Converting a String into a Reference Using $ on page IV-62.

SVAR_Exists

SVAR_Exists(*name*)

The SVAR_Exists function returns 1 if the specified SVAR reference is valid or 0 if not. It can be used only in user-defined functions.

For example, in a user function you can test if a global string variable exists like this:

```
SVAR /Z str1 = gStr1       // /Z prevents debugger from flagging bad SVAR
if (!SVAR_Exists(str1))    // No such global string variable?
   String/G gStr1 = ""     // Create and initialize it
endif
```

See Also

WaveExists, **NVAR_Exists**, and **Accessing Global Variables and Waves** on page IV-65.

switch-case-endswitch

switch(<***numeric expression***>**)**
   case <***literal***><***constant***>**:**
      <***code***>
      [**break**]
   [**default:**
      <***code***>]
endswitch

A switch-case-endswitch statement evaluates the numeric expression and rounds the result to the nearest integer. If a case label matches *numerical expression*, then execution proceeds with *code* following the matching case label. When no cases match, execution continues at the default label, if present, or otherwise the switch exits with no action taken. Although the break statement is optional, in almost all case statements it is required for the switch to work correctly.

Literal numbers used as case labels are required by the compiler to be integers. Numeric constants used as case labels are rounded by the compiler to the nearest integers.

See Also

Switch Statements on page IV-43, **default** and **break** for more usage details.

t

t

The t function returns the T value for the current chunk of the destination wave when used in a multidimensional wave assignment statement. T is the scaled chunk index while **s** is the chunk index itself.

Details

Unlike **x**, outside of a wave assignment statement, t does not act like a normal variable.

See Also

Waveform Arithmetic and Assignments on page II-74.

For other dimensions, the **p**, **q**, **r**, and **s** functions.

For scaled dimension indices, the **x**, **y**, and **z** functions.

TabControl

TabControl [**/Z**] ***ctrlName*** [***keyword = value*** [**,** ***keyword = value*** …]]

The TabControl operation creates tab panels for controls.

For information about the state or status of the control, use the **ControlInfo** operation.

Parameters

ctrlName is the name of the TabControl to be created or changed.

The following keyword=value parameters are supported:

align=*alignment*

Sets the alignment mode of the control. The alignment mode controls the interpretation of the *leftOrRight* parameter to the pos keyword. The align keyword was added in Igor Pro 8.00.

If *alignment*=0 (default), *leftOrRight* specifies the position of the left end of the control and the left end position remains fixed if the control size is changed.

If *alignment*=1, *leftOrRight* specifies the position of the right end of the control and the right end position remains fixed if the control size is changed.

appearance={*kind* [, *platform*]}

Sets the appearance of the control. *platform* is optional. Both parameters are names, not strings.

kind can be one of `default`, `native`, or `os9`.

platform can be one of `Mac`, `Win`, or `All`.

See **DefaultGUIControls Default Fonts and Sizes** for how enclosed controls are affected by native TabControl appearance.

See **Button** for more appearance details.

disable=*d*

Sets user editability of the control.

d=0: Normal.

d=1: Hide.

d=2: Draw in gray state; disable control action.

fColor=(*r*,*g*,*b*[,*a*])

Sets the initial color of the tab labels. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. Use of transparency is discouraged. The default is opaque black.

To further change the color of the tab labels text, use escape sequences in the text specified by the tabLabel keyword.

focusRing=*fr*

Enables or disables the drawing of a rectangle indicating keyboard focus:

fr=0: Focus rectangle will not be drawn.

fr=1: Focus rectangle will be drawn (default).

On Macintosh, regardless of this setting, the focus ring appears if you have enabled full keyboard access via the Shortcuts tab of the Keyboard system preferences.

font= "*fontName*"

Sets the font used for tabs, e.g., `font="Helvetica"`.

fsize= *s*

Sets the font size for tabs.

fstyle=*fs*

fs is a bitwise parameter with each bit controlling one aspect of the font style as follows:

Bit 0: Bold

Bit 1: Italic

Bit 2: Underline

Bit 4: Strikethrough

See **Setting Bit Parameters** on page IV-12 for details about bit settings.

help={*helpStr*}

Sets the help for the control.

helpStr is limited to 1970 bytes (255 in Igor Pro 8 and before).

You can insert a line break by putting "\r" in a quoted string.

labelBack=(*r*,*g*,*b*[,*a*]) or 0	Sets fill color for current tab and the interior. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. If not set, then interior is transparent and the current tab is filled with the window background, though this style is platform-dependent. If you use an opaque fill color, drawing objects will not be seen because they will be covered up.
noproc	Specifies that no function is to run when clicking a tab.
pos={*leftOrRight*,*top*}	Sets the position in **Control Panel Units** of the top/left corner of the control if its alignment mode is 0 or the top/right corner of the control if its alignment mode is 1. See the align keyword above for details.
pos+={*dx*,*dy*}	Offsets the position of the control in **Control Panel Units**.
proc=*procName*	Specifies the function to run when the tab is pressed. Your function must hide and show other controls as desired. The TabControl does not do this automatically.
size={*width*,*height*}	Sets TabControl size in **Control Panel Units**.
tabLabel(*n*)=*lbl*	Sets *n*th tab label to *lbl*. Set the label of the last tab to `""` to remove the last tab. Using escape codes you can change the font, size, style, and color of the label. See **Annotation Escape Codes** on page III-53 or details.
userdata(*UDName*)=*UDStr*	Sets the unnamed user data to *UDStr*. Use the optional (*UDName*) to specify a named user data to create.
userdata(*UDName*)+=*UDStr*	Appends *UDStr* to the current unnamed user data. Use the optional (*UDName*) to append to the named *UDStr*.
value=*v*	Sets current tab number. Tabs count from 0.
win=*winName*	Specifies which window or subwindow contains the named control. If not given, then the top-most graph or panel window or subwindow is assumed. When identifying a subwindow with *winName*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.

Flags

/Z	No error reporting.

Tab Control Action Procedure

The action procedure for a TabControl takes a predefined `WMTabControlAction` structure as a parameter to the function:

```
Function ActionProcName(TC_Struct) : TabControl
   STRUCT WMTabControlAction &TC_Struct
   …
   return 0
End
```

The "`: TabControl`" designation tells Igor to include this procedure in the Procedure pop-up menu in the Tab Control dialog.

See **WMTabControlAction** for details on the WMTabControlAction structure.

Although the return value is not currently used, action procedures should always return zero.

When clicking a TabControl with the selector arrow, click in the title region. The control is not selected if you click in the body. This is to make it easier to select controls in the body rather than the TabControl itself.