ValDisplay

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

The ValDisplay operation creates or modifies the named control that displays a numeric value in the target window. The appearance of the control varies; see the **Examples** section.

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

Parameters

ctrlName is the name of the ValDisplay control 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 **Button** and **DefaultGUIControls** for more appearance details.

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

Sets the background color under the bar (if any). *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

barBackColor=0

Sets the background color under the bar to the default color, the standard document background color used on the current operating system, which is usually white.

barmisc={*lts*, *valwidth*}

Sets the "limits text size" and the size of the type showing the bar limits. If *lts* is zero, the bar limits are not displayed. Otherwise, *lts* must be between 5 and 100.

valwidth is the "value readout width". It claims the amount of horizontal space for the numeric part of the display.

If *valwidth* equals or exceeds the control width available to it, the numeric readout uses all the room, and prevents display of any bar.

If *valwidth* is zero, there is no numeric readout, and only the bar is displayed.

valwidth can range from zero to 4000, and it defaults to 1000 (which usually leaves no room for the display bar).

bodyWidth=*width*

Specifies an explicit size for the body (nontitle) portion of a ValDisplay control. By default (bodyWidth=0), the body portion is the amount left over from the specified control width after providing space for the current text of the title portion. If the font, font size, or text of the title changes, then the body portion may grow or shrink. If you supply a bodyWidth>0, then the body is fixed at the size you specify regardless of the body text. This makes it easier to keep a set of controls right aligned when experiments are transferred between Macintosh and Windows, or when the default font is changed.

disable=*d*

Sets user editability of the control.

d=0: Normal.

d=1: Hide.

d=2: Disable user input.
Does not change control appearance because it is read-only.

d=3: Hide and disable the control.
This is useful to disable a control that is also hidden because it is in a hidden tab.

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

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

To further change the color of the title text, use escape sequences as described for title=*titleStr*.

font="*fontName*"

Sets the font used to display the value of the variable, e.g., `font="Helvetica"`

format=*formatStr*

Sets the numeric format of the displayed value. The default format is "%g". For a description of *formatStr*, see the **printf** operation.

frame=*f*

Sets frame style:

f=0: Value is unframed.

f=1: Default; value is framed (same as *f*=3).

f=2: Simple box.

f=3: 3D sunken frame.

f=4: 3D raised frame.

f=5: Text well.

fsize=*s*

Sets the size of the type used to display the value in the numeric readout. The default is 12 points.

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.

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

Specifies the bar color when the value is greater than *base* in the limits keyword. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

labelBack=(*r*,*g*,*b*[,*a*]) or 0

Specifies the background fill color for labels. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is 0, which uses the window's background color.

limits={*low*,*high*,*base*}

Controls how the value is translated into a graphical representation when the display includes a bar (described fully in **Details**). Defaults are {0,0,0}, which aren't too useful.

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

Sets the color of the limits text, if any. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

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

Sets the background color under the limits text. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

limitsBackColor=0

Sets the background color under the limits text to the default color, the standard document background color used on the current operating system, which is usually white.

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

Specifies the bar color when the value is less than *base* in the limits keyword. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

mode=*m*

Specifies the type of LED display to use, if any.

m=0: Bar mode (default).

m=1: Oval LED.

m=2: Rectangular LED.

m=3: Bar mode with no fractional part.

m=4: Candy-stripe effect for the bar area to support indefinite-style progress windows. The value is taken to be the phase of the candy stripe. When using value= _NUM:n, n is taken as an increment value so you would normally just use 1. Uses the native platform appearance if the high and low colors are left as default. Note native formats may not fill vertical space. See **Progress Windows** on page IV-156 for an example.

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 display in **Control Panel Units**.

rename=*newName*

Gives the ValDisplay control a new name.

size={*width*,*height*}

Sets width and height of display in **Control Panel Units**. *width* can range from 10 to 200 units, *height* from 5 to 200 units. Default width is 50, default height is determined by the numeric readout font size.

title=*titleStr*

Sets title of display to the specified string expression. The title appears to the left of the display. If this title is too long, it won't leave enough room to display the bar or even the numeric readout! Defaults to "" (no title).

Using escape codes you can change the font, size, style, and color of the title. See **Annotation Escape Codes** on page III-53 or details.

value=*valExpr*

Displays the numeric expression *valExpr*. It is *not* a string.

As of version 6.1, you can use the syntax _NUM:num to specify a numeric value without using a dependency.

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

Sets the color of the value readout text, if any. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

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

Sets the background color under the value readout text. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

valueBackColor=0

Sets the background color under the value readout text to the default color, the standard document background color used on the current operating system, which is usually white.

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.

zeroColor=(*r*,*g*,*b*[,*a*]) Governs the LED color (in LED mode only). *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. Used in conjunction with the limits keyword such that zeroColor determines one endpoint color when base is between *low* and *high*, or LED color when the value is less than *low*.

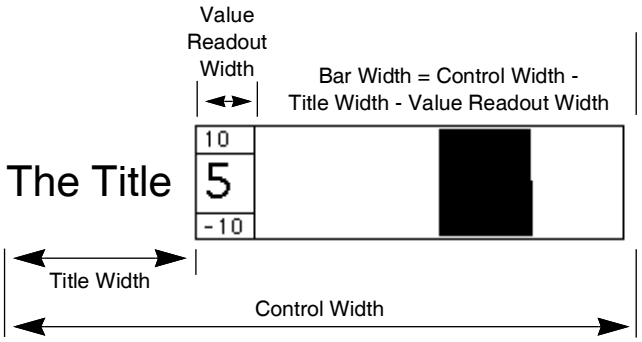
Flags

/Z No error reporting.

Details

The target window must be a graph or panel.

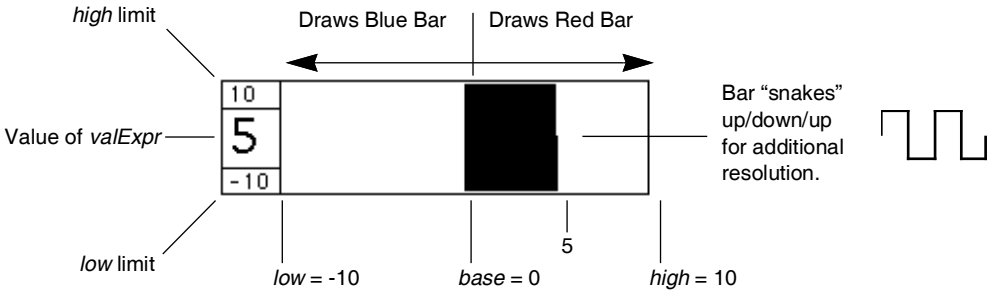
The appearance of the ValDisplay control depends primarily on the *width* and *valwidth* parameters and the width of the title. Space for the individual elements is allocated from left to right, with the title receiving first priority. If the control width hasn't all been used by the title, then the value display gets either *valwidth* **Control Panel Units** of room, or what is left. If the control width hasn't been used up, the bar is displayed in the remaining control width:



If you use the bodyWidth keyword, the value readout width and bar width occupy the body width. The total control width is then bodyWidth+title width, and the width from the size keyword is ignored.

The limits values *low*, *high*, and *base* and the value of *valExpr* control how the bar, if any, is drawn. The bar is drawn from a starting position corresponding to the *base* value to an ending position determined by the value of *valExpr*, *low* and *high*. *low* corresponds to the left side of the bar, and *high* corresponds to the right. The position that corresponds to the *base* value is linearly interpolated between *low* and *high*.

For example, with *low*= -10, *high*=10, and *base*= 0, a *valExpr* value of 5 will draw from the center of the bar area (0 is centered between -10 and 10) to the right, halfway from the center to the right of the bar area (5 is halfway from 0 to 10):



The *valExpr* must be executable at any time. The expression is stored and executed when the ValDisplay needs to be updated. However, execution will occur outside the routine that creates the ValDisplay, so you must not use local variables in the expression.

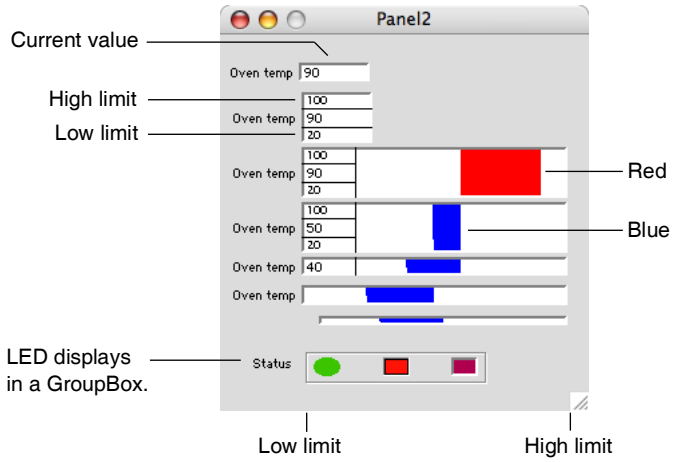
valExpr may be enclosed in quotes and preceded with a # character (see **When Dependencies are Updated** on page IV-233) to defer evaluation of the validity of the numeric expression, which may be needed if the expression references as-yet-nonexistent global variables or user-defined functions:

```
ValDisplay valdisp0 value=notAVar*2     // "unknown name or symbol" error
ValDisplay valdisp0 value=#"notAVar*2"      // still not valid, no error
Variable notAVar=3                      // now valid; ValDisplay works
```

In a ValDisplay, the #"" syntax permits use of a string expression. Normally, the # prefix signifies that the following text must be a literal quoted string. String expressions are evaluated at runtime to obtain the final expression for the ValDisplay. In other words, there is a level of indirection.

Examples

Here is a sampling of the various types of ValDisplay controls available:



You can use a ValDisplay to replace the bar mode with a solid color fill designed to look like an LED. Use the mode keyword with mode=1 to create an oval LED or mode=2 to create a rectangular LED. You can specify different frames with the rectangular LED but only a simple frame is available for the oval mode. Use mode=0 to revert to bar mode.

The color and brightness of the LED depends on the value that the ValDisplay is monitoring combined with the limits={*low*, *high*, *base*) setting, the two color settings used in bar mode along with a third color (zeroColor) that is used only in LED mode. When the value is between *low* and *high*, the color is a linear combination of endpoint colors. If *base* is between *low* and *high*, the endpoint colors are the low color and the zero color, or the zero color and the high color. For values outside the limits, the appropriate limiting color is chosen.

If *base* is less than the *low*, the endpoint colors are the low color and the high color. In this case, if the value is less than *low* the LED takes on the zero color.

You should use the bodyWidth setting in conjunction with LED mode to keep the LED from dramatically changing size or disappearing when the title is changed or if your experiment is moved to a different platform (Macintosh vs PC).

Try the ValDisplay Demo example experiment to see these different modes in action. Choose File→Example Experiments→Feature Demos→ValDisplay Demo.

See Also

See **Creating ValDisplay Controls** on page III-433 for more examples.

printf for an explanation of *formatStr*.

Chapter III-14, **Controls and Control Panels**, for details about control panels and controls.

Control Panel Units on page III-444 for a discussion of the units used for controls.

The **GetUserData** function for retrieving named user data.

The **ControlInfo** operation for information about the control.

Progress Windows on page IV-156 for an example of candy-stripe mode=4.