList ./ampsim.raw)))
```

vvDisplayBrowser

```
vvDisplayBrowser => t | nil
```

Description

Invokes the Results Browser within a window.

Value Returned

`t`	If the browser is invoked.
`nil`	Otherwise.

vivaInitBindkeys

`vivaInitBindkeys()`

Description

Initializes the Virtuoso Visualization and Analysis XL bindkeys called from `viva.ini` context initialization file

Value Returned

None.