

Chapter
III-2

Annotations

| | |
|--|----|
| Overview | 41 |
| Annotations Quick Start | 41 |
| The Annotation Dialog..... | 42 |
| Modifying Annotations..... | 43 |
| Text Content..... | 43 |
| About Text Escape Codes | 44 |
| Font Escape Codes..... | 44 |
| Font Size Escape Codes..... | 44 |
| Relative Font Size Escape Codes | 45 |
| Special Escape Codes | 45 |
| Dynamic Escape Codes for Tags | 46 |
| Other Dynamic Escape Codes..... | 46 |
| TagVal and TagWaveRef Functions..... | 47 |
| Tabs | 48 |
| General Annotation Properties | 48 |
| Name..... | 48 |
| Frame | 48 |
| Color | 49 |
| Annotation Positioning..... | 49 |
| Textbox, Legend, and Color Scale Positioning in a Graph..... | 50 |
| Textbox and Legend Positioning in a Page Layout | 52 |
| Legends..... | 52 |
| Legend Text | 52 |
| Symbol Conditions at a Point | 53 |
| Freezing the Legend Text | 53 |
| Marker Size..... | 53 |
| Wave Symbol Centering | 53 |
| Wave Symbol Width | 54 |
| Tags..... | 54 |
| Tag Text..... | 55 |
| Tag Wave and Attachment Point | 55 |
| Changing a Tag's Attachment Point..... | 57 |
| Tag Arrows..... | 57 |
| Tag Line and Arrow Standoff..... | 58 |
| Tag Anchor Point..... | 58 |
| Tag Positioning | 58 |
| Tags Attached to Offscreen Points..... | 59 |
| Contour Labels Are Tags..... | 59 |
| Color Scales..... | 59 |
| ColorScale Main Tab | 60 |
| ColorScale Size and Orientation..... | 60 |
| ColorScale Axis Labels Tab | 61 |
| ColorScale Ticks Tab | 62 |
| Elaborate Annotations and Axis Labels..... | 63 |

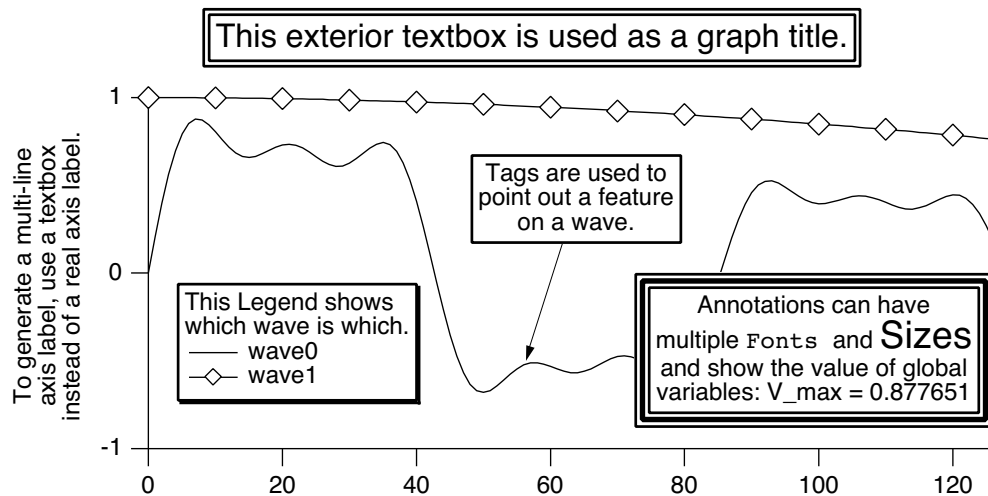
Chapter III-2 — Annotations

| | |
|---|----|
| Elaborate Annotations Versus Equation Editors | 63 |
| About Text Info Variables..... | 64 |
| Simple Text Info Variables Example..... | 64 |
| Text Info Variables Escape Codes | 64 |
| Elaborate Text Info Variables Example | 65 |
| More Examples..... | 65 |
| Programming with Annotations..... | 66 |
| Changing Annotation Names | 66 |
| Changing Annotation Types..... | 66 |
| Changing Annotation Text..... | 66 |
| Generating Text Programmatically..... | 66 |
| Deleting Annotations | 66 |

Overview

Annotations are custom objects that add information to a graph or a page layout. Most annotations contain text that you might use to describe the contents of a graph, point out a feature of a wave, identify the axis that applies to a wave, or create a legend. Igor automatically creates annotations for labeling contour plots. An annotation can also contain color scales showing the data range associated with colors in contour and image plots.

There are four types of annotation: textboxes, legends, color scales, and tags.

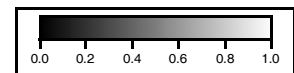


A textbox contains one or more lines of text which optionally may be surrounded by a frame, rotated, colored and aligned.

A legend is similar to a textbox except that it contains wave symbols for one or more waves in a graph. Legends are automatically updated when waves are added to or removed from the graph, or when a wave's appearance is modified.

A tag is also similar to a textbox except that it is attached to a point in a wave and can contain dynamically updated text describing that point. Tags can be added to a graph, but not to a page layout. In contour plots, Igor automatically generates tags to label the contour lines.

A color scale is similar to a legend except that it contains a color bar with an axis that spans the range of colors associated with the data. Color scales are automatically updated when the associated image plot, contour plot, $f(z)$ trace, or color index wave is modified. A color scale can also be completely disassociated from any data by directly specifying a named color table and an explicit numeric range for the axis.



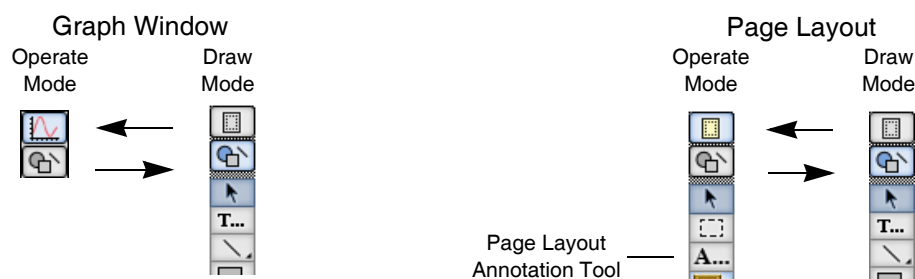
Annotations Quick Start

| To Do This | Do This |
|--|--|
| To add an annotation to a graph | Choose Add Annotation from the Graph menu. |
| To add an annotation to a page layout | Choose Add Annotation from the Layout menu or click with the annotation ("A") tool. For more information about annotations in page layouts, see Annotations in the Layout Layer on page II-381. |
| To modify an annotation in a graph | Double-click the annotation to bring up the Modify Annotation dialog. |
| To modify an annotation in a page layout | Single-click the annotation with the annotation tool. This brings up the Modify Annotation dialog. |

Chapter III-2 — Annotations

| To Do This | Do This |
|---|---|
| To change the annotation type | Use the Annotation pop-up menu in the Modify Annotation dialog, or use the proper Tag, TextBox, ColorScale, or Legend operation. |
| To move an existing annotation | Click in the annotation and drag it to the new position. If the annotation is frozen, this won't work — double-click it and make it moveable in the Annotation Position tab. To change a tag's attachment point, press Option (<i>Macintosh</i>) or Alt (<i>Windows</i>) and drag the tag text to the new attachment point on the wave. This works whether or not the tag is frozen. |
| To duplicate an existing annotation | Double-click the annotation, then click the Duplicate Textbox button in upper-right corner of the Modify Annotation dialog. If the annotation is a tag, the button is titled Duplicate Tag, etc. |
| To delete an annotation | Double-click the annotation, then click the Delete button in the Modify Annotation dialog. A tag can be deleted by dragging its attachment point off the graph. |
| To show the value of a global variable in an annotation | Type “\{variableName}” in the annotation text. See also Other Dynamic Escape Codes on page III-46. |

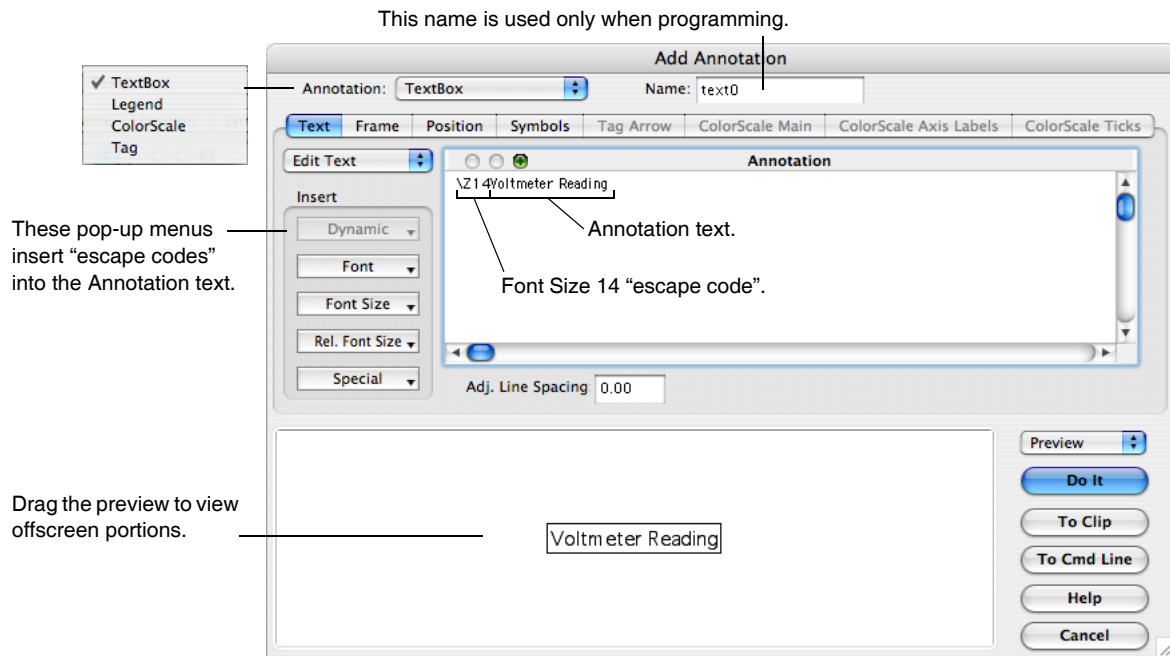
When manipulating annotations with the mouse, be sure that the graph or page layout are in the “operate” mode; *not* the “drawing” mode. The Tool bar indicates which mode the window is in:



The Annotation Dialog

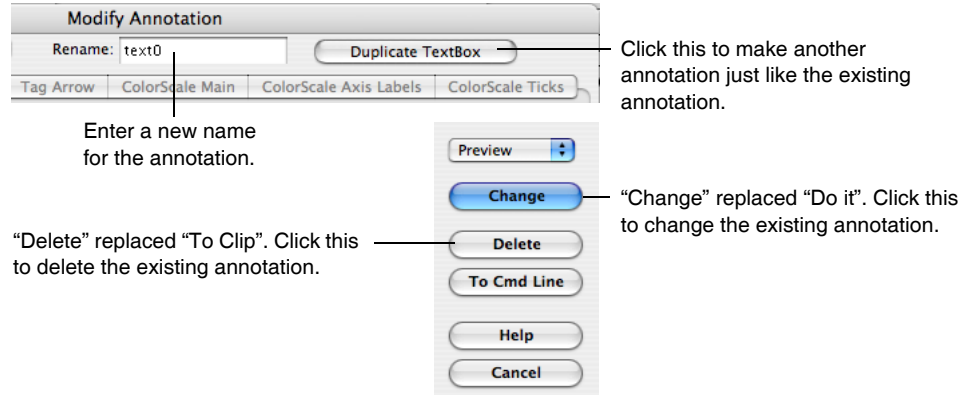
You can use the annotation dialog to create new or modify existing annotations. The dialog title is Add Annotation when a new annotation will be created and Modify Annotation when an existing annotation will be changed or duplicated. See **Modifying Annotations** on page III-43. The annotation dialog seems complex but comprises only a few major functions:

- A pop-up menu to choose the annotation type.
- A Name: setting.
- Tabs that group related Annotations settings. Some of the tabs apply to color scales only, and some of the settings in various tabs apply only to specific types of annotations.
- A Preview box to show what the annotation will look like or to display commands.
- The normal Igor dialog buttons.



Modifying Annotations

If an annotation is already in a graph you can modify it by double-clicking it while the graph is in the “operate” mode (see **Annotations Quick Start** on page III-41). The resulting Modify Annotation dialog is similar to the Add Annotation dialog except for a few items:



Single-clicking an annotation with the annotation (“A”) tool in a page layout also brings up the Modify Annotation dialog.

Text Content

You enter text into the Annotation text entry area in the Text tab. This “windoid” supports copy, cut, paste and undo using Command-C, Command-X, Command-V and Command-Z (*Macintosh*) or Ctrl+C, Ctrl+X, Ctrl+V and Ctrl+Z (*Windows*). Tab stops are provided; see **Tags** on page III-48. If your annotation has a lot of text, you can scroll and zoom this area for easier editing.

The annotation text may contain both plain text and “escape code” text which produces special effects such as superscript, font, font size and style, alignment, text color and so on. The text can contain multiple lines; just press Return. At any point when entering plain text you can choose a special effect from a pop-up menu within the Insert group, and Igor will enter the correct escape code. Igor wizards can type them in directly.

Chapter III-2 — Annotations

You can enter numbers into the text by simply typing them, or by referencing global variables or functions using **dynamic text**. Dynamic text is explained in **Other Dynamic Escape Codes** on page III-46.

As you type annotation text, the Preview box shows what the resulting annotation will look like. You can not enter text in the Preview box.

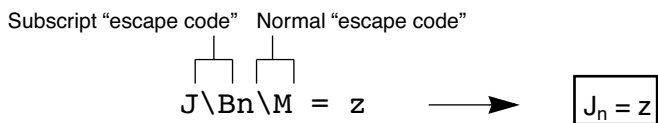
Sometimes the textbox you are creating will not fit in the preview box. You can move the previewed annotation around to see the hidden parts. When you position the cursor over the preview box, it changes to a hand; just drag the annotation.

If the preview area isn't showing the annotation, change the pop-up menu (just above the Do It or Change button) from Commands to Preview.

About Text Escape Codes

An escape code consists of a backslash character followed by one or more characters. It represents the special effect you selected. The effects of the escape code persist until overridden by a following escape code. The escape codes are cryptic but you can see their effects in the Preview box.

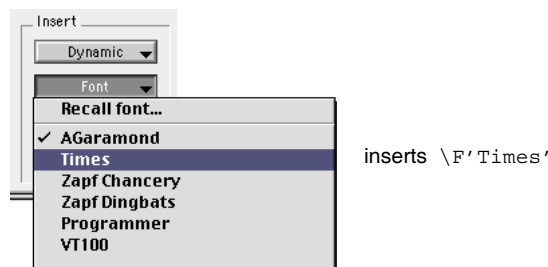
In the adjacent example, the subscript escape code “\B” begins a subscript and is not displayed in the annotation; the “n” that follows is plain text displayed as a subscript. The normal escape code “\M” overrides the subscript mode so that the plain text “= z” that follows has the original size and Y position (vertical offset) used for the “J”.



Font Escape Codes

Choosing an item from the Font pop-up menu inserts a code that changes the font for subsequent characters in the annotation. The checked font is the font currently in effect at the current insertion point in the annotation text entry area.

If you don't choose a font, Igor uses the default font or the graph font for annotations in graphs. You can set the default font using the Default Font item in the Misc menu, and the graph font using the Modify Graph item in the Graph menu. The Font pop-up menu also has a “Recall font” item. This item is used to make elaborate annotations as described under **Text Info Variables Escape Codes** on page III-64.

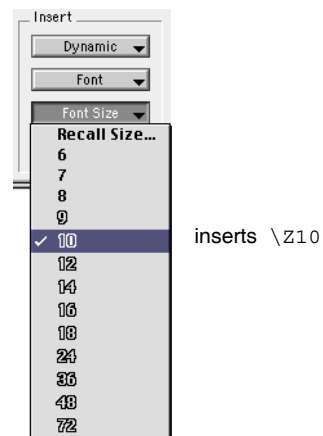


Font Size Escape Codes

Choosing an item from the Font Size pop-up menu inserts a code that changes the font size for subsequent characters in the annotation. The checked font size is the size currently in effect at the current insertion point in the annotation text entry area.

(To insert a size not shown, choose any shown size, and edit the escape code to contain the desired font size. Annotation font sizes may be 03 to 99 points; two digits are required after the “\Z” escape code.)

If you specify no font size escape code for annotations in graphs, Igor chooses a font size appropriate to the size of the graph unless you've specified a graph font size in the Modify Graph dialog. The default font size for annotations in page layouts is 10 points. The Font Size pop-up menu contains a “Recall size” item which is used to make elaborate annotations as described in the section **About Text Info Variables** on page III-64.



Relative Font Size Escape Codes

Choosing an item from the Rel. Font Size pop-up menu inserts a code that changes the relative font size for subsequent characters in the annotation. Use values larger than 100 to increase the font size, and values smaller than 100 to decrease the font size.

To insert a size not shown, choose any shown relative size, and edit the escape code to contain the desired relative font size. Annotation relative font sizes may be 001 to 999 (1% to 999%). Three digits are required after the “\Zr” escape code.

Don’t use, say, 50% followed by 200% and expect to get exactly the original font size back; rounding inaccuracies will prevent success (because font sizes are handled as only integers). For example, if you start with 15 point text and use \Zr050 (50%) the result is 7 point text. 200% of 7 points is only 14 point text:

| | |
|---|---|
| <p>Voltmeter Readings (past 30 days) Voltmeter Readings</p> | <p>\C\Z15Voltmeter Readings \Zr050(past 30 days) \Zr200Voltmeter Readings</p> |
|---|---|

Instead, use the Normal “\M” escape code (or an absolute font size or a recalled font size) to return to a known font size. In the following example, the initial 15 point font size is saved in the main text info variable (the “[0]” escape code, see **Text Info Variables Escape Codes** on page III-64), whose size is recalled by the Normal escape code.

| | |
|---|---|
| <p>Voltmeter Readings (past 30 days) Voltmeter Readings</p> | <p>\C\Z15[0Voltmeter Readings \Zr050(past 30 days) \MVoltmeter Readings</p> |
|---|---|

Special Escape Codes

Choosing an item from the Special pop-up menu inserts an escape code that makes subsequent characters superscript, subscript or normal, affects the style, position or color of subsequent text, or inserts the symbol with which a wave is plotted in a graph.

The first four items, Store Info, Recall Info, “Recall X position”, and “Recall Y position” are used to make elaborate annotations as described under **Elaborate Annotations and Axis Labels** on page III-63.

The Style item brings up a subdialog that you use to change the style (bold, italic, etc.) for the annotation at the current insertion point in the annotation text entry area. This subdialog has a Recall Style checkbox that is used in elaborate annotations with text info variables.

The Superscript and Subscript items insert an escape code that makes subsequent characters superscript or subscript. Use the Normal item to return the text to the original text size and Y position.

Chapter III-2 — Annotations

The Backslash item inserts a code to insert a backslash that prints, rather than one which introduces an escape code. Igor does this by inserting two backslashes, which is an escape code that prints a backslash. Weird, huh?

The Normal item inserts a code to return to the original font size and baseline (which has no vertical offset such as is used to produce super- and subscripts). More precisely, Normal sets the font size and baseline to the values stored in text variable 0 (see **About Text Info Variables** on page III-64). The font and style are not affected.

The Justify items insert a code to align the current and following lines.

The Color item inserts a code to color the following text. The initial text color and the annotation *background* color are set in the Frame Tab.

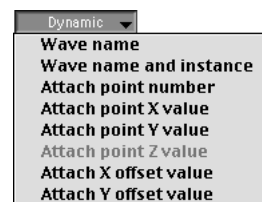
The Wave Symbol item inserts a code that prints the symbol (line, marker, etc.) used to display the wave trace in the graph. This code is inserted automatically in a legend. You can use this menu item to manually insert a symbol into a tag, textbox, or color scale. For graph annotations, the submenu lists all the trace name instances in the top graph. For layout annotations, all the trace name instances in all graphs in the layout are listed.

The Character item presents a table from which you can select text and special characters to add to the annotation.

The Marker item inserts a code to draw a marker symbol. These symbols are independent of any traces in the graph.

Dynamic Escape Codes for Tags

The Dynamic pop-up menu inserts escape codes that apply only to tags. These codes insert information about the wave or point in the wave to which the tag is attached. This information automatically updates whenever the wave or the attachment point changes.



| Dynamic Item | Effect |
|-------------------------|---|
| Wave name | Displays the name of the wave to which the tag is attached. |
| Trace name and instance | Same as wave name but appends an instance number (e.g., #1) if there is more than one trace in the graph associated with a given wave name. |
| Attach point number | Displays the number of the tag attachment point. |
| Attach point X value | Displays the X value of the tag attachment point. |
| Attach point Y value | Displays the Y value of the tag attachment point. |
| Attach point Z value | Displays the Z value of the tag attachment point. Available only for contour traces, waterfall plots, or image plots. |
| Attach X offset value | Displays the trace's X offset. |
| Attach Y offset value | Displays the trace's Y offset. |

See also **TagVal and TagWaveRef Functions** on page III-47. These functions provide the same information as the Dynamic pop-up menu items but with greater flexibility.

Other Dynamic Escape Codes

You can type the “dynamic text escape code” which inserts dynamically evaluated text into any kind of annotation using the escape code sequence:

```
\{ dynText }
```

where *dynText* may contain numeric and string expressions. If *dynText* references a numeric variable, string variable or wave “object”, this makes the annotation dependent on the referenced object (see Chapter IV-9, **Dependencies**, for further details). If the object changes, Igor automatically updates the dynamic text.

Note: Making an annotation dependent on a variable or wave is often a bad idea because you might inadvertently forget to create the variable or wave, delete it, or change its value. Thus, you should use this feature only when other techniques are insufficient. In most cases, it is better to generate text programmatically, as described in **Generating Text Programmatically** on page III-66.

The numeric and string expressions are evaluated in the context of the “root” data folder. If you are not using data folders, just use the names of the waves and variables. If you are using data folders, use the full data folder path of any nonroot objects in the expressions. For more on Data Folders, see Chapter II-8, **Data Folders**.

dynText can take two forms: an easy form for a single numeric expression, and a more complex form that provides precise control over the formatting of the result.

The easy form is:

```
\{numeric-expression}
```

This evaluates the numeric expression and prints with generic (“%g”) formatting. For example:

```
Twice \{K0} is \{K0*2}
```

creates this textbox when K0 is 7:

Twice 7 is 14

If K0 changes, Igor automatically updates the textbox.

The full form for *dynText* is:

```
\{formatStr, list-of-numeric-or-string-expressions}
```

formatStr and *list-of-numeric-or-string-expressions* are treated the same as for the Printf operation. For instance, this example has a format string, a numeric expression and a string expression:

```
\{"Twice K0 is %g, and today is %s", 2*K0, date() }
```

It produces this result:

Twice K0 is 14, and today is Thu, May 4, 2000

Don’t try to use any annotation escape codes in the format string or numeric or string expressions; they don’t work within the `\{ ... }` context.

Also, the format string and string expressions do not support multiline text. If you need to use multiline text, use the technique described in **Generating Text Programmatically** on page III-66.

As an aid in typing the expressions, Igor considers carriage returns within the braces to be equivalent to spaces. Thus you can type in the Add Annotation dialog:

```
\{
  "Twice K0 is %g, and today is %s",
  2*K0,
  date()
}
```

and get the same result as above. These carriage returns can be typed directly in the Add Annotations dialog, or be typed as “\r” in a macro, function, or the command line.

TagVal and TagWaveRef Functions

If the annotation is a tag, you can use the functions **TagVal** (page V-694) and **TagWaveRef** (page V-695) to display information about the data point to which the tag is attached. For example, the following displays the Y value of the tag’s data point:

```
\{"%g", TagVal(2) }
```

This is identical in effect to the “\0Y” escape code which you can insert by choosing the “Attach point Y value” item from the Dynamic pop-up menu. The benefit of using the TagVal function is that you can use a formatting technique other than %g. For example:

```
\{"%5.2f", TagVal(2) }
```

Chapter III-2 — Annotations

TagVal is capable of returning all of the information that you can access via the Dynamic menu escape codes. Use it when you want to control the numeric format of the text.

The TagWaveRef function returns a reference to the wave to which the tag is attached. You can use this reference just as you would use the name of the wave itself. For example, given a graph displaying a wave named wave0 (in the root data folder), the following tag text displays the average value of the wave:

```
\{ "%g", mean (wave0, -INF, INF) }
```

This is fine, but if you move the tag to another wave it will still show the average value of wave0. Using TagWaveRef, you can make this show the average value of whichever wave is tagged:

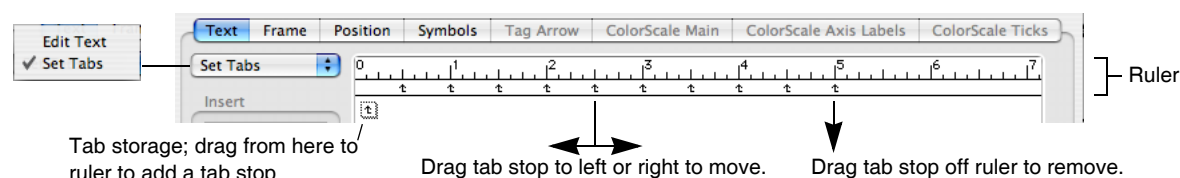
```
\{ "%g", mean (TagWaveRef ( ) , -INF, INF) }
```

The TagVal and TagWaveRef functions work only while Igor is in the process of evaluating the annotation text, so you should use them only in annotation dynamic text or in a function called from annotation dynamic text.

Also see the **TraceNameToWaveRef** function (page V-711), which returns a reference to a wave given a graph's trace name and can be freely used in macros and functions.

Tabs

The Text Tab's Annotation text area actually has two functions which are controlled by the pop-up menu at its top-left corner. If you choose Set Tabs from this pop-up menu, Igor shows the tab stops for the annotation.



By default, an annotation has 10 tab stops spaced 1/2 inch apart. You can change the tab stops by dragging them along the ruler. You can remove a tab stop by dragging it down off the ruler. You can add a tab by dragging it from the tab storage area at the left onto the ruler.

Igor supports a maximum of 10 tab stops per annotation and they are always left-aligned tabs. There is only one set of tab stops per annotation and they affect the entire annotation.

When setting the tabs, the Insert Text pop-up menus are disabled; return the pop-up menu to Edit Text to reenable them.

General Annotation Properties

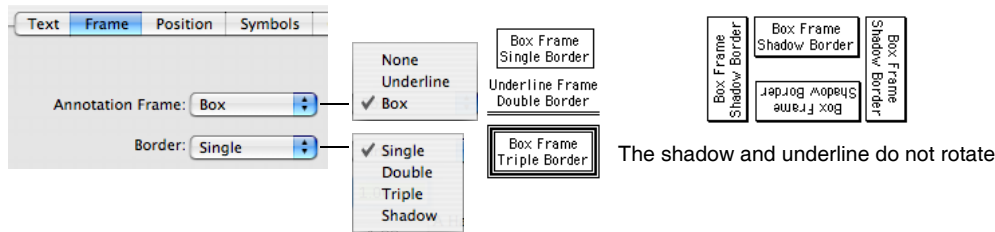
Most annotation properties are common to all kinds of annotations.

Name

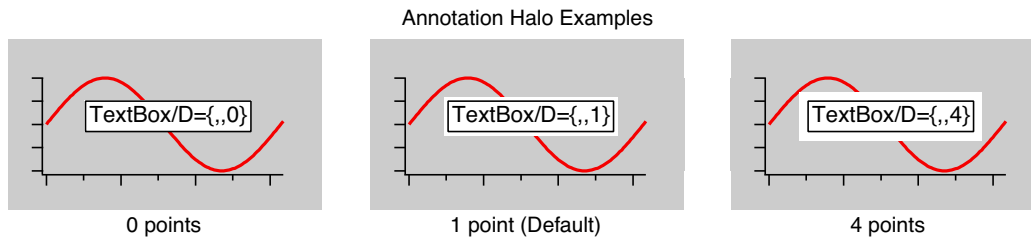
You can assign a name to the annotation with the Name item. In the Modify Annotation dialog, this is the Rename item. The name is used to identify the annotation in a Tag, TextBox, ColorScale, or Legend operation. Annotation names must be unique to the window they are in. The names Igor automatically puts here already are unique, but you can change them if you want. See **Programming with Annotations** on page III-66 for more information.

Frame

In the Frame Tab, the Frame and Border pop-up menus allow you to frame the annotation with a box or shadow box, to underline the textbox, or to have no frame at all. The line size of the frames and the shadow are set by the Thickness and Shadow values.



By default, framed annotations also have a 1-point “halo” that surrounds them to separate them from their surroundings. The halo takes on the color of the annotation’s background color. You can change the width of this halo to a value between 0 and 10 points by setting the desired thickness in the Halo box in the Frame tab. A fractional value such as 0.5 is permitted.



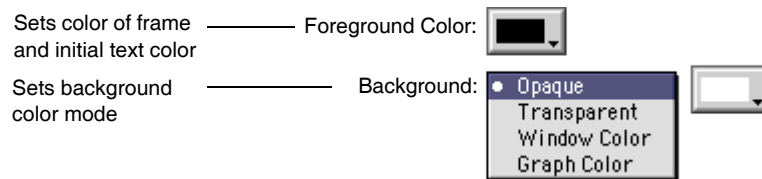
Specifying a negative value for Halo allows the halo thickness to be overridden by the global variable `V_TBBufZone` in the root data folder. If the variable doesn’t exist, the absolute value of the entered value is used. The default halo value is -1. You can override the default halo by setting the `V_TBBufZone` global in a `IgorStartOrNewHook` hook function. See the example in **User-Defined Hook Functions** on page IV-251.

Color

The Frame tab contains most of the annotation’s color settings.

Use the Foreground Color pop-up menu to set the *initial* text color. You can *change* the color of the text from the initial foreground color by inserting a color escape code using the Special pop-up menu in the Text tab.

Use the Background pop-up menu to set the background mode and color.



| Background Color Mode | Effect |
|-----------------------|---|
| Opaque | The annotation background covers objects behind. You choose the background color from a pop-up menu. |
| Transparent | Objects behind the annotation show through. |
| Graph color | The background is opaque and is the same color as the graph background color. This is not available for annotations added to page layout windows. |
| Window color | The background is opaque and is the same color as the window background color. |

Annotation Positioning

You can rotate the annotation into the four principal orientations with the in the Position tab's Rotation pop-up menu. You can also enter an arbitrary rotation angle (in integral degrees) directly. Tags attached to

Chapter III-2 — Annotations

contour traces and color scales have specialized rotation settings; see **Modifying Labels** on page II-336 and **ColorScale Size and Orientation** on page III-60.

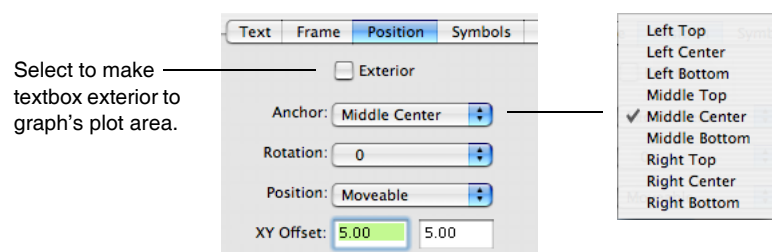
You can position an annotation anywhere in a window by dragging it and in many cases this is all you need to know. However, if you attend to a few extra details you can make the annotation go to the correct position even if you resize the window or print the window at a different size.

This is particularly important when a graph is placed into a page layout window, where the size of the placed graph usually differs from the size of the graph window.

Annotations are positioned using X and Y offsets from “anchor points”. The meaning of these offsets and anchors depends on the type of annotation and whether the window is a graph or layout. Tags, for instance, are positioned with offsets expressed as a percentage of the horizontal and vertical sizes of the graph. See **Tag Positioning** on page III-58.

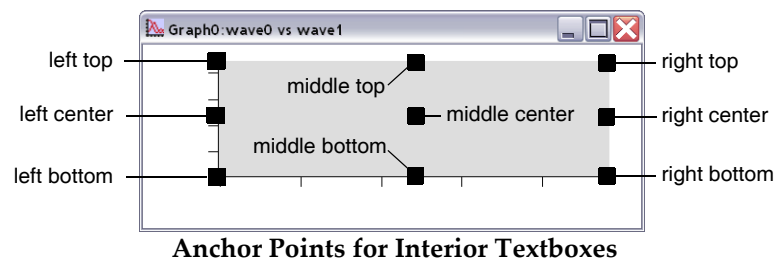
Textbox, Legend, and Color Scale Positioning in a Graph

A textbox, legend, and color scale are positioned identically, so this description will use “textbox” to refer to all of them. A textbox in a graph can be “interior” or “exterior” to the graph’s plot area. You choose this positioning option with the Exterior checkbox:



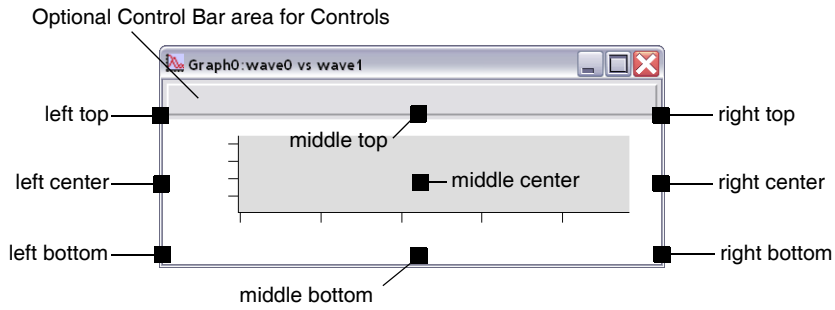
The Anchor pop-up menu specifies the precise location of the reference point on the plot area or graph window edges. It also specifies the location *on the textbox* which Igor considers to be the “position” of the textbox.

An interior textbox is positioned relative to a reference point on the edge of a graph’s plot area. (The plot area is the central rectangle in a graph window where traces are plotted. The standard left, right, bottom, and top axes surround this rectangle.)



Anchor Points for Interior Textboxes

An exterior textbox is positioned relative to a reference point on the edge of the window and the textbox is normally outside the plot area.



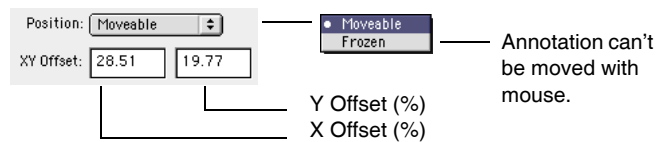
Anchor Points for Exterior Textboxes

The purpose of the exterior textbox is to allow you to place a textbox away from the plot area of the graph. For example, you may want it to be above the top axis of a graph or to the right of the right axis. Igor tries to keep exterior textboxes away from the graph by pushing the graph away from the textbox.

The direction in which it pushes the graph is determined by the textbox's anchor. If, for example, the textbox is anchored to the top then Igor pushes the graph down, away from the textbox. If the anchor is middle-center, Igor does not attempt to push the graph away from the textbox. So, an exterior textbox anchored to the middle-center behaves like an interior textbox.

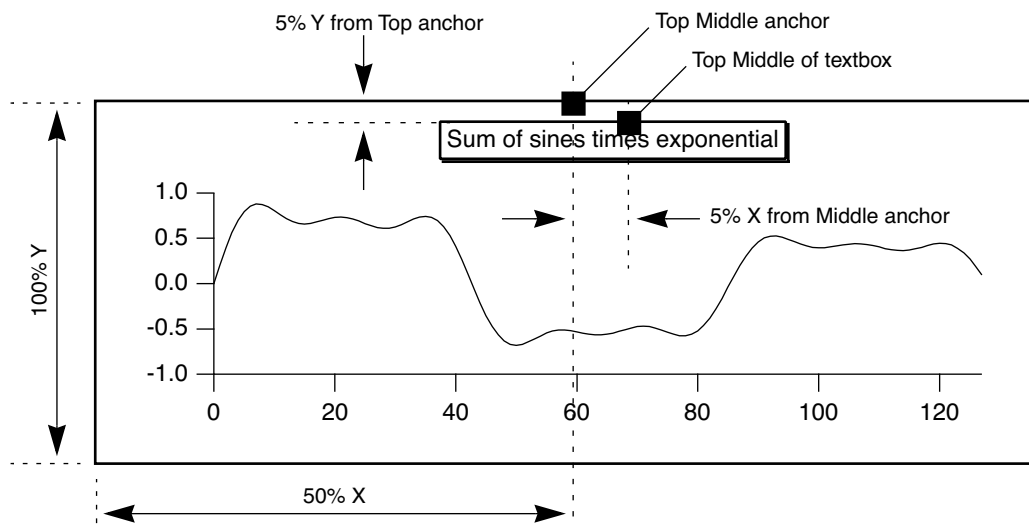
If you specify a margin, using the Modify Graph dialog, this overrides the effect of the exterior textbox, and the exterior textbox will not push the graph.

The XY Offset in the Position Tab gives the horizontal and vertical offset from the anchor to the textbox as a *percentage* of the horizontal and vertical sizes of the graph's plot area for interior textboxes or the window sizes for exterior textboxes.



The **Position** pop-up menu "freezes" the position of the textbox so that it can not be moved with the mouse. This is useful if you are using the textbox to label an axis tick mark and don't want to accidentally move it.

In the following example we wanted to center the textbox above and outside the plot area, so we chose an exterior textbox, a middle top anchor point and X and Y offsets of 5 (percent of the graph window's width and height). The choice of a top anchor pushed the graph below the textbox; a middle anchor does not push the graph left or right.

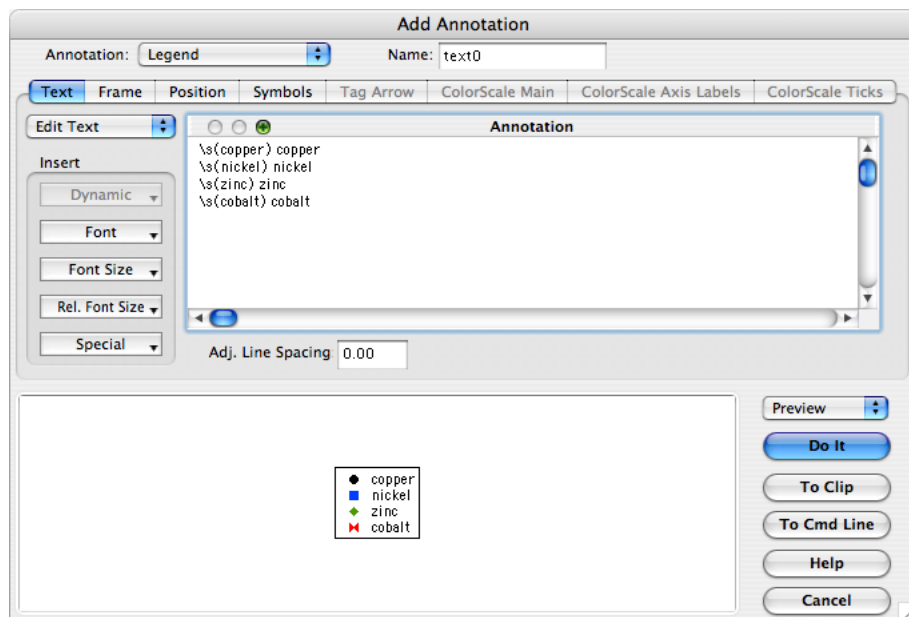


Textbox and Legend Positioning in a Page Layout

Annotations in a page layout window are positioned relative to an anchor point on the edge of the printable part of the page. The distance from the anchor point to the textbox is determined by the X and Y offsets which are in percent of the printable page. Annotations in a page layout can not be “frozen” as they can in a graph (see above).

Legends

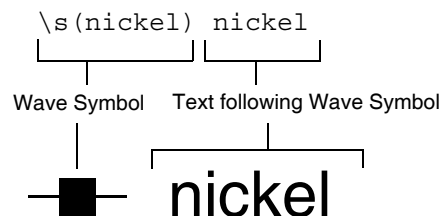
A legend is very similar to a textbox. It shows the “wave symbol” for some or all of the waves in a graph or page layout. (We also call this a “legend symbol”.) To make a legend, choose Add Annotation from the Graph or Layout menu.



The pop-up menu at the top left of the dialog sets the type of the annotation (TextBox, Legend, ColorScale or Tag). If you choose Legend *when there is no text in the text entry area*, Igor automatically generates the text needed for a “standard legend”. To keep the standard legend, just click Do It. However, you can also modify the legend text as you can for any type of annotation.

Legend Text

The legend text consists of an escape sequence to specify the wave whose symbol you want in the legend plus plain text. In this example dialog above, `\s(nickel)` is the escape sequence that inserts the wave symbol (a line and a filled square marker) for the wave whose name is nickel. This escape sequence is followed by a space and the name of the wave. The part after the escape sequences is plain text that you can edit as needed.



Instead of specifying the name of the trace for a legend symbol, you can specify the trace number. For example, `\s(#0)` displays the legend for trace number 0.

There are only two differences between a legend and a textbox. First, text for a legend is automatically generated when you choose Legend from the pop-up menu while there is no text in the text entry area. Second, if you append or remove a wave from the graph or rename a wave, the legend is automatically updated by adding or removing wave symbols. Neither of these two actions occur for a textbox (or a tag or color scale, for that matter).

Symbol Conditions at a Point

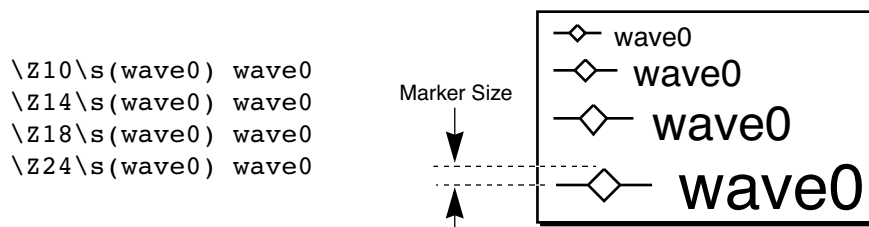
You can specify that a legend symbol shows the conditions at a specific point on a trace by appending the point number in brackets to the trace name. For example `\s(nickel[3])`. This feature is useful when a trace uses $f(z)$ mode or when a single point on a trace has been customized. This feature was added in Igor Pro 6.20.

Freezing the Legend Text

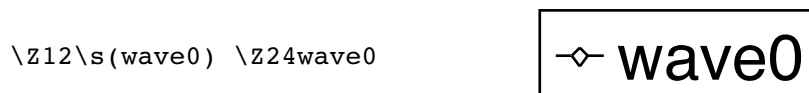
Occasionally you may not want the legend to update automatically when you append waves to the graph. You can freeze the legend text by converting the annotation to a textbox. To create a nonupdating legend, bring up the Add Annotation dialog. Choose Legend from the pop-up menu to get Igor to create the legend text, then choose TextBox from the pop-up menu. Now you have converted the legend to a textbox so it will not be automatically updated.

Marker Size

A wave symbol will contain a marker if the wave is drawn with one. Normally the size of the marker drawn in the annotation is based on the font size in effect when the marker is drawn. When you set the font size before the wave symbol escape code, both the marker and following text will be adjusted in proportion:

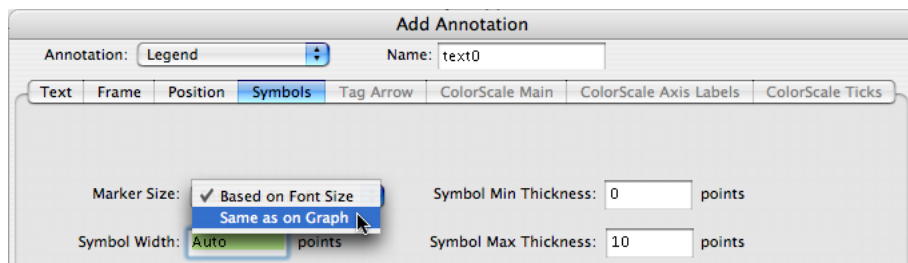


You can also change the font size after the wave symbol, which sets the size of the following text. Here is an example that uses a small marker size with large text.



The `\Z12` sets the font size to 12 points. This controls the size of the marker. The `\Z24` sets the font size of the following text to 24 points.

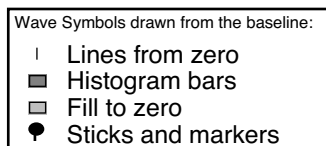
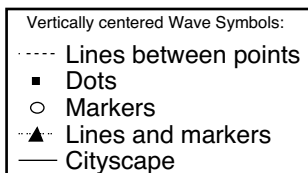
The second method for setting the size of a marker is to choose “Same as on Graph” from the Marker Size pop-up menu in the Symbols Tab.



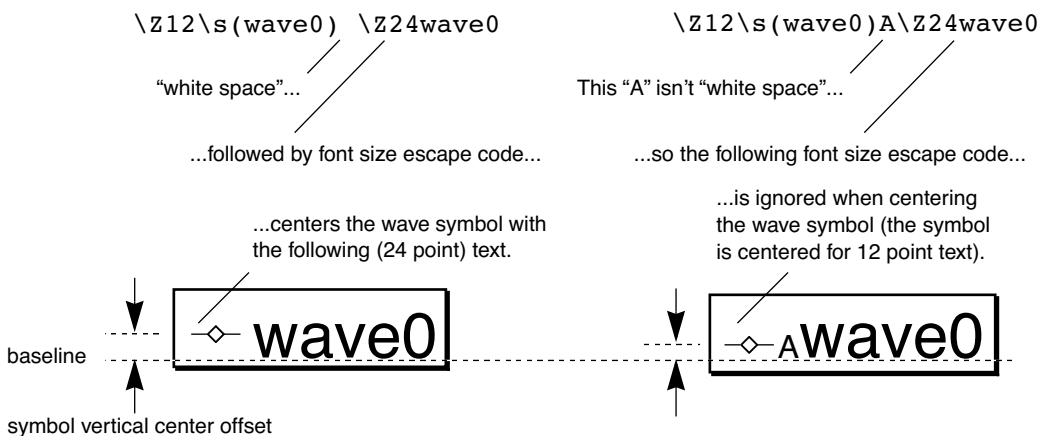
With “Same as on Graph” chosen, the marker size will match the size of the corresponding marker in the graph, regardless of the size of the annotation’s text font.

Wave Symbol Centering

Some wave symbols are vertically centered relative to either the text that precedes or the text that follows the wave symbol escape code, and other symbols are drawn with their bottom at the baseline:



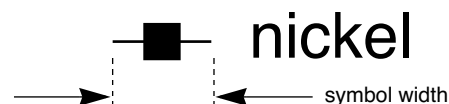
Vertically centered symbols are centered based on the height of the current font and size except that, if the symbol is followed by optional white space plus a font size escape code (i.e. `\Z09`, `\Z] n`, `\] n`, or `\M`), then the centering is based on the *following* font size. This automatically centers a symbol with following text that is much bigger or smaller than the symbol.



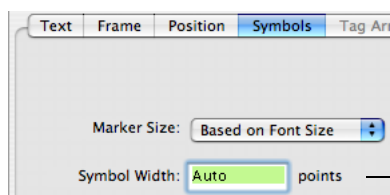
"White space" in this context is a run of space characters or tab characters. Note that on the Macintosh Option-Space is *not* considered to be "white space", and line breaks are not white space on any platform.

Wave Symbol Width

The wave symbol width is the width in which all wave symbols are drawn.



This width is controlled by the font size of the text preceding the wave symbol, or it is set explicitly to a given number of points using the Symbol Width value in the Symbols Tab.

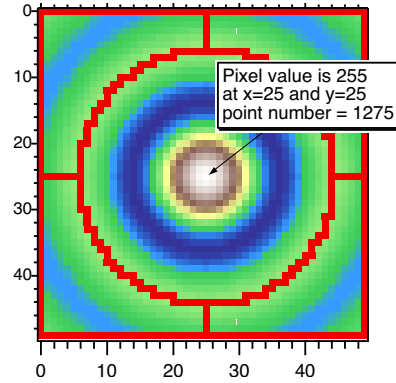
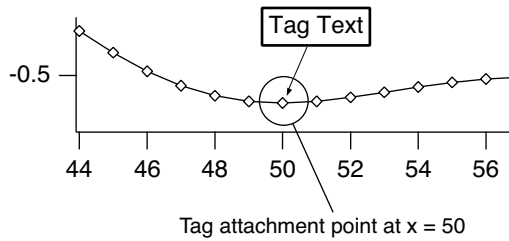


The word "Auto" or the number 0 means the Symbol Width is controlled by font size

You can widen or narrow the overall symbol size by entering a nonzero width value. If you use large markers with small text, you may find it necessary to reduce the wave symbol width using this setting. For some line styles that have long dash/gap patterns, you will want to enter an explicit value large enough to show the pattern, such as 36 (1/2 inch) or 72 points. The maximum is 1000 points.

Tags

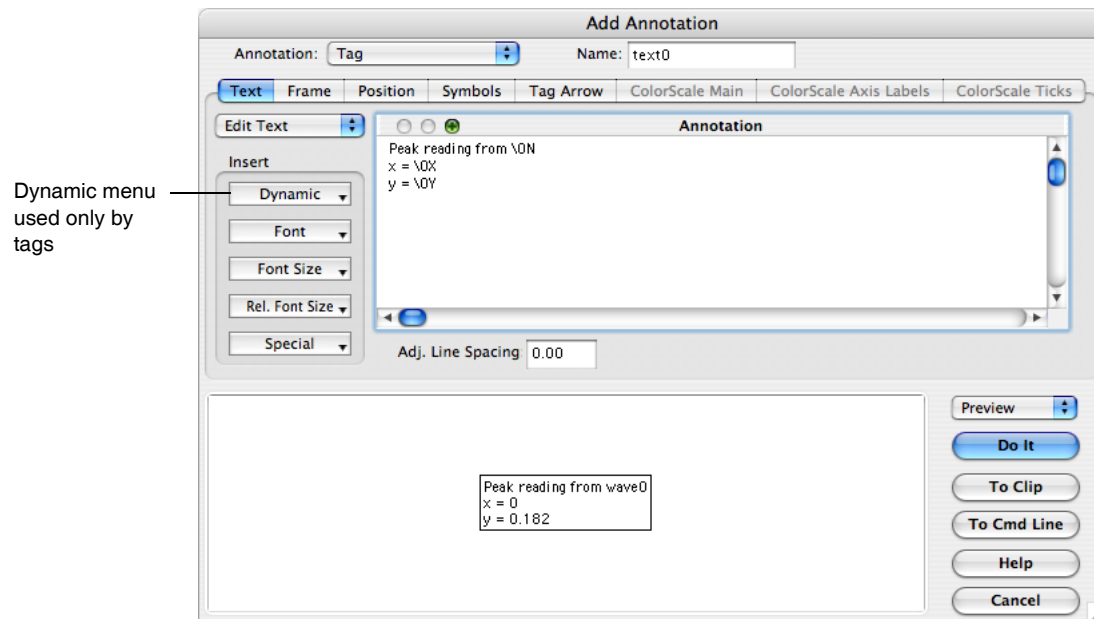
A tag is like a textbox but with several added capabilities. A tag is attached to a particular point on a particular trace, image, or waterfall plot in a graph:



Tags can not be added to page layouts (a graph containing a tag can be added to the page layout, of course). Igor automatically generates tags to label contour plots.

Tag Text

Text in a tag can contain anything a textbox or legend can handle, and more. The Dynamic pop-up menu of the Text Tab inserts escape codes that apply only to tags. These codes insert information about the wave the tag is attached to, or about the point in the wave to which the tag is attached. This information is “dynamically” updated whenever the wave or the attachment point changes. See **Dynamic Escape Codes for Tags** on page III-46.

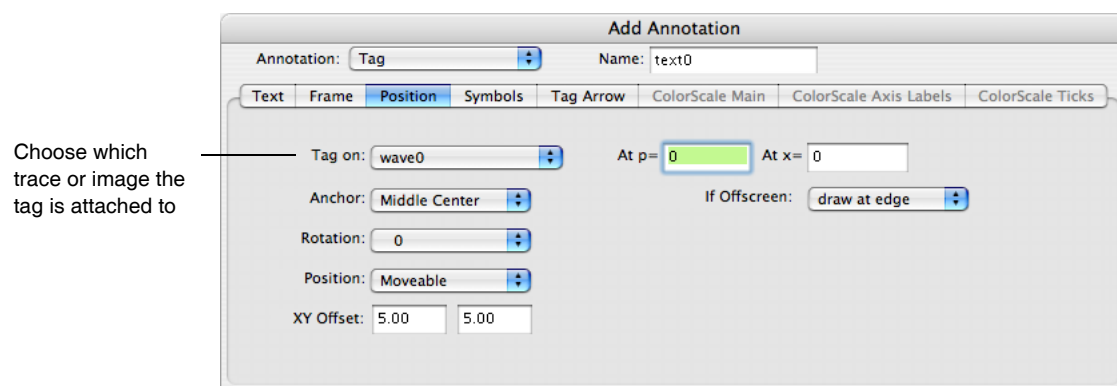


The TagVal and TagWaveRef functions are also useful when creating a tag with dynamic text. See **TagVal and TagWaveRef Functions** on page III-47.

Tag Wave and Attachment Point

You specify which wave the tag is attached to in the Position Tab, by choosing a wave from the “Tag on” pop-up menu.

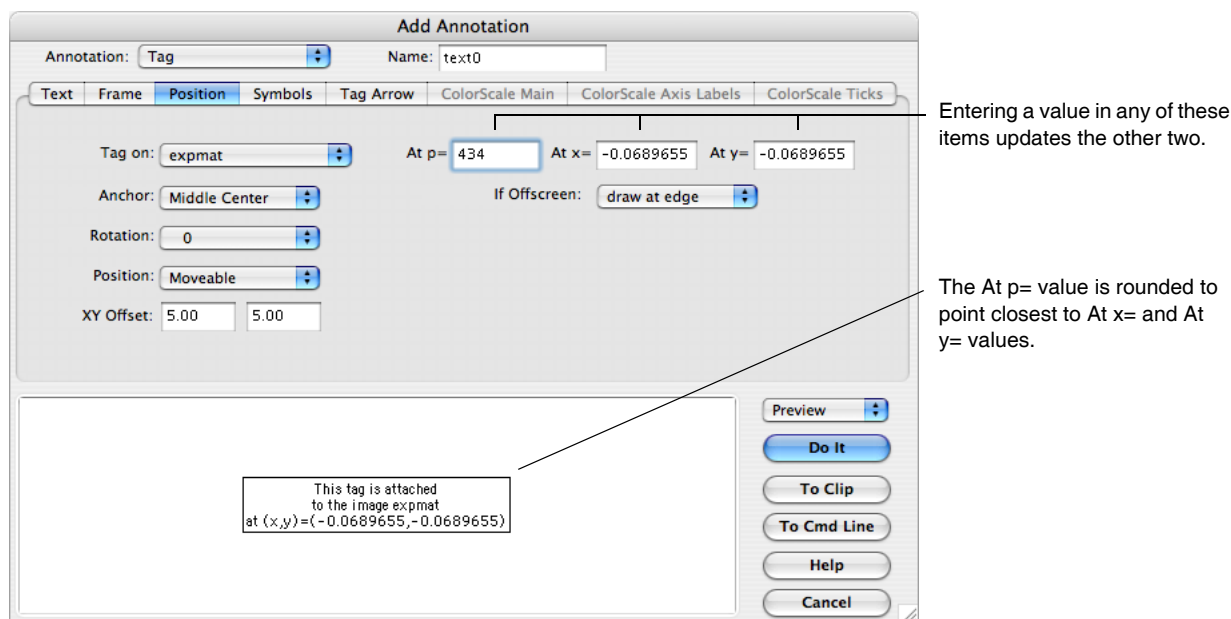
Chapter III-2 — Annotations

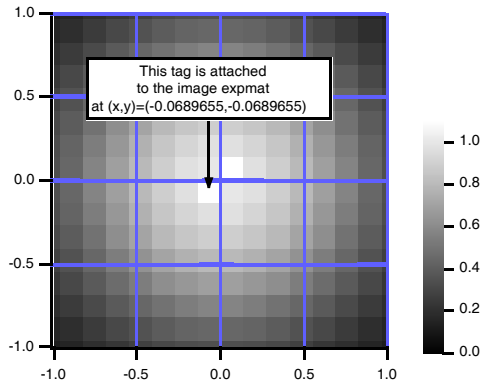


You specify which point the tag is attached to by entering the point number in the “At p=” entry or an X value in the “At x=” entry. The X value is in terms of the X scaling of the wave to which you are attaching the tag. This is not necessarily the same as the X axis position of the point if the wave is displayed in XY mode (versus a wave providing X values). This is the X value *of the wave’s point* to which the tag is attached. If this distinction mystifies you, see **Waveform Model of Data** on page II-77.

The attachment point of a tag in a (2D) image or waterfall plot is treated a bit differently than for 1D waves. In images it is the sequential point number linearly indexed into the matrix array. The dialog will convert this point number displayed in the “At p=” entry into the X and Y values, and vice versa.

Because it is the point number that determines the actual attachment point, entered “At x=” and “At y=” values are not necessarily exactly where the tag is attached. In the following dialog the X and Y values were entered as 0, but the nearest point (434) results in an actual attachment point of (-0.0689655,-0.0689655):




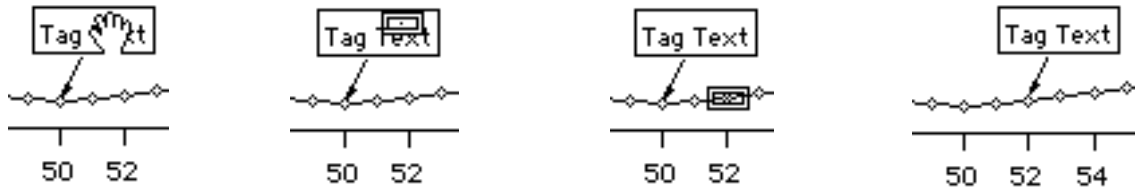


Notice that no point in the image is centered on $(x,y) = (0,0)$.

Sometimes, however, it is just easier to manually position the tag by dragging it with the cursor, as described in the next section.

Changing a Tag's Attachment Point

Once a tag is on a graph you can attach it to a different point by pressing Option (Macintosh) or Alt (Windows), clicking in the tag, and dragging the special tag cursor  you will see to the new attachment point of the wave. You must drag the tag cursor to the point *on the trace* to which you want to attach the tag, not to the position on the screen where you want the tag text to appear. The dot in the center of the tag cursor shows where the tag will be attached.

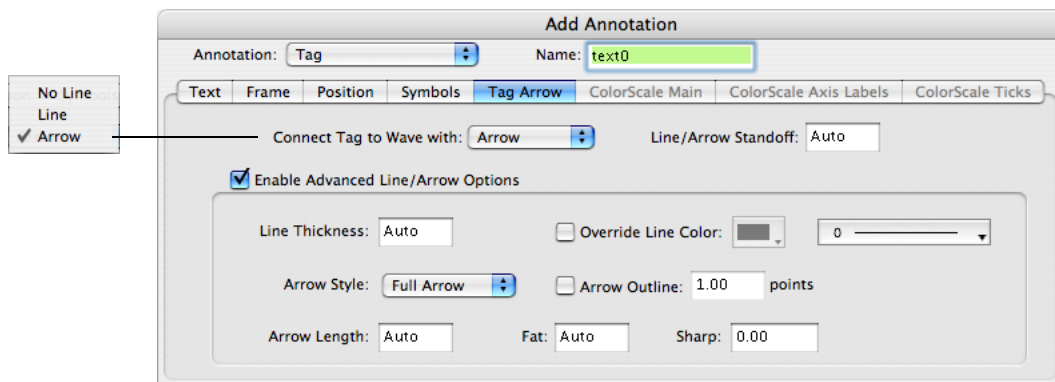


1. Move cursor over tag text.
2. Hold down Option or Alt.
3. Drag to new attachment point.
4. Tag attaches to and moves to the new point.

If you drag the tag cursor off the graph, the tag will be deleted from the graph.

Tag Arrows

You can indicate a tag's attachment point with an arrow or line drawn from the tag's anchor point using the "Connect Tag to Wave with" pop-up menu in the Tag Arrows tab.



You can adjust how close the arrow or line comes to the data point by setting the Line/Arrow Standoff distance (in points). In the Preview, the standoff is demonstrated by how close the arrow is drawn to the edge of the preview area

Chapter III-2 — Annotations

The Advanced Line/Arrow options are compatible with Igor 6.02 or later; they give you added control of the line and arrow characteristics.

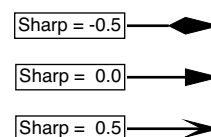
A Line Thickness value of 0 corresponds to the default (nonadvanced) line thickness of 0.5 points, otherwise, enter a value up to 10.0 points. To make the line disappear, select No Line from the “Connect Tag to Wave with” popup menu.

The line color is normally set by the annotation frame color (in the Frame tab). You can override this by checking the Override Line Color checkbox and choosing a color from the popup menu.

Change the attachment line’s style from the default solid line using the line style popup menu.

If “Connect Tag to Wave with” popup is set to Arrow, you can alter the default appearance of the arrow head using the remaining controls: full or half arrow head (left or right), filled or outlined arrow head, and alter the arrow head’s length from the default (Auto) to the given length in points.

The Sharp option is a small value between -1.0 and 1.0 (0 is the default).



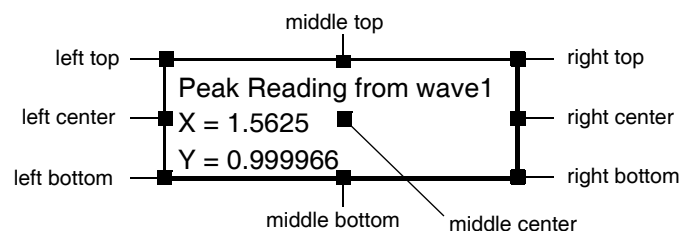
The Fat option can be Auto (or 0) for the default width-to-length ratio of 0.5. Larger numbers result in fatter arrows. If the number is small (say, 0.1), the arrow may seem to disappear unless the Arrow Length is made longer. Printed arrows can be narrower than screen-displayed arrows.

Tag Line and Arrow Standoff

You can specify how close to bring the line or arrow to that trace with the “Line/Arrow standoff” setting in the Frame Tab. You can set the distance by entering an explicit distance in points, or typing “auto” (a value of “0” also means “auto”) which varies the distance according to the output device resolution and graph size. Use a value of 1 to bring the line as near to the trace as possible. When the wave is graphed with markers, you might prefer to set the standoff to a value larger than the marker size so that the line or arrow does not intersect the marker. The left tag in the above example is using the auto standoff, and the right tag has a standoff of 6 points.

Tag Anchor Point

A tag has an Anchor point that is on the tag itself. If there is an arrow or line, it is drawn from the anchor point on the tag to the attachment point on the trace. The anchor setting also determines the precise spot on the tag which represents the position of the tag.



Anchors on Tags

The line is always drawn behind the tag so that if the anchor point is middle center the line doesn’t deface the text; the examples above have a middle center anchor.

Tag Positioning

The position of a tag is determined by the position of the point to which it is attached and by the “Tag XY offset” settings. The “XY offset” gives the horizontal and vertical distance from the attachment point to the tag’s anchor in percentage of the horizontal and vertical sizes of the graph’s plot area.

Once a tag is on a graph you can change its “XY offset” and therefore its position by merely dragging it. You can prevent the tag from being dragged by choosing “frozen” in the Position pop-up menu in the Position Tab. Igor freezes tags when it creates them for contour labels.

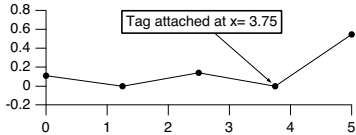
The interior/exterior setting used with textboxes does not apply to tags (see **Textbox, Legend, and Color Scale Positioning in a Graph** on page III-50).

Tags Attached to Offscreen Points

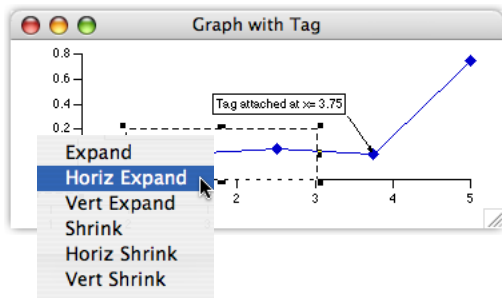
When only a portion of a wave is shown in a graph, it is possible that the attachment point of a tag isn't shown in the graph; it is "off screen" or "out-of-range".

This usually occurs because the graph has been manually expanded or the axes are not autoscaled. Igor draws the attachment line toward the offscreen attachment point.

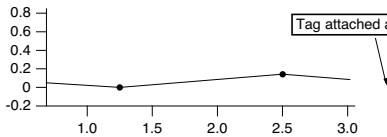
In this example graph, the attachment point at $x=3.75$ falls within the range of displayed X axis values.



If we zoom the graph's X range to exclude the $x=3.75$ attachment point...



... the tag attachment point is offscreen but the tag is still drawn.



However, you can suppress the drawing of a tag whose attachment point is offscreen by choosing "Hide the Tag" from the If Offscreen pop-up menu in the Position Tab.

If you want to see or modify a tag that is hidden, autoscale the graph so that it will no longer be hidden.

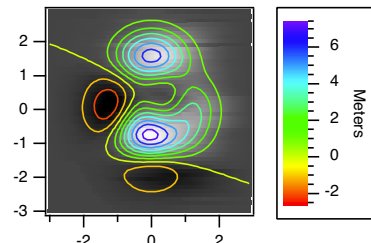
Contour Labels Are Tags

Igor uses specialized tags to create the numerical labels for contour plots. The specialization adds a "tangent" feature to automatically orient the tag along the path of the contour lines. See **Contour Labels** on page II-335 for details.

Color Scales

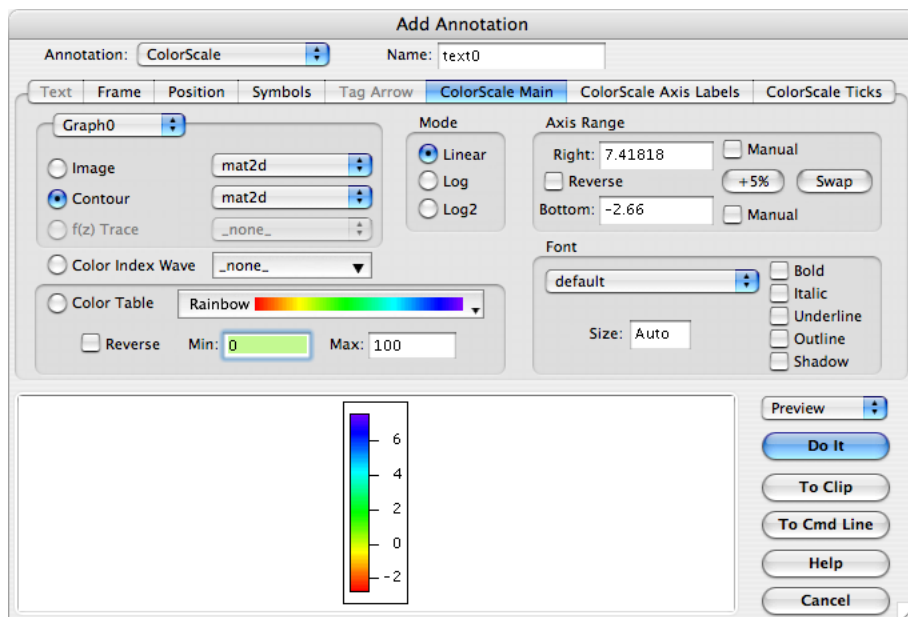
A color scale is like a tag because it is associated with data, except the color scale describes the range of the data rather than one particular value.

A color scale summarizes the range of data using a color bar and one or more axes.



ColorScale Main Tab

A color scale is associated with an $f(z)$ trace, image plot, contour plot in a graph, or with any color index wave or color table. This association can be changed in the ColorScale Main tab.

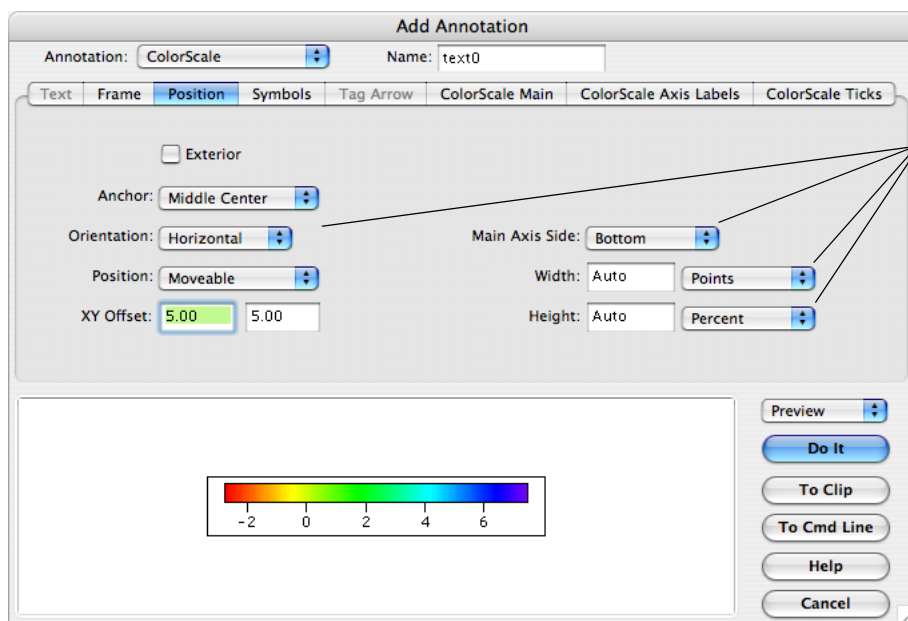


You can infer from the Graph pop-up menu that color scales can be associated with image and contour plots and $f(z)$ traces in a graph other than the graph (or layout) in which the color scale is displayed.

Many of the color scale settings are similar to graph axis settings, such as the Linear/Log/Log2 setting and the default font settings in this ColorScale Main tab.

ColorScale Size and Orientation

The size and orientation of the color scale is set in the dialog's Position tab:



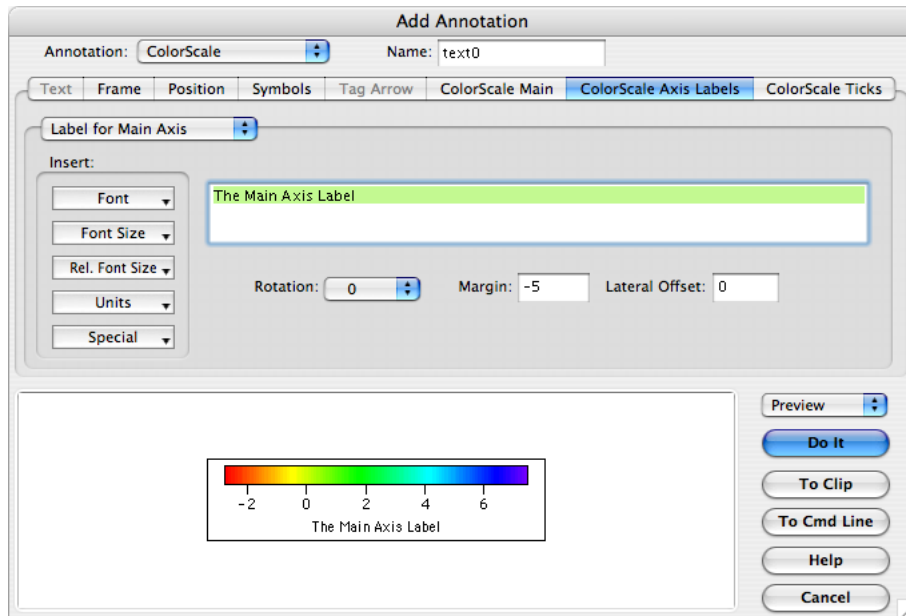
The size of a color scale is indirectly controlled by the size and orientation of the “color bar” inside the annotation, and by the various axis and ticks parameters. The annotation adjusts itself to accommodate the color bar, tick labels, and axis label(s).

When set to “Auto” or “0” the Width and Height settings cause the color scale to auto-size with the graph along the color scale’s axis dimension. Horizontal color scales auto-size horizontally but not vertically, and vice versa. The long dimension of the color bar is maintained at 75% of the corresponding plot area dimension. The short dimension is set to 15 points.

You can specify custom setting of either scale dimension. Choosing Percent from the menu resizes the corresponding dimension in response to graph size changes. Points fixes the dimension so that it never changes.

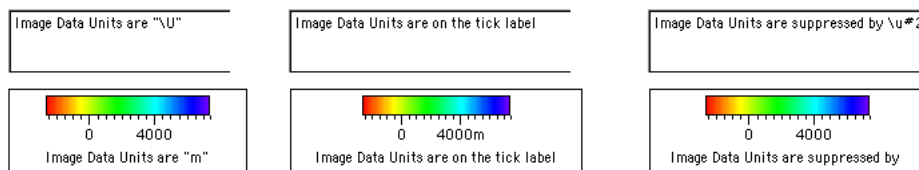
ColorScale Axis Labels Tab

The axis label for the main axis and the second axis (if any) is set in the ColorScale Axis Labels tab:



The axis label text is limited to one line. This text is the same as is used for text boxes, legends, and tags in the Text tab, but it is truncated to one line when the Annotation pop-up menu is changed to ColorScale.

The Units pop-up menu inserts escape codes related to the data units of the item the color scale is associated with. In the case of an image or contour plot, the codes relate to the data units of the image or contour matrix, or of an XYZ contour’s Z data wave. See **Changing Dimension and Data Scaling** on page II-83. These codes work as they do in the Modify Axis dialog. For example, inserting the code “\U” adds the data units to the axis label, and “\u#2” removes the data units from the axis:



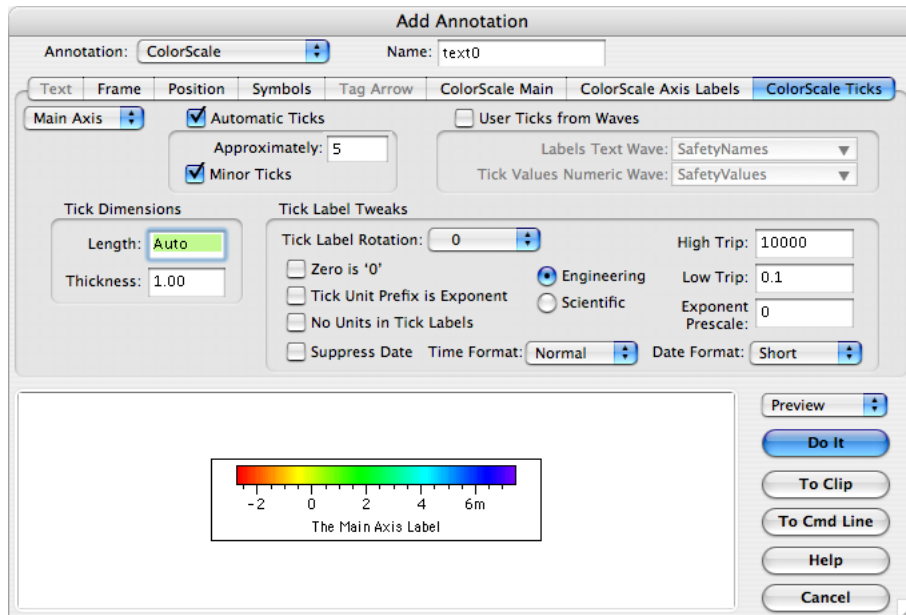
Rotation, Margin, and Lateral Offset adjust the axis label’s orientation and position relative to the color axis.

The second axis label is enabled only if the Color Scale Ticks tab has created a second axis through user-supplied tick value and label waves.

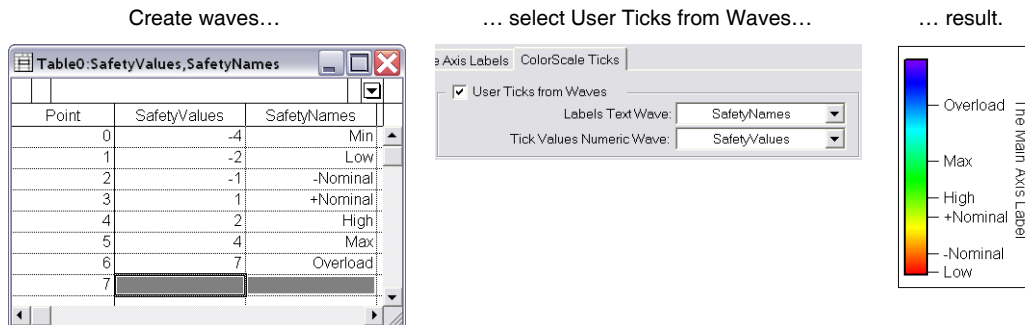
Chapter III-2 — Annotations

ColorScale Ticks Tab

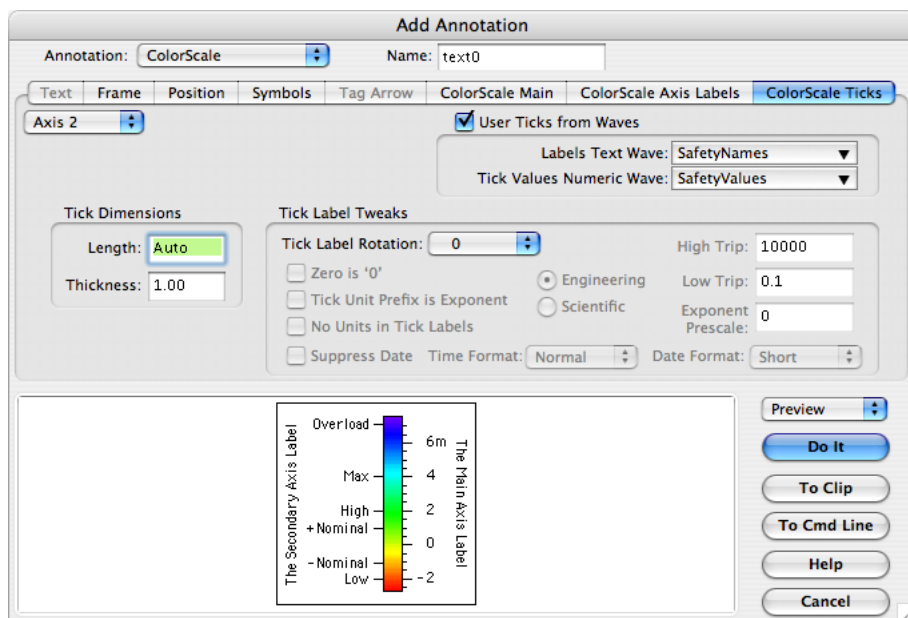
The color scale's axis ticks settings are similar to those for a graph axis:



The main axis tick marks can be automatically computed by selecting Automatic Ticks, or you can supply two waves (one numeric with the tick positions, and one text wave with the corresponding tick labels):



You can also specify user-defined tick values and labels to create a second axis, which might be useful to display a temperature range in Fahrenheit and in Celsius or a distance in feet and meters. The second axis is drawn on the opposite side of the color bar from the main axis:



The Tick Dimensions are common to both axes. The length of -1 means Auto (0 means 0), otherwise the dimensions are in points. To eliminate tick marks (but not the labels), set the Thickness (not the Length) to 0.

Elaborate Annotations and Axis Labels

For the purposes of this discussion we will use the term “annotation” to include textboxes, tags, legends *and axis labels*.

You control the nuances of text in annotations by embedding escape sequences in the text. You can insert these escape sequences by choosing an item from a pop-up menu in the Insert group in the Label Axis and Add Annotation dialogs or by directly typing them into the text. When you choose an item from a pop-up menu the escape sequence is inserted at the current insertion point in the annotation text or, if a range of text is selected, the escape sequence replaces the range of text.

Simple escape sequences allow you to set things like the font, font size, subscript or superscript and inserting wave symbols. Tags support sequences for inserting dynamic text. All of this simple stuff is described in the sections following **Modifying Annotations** on page III-43. The secret to creating elaborate annotations is something we call **text info variables**.

Remember, you can use text info variables in annotations *and in axis labels*.

Elaborate Annotations Versus Equation Editors

Creating simple annotations and axis labels is very easy in Igor. Creating more elaborate mathematical, scientific, or engineering annotations involves subscripts, superscripts, changes of X and Y position and of font and style.

The good news is that Igor can do all of this. The bad news is that Igor uses escape codes to do it rather than a nifty WYSIWYG equation editor.

If you create many of these elaborate annotations, you might consider using an equation editor. You can export their output as a picture and then import it into the drawing layer of a graph or into any layer of a page layout. The main disadvantages of this approach are the additional cost of the editor and the fact that the picture format may not be cross-platform or give the highest quality. If you will be exporting your graph or page layout as EPS or printing to a PostScript printer, you should seek out an editor that can export as EPS for best results in Igor.

About Text Info Variables

The **text info variable** is a mechanism that uses escape sequences that have a higher degree of “intelligence” than simple changes of font, font size, or style. Using text info variables, you can create quite elaborate annotations if you have the patience to do it. Since you need to know about them only to do fancy things, *if you are satisfied with simple annotations, skip the rest of these Text Info Variable sections.*

A text info variable saves information about a particular “spot” (text insertion point) in an annotation. Specifically, it saves the font, font size, style (bold, outlined, etc.), and horizontal and vertical positions of the spot. Each annotation has 10 text info variables, numbered 0 through 9. You can embed an escape sequence in an annotation’s text to store information about the insertion point in a particular variable. Later, you can embed an escape sequence to recall part or all of that information. In the Label Axis and Add Annotation dialogs, there are items in the Font, Font Size and Special pop-up menus to do this.

Simple Text Info Variables Example

To get a feel for this, let’s look at a simple example. We want to create a textbox that says:

$x = A \cos(\omega t)$

To do this, we need to switch to the Symbol font to do the omega. Then we want to switch back to the normal font, whatever that was, to finish the annotation. We could do this without a text info variable as follows:

```
Macintosh:  x = A cos (\F'Symbol'w\F'Geneva't)
Windows:    x = A cos (\F'Symbol'w\F'Arial't)
```

Here the escape sequence `\F'Symbol'` sets the font to Symbol and `\F'Geneva'` sets it back to Geneva (`\F'Arial'` sets it back to Arial). We have assumed that the default font is Geneva (or Arial). Using a text info variable we can accomplish the same thing without making that assumption. The text to do this is:

```
\[1x = A cos (\F'Symbol'w\F]1t)
```

The `\[1` is an escape sequence that says “save all of the information about the current insertion point in text info variable 1”. So text info variable 1 now contains the font, font size, style, and horizontal and vertical positions of the insertion point. You can insert this escape sequence by choosing Store Info from the Special pop-up menu. (Note that one advantage is enhanced cross-platform compatibility.)

The `\F'Symbol'` escape sequence says “start using Symbol font”. You can insert this escape sequence by choosing Symbol from the Font pop-up menu (assuming you have Symbol font installed). The `w` is an omega in the Symbol font.

The `\F]1` is an escape sequence that says “restore the font to what it was when text info variable 1 was last saved”. You can insert this escape sequence by choosing “Recall font” from the Font pop-up menu.

Text Info Variables Escape Codes

At the start of the annotation, each text info variable is initialized to the default font and size for the graph or page layout the annotation is (or will be) in. The default font and font size of a *graph* is established in the Modify Graph dialog. The default font for a *page layout* is the experiment’s default font (often Geneva (*Macintosh*) or Arial (*Windows*), see the Misc menu); the default font size is 10 points. The X and Y positions are initially undefined.

Here are all of the text info variable escape codes; *digit* means one of 0, 1, 2, ... 9.

| | |
|-----------------------|---|
| <code>\[digit</code> | Saves font, size, style and current X and Y positions in text info variable. |
| <code>\]digit</code> | Restores all but XY position from text info variable. |
| <code>\Xdigit</code> | Restores X position from text info variable. |
| <code>\Ydigit</code> | Restores Y position from text info variable. X and Y positions of a variable are undefined until you store into it. |
| <code>\F]digit</code> | Restores font from text info variable. |
| <code>\Z]digit</code> | Restores font size from text info variable. |
| <code>\f]digit</code> | Restores style of type from text info variable. |

Text info variable 0 has a special property. It defines the “main” font size which is restored using the \M escape sequence. \M also sets the Y offset from the baseline to zero. No other text info variable 0 settings (font, style, or offsets) are used by \M.

Because the initial font, size and style are stored in text info variable 0, the previous example could be made even more simple:

$$x = A \cos(\text{Symbol}'w\text{F}0t)$$

Note the right bracket,], used with \F, \Z, and \f. This indicates that what follows is the number of a text info variable. The sequence \Z09 means “set the font size to 9” whereas the sequence \Z]9 means “set the font size as stored in text info variable 9”.

Elaborate Text Info Variables Example

Let’s look at an elaborate example using text info variables. We want to create a textbox that looks something like this:

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} \cos(nx) dx$$

Here is the annotation text with escape sequences to produce an approximation to this equation:

Macintosh (the integral sign is Option-b in many fonts):

```
\Z14\[0a\Bn\M = \[1\S1\X1\M\B\pi\M\X1-½\B-\[2\pi\M\X2\S\pi\M cos(nx) dx
```

Windows (the integral sign and Greek letters are in Symbol font):

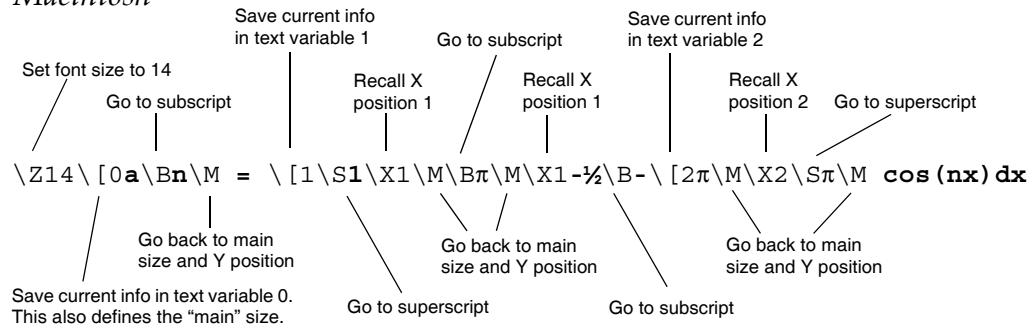
```
\Z14\[0a\Bn\M = \[1\S1\X1\M\B\F'Symbol'p\M\X1\X1-δ\B-\[2p\M\X2\S\p\M\F]0cos(nx) dx
```

This produces the following textbox:

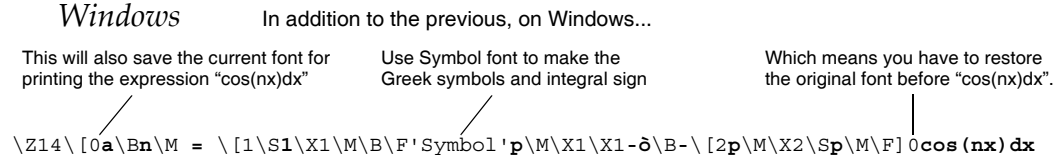
$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} \cos(nx) dx$

The escape sequences were generated using the Font, Font Size and Special items in the pop-up menus of the Add Annotation dialog. Here is an explanation of each escape sequence:

Macintosh



Windows



More Examples

$$y = \sum_{i=0}^N f(i) \quad \backslashZ14[0y = \Z20\F'Symbol'[1S\F]0\X1\Z20\B\Bi=0\M\X1\Z20\S\S\Z09 N\M f(i)$$

$$y = K_a^b + 2 \quad \backslashZ10[0y = \Z14K[1\Ba\M]1\X1\Sb\M]0 + 2$$

Programming with Annotations

You can create, modify and delete annotations with the Legend, Tag, and Textbox operations. The AnnotationInfo function returns information about one existing annotation. The **AnnotationList** function (see page V-20) returns a list of the names of existing annotations. Look at the demo experiment Tags as Markers Demo in the Examples:Techniques folder for inspiration.

Changing Annotation Names

Each annotation has a name which is unique within the graph or page layout it is in. You supply this name to the Legend, Tag, Textbox, and AnnotationInfo routines to identify the annotation you want to change. You can rename an annotation by using the `/C/N=oldName/R=newName` syntax with the operations. For example:

```
TextBox/C/N=oldTextBoxName/R=newTextBoxName
```

Changing Annotation Types

To change the type of an annotation, apply the corresponding operation to the named annotation. For example, to change a Tag or Legend into a TextBox, use:

```
TextBox/C/N=annotationName
```

Changing Annotation Text

To change the text of an existing annotation, identify the annotation using `/N=annotationName`, and supply the new text. For example, to supply new text for the textbox named text0, use:

```
TextBox/C/N=text0 "This is the new text"
```

To append text to an annotation, use the AppendText operation:

```
AppendText/N=text0 "but this text appears on a new line"
```

Generating Text Programmatically

You can write an Igor procedure to create or update an annotation using text generated from the results of an analysis or calculation. For example, here is a function that creates or updates a textbox in the top graph or layout window. The textbox is named FitResults.

```
Function CreateOrUpdateFitResults(slope, intercept)
    Variable slope, intercept

    String fitText
    sprintf fitText, "Fit results: Slope=%g, Intercept=%g", slope, intercept
    TextBox/C/N=FitResults fitText
End
```

You would call this function, possibly from another function, after executing a CurveFit command that performed a fit to a line, passing K0 as the intercept parameter and K1 as the slope parameter. K0 and K1 are outputs from the CurveFit operation.

Usually it is better to calculate text this way, using the sprintf operation, than to use dynamic text, as described in **Dynamic Escape Codes for Tags** on page III-46, because dynamic text relies on global variables that you might inadvertently delete or whose value you might inadvertently change.

Deleting Annotations

To programmatically delete an annotation, use:

```
TextBox/K/N=text0
```