Flags

/W=*winName*	Modifies the named graph window or subwindow. When omitted, action will affect the active window or subwindow. This must be the first flag specified when used in a Proc or Macro or on the command line. When identifying a subwindow with *winName*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.
/Z	Does not generate an error if the indexed trace, named wave, or named axis does not exist in a style macro.

Examples

The following code creates a graph where all the text expands and contracts directly in relation to the window size:

```
Make jack=sin(x/8);display jack
ModifyGraph mode=4,marker=8,gfRelSize= 5.0
TextBox/N=text0/A=MC "Some \\Zr200big\\]0 and \\Zr050small\\]0\rtext"
```

The *widthSpec* and *heightSpec*s set the width and height mode for the top graph. The following examples illustrate how to specify the various modes.

`ModifyGraph width=0, height=0`	Set to auto height, width mode. The width, height of horizontal and vertical axes are automatically determined based on the overall size of the graph and other factors such as axis offset setting and effect of exterior textboxes. This is the normal, default mode.
`Variable n=72*5` `ModifyGraph width=n`	Five inches as points absolute width mode, horizontal axis width constrained to `n` points.
`ModifyGraph height=n`	Absolute height mode, `n` is in points. The height of the vertical axes is constrained to `n` points.

`Variable n=2`
`ModifyGraph`
`width={perUnit,n,bottom}`
	Per unit width mode. The width of the horizontal axes is `n` points times the range of the bottom axis.

`ModifyGraph height={Aspect,n}`	Aspect height mode, `n = aspect ratio`. The height of the vertical axes is `n` times the width of the horizontal axes.
`ModifyGraph` `width={Plan,n,bottom,left}`	Plan width mode. The width of the horizontal axes is `n` times the height of the vertical axes times range of the bottom axis divided by the range of the left axis.

ModifyGraph *(traces)*

ModifyGraph [**/W=*winName*/Z**] ***key*** [**(*traceName*)**] **= *value***
    [**, *key*** [**(*traceName*)**] **= *value***]…

This section of ModifyGraph relates to modifying the appearance of wave "traces" in a graph. A trace is a representation of the data in a wave, usually connected line segments.

Parameters

Each *key* parameter may take an optional *traceName* enclosed in parentheses. Usually *traceName* is simply the name of a wave displayed in the graph, as in "`mode(myWave)=4`". If "(*traceName*)" is omitted, all traces in the graph are affected. For instance, "`ModifyGraph lSize=0.5`" sets the lines size of all traces to 0.5 points.

For multiple trace instances, *traceName* is followed by the "#" character and instance number. For example, "`mode(myWave#1)=4`". See **Instance Notation** on page IV-20.

A string containing a trace name can be used with the $ operator to specify *traceName*. For example, `String MyTrace="myWave#1"; mode($MyTrace)=4`.

Though not shown in the syntax, the optional "(*traceName*)" may be replaced with "[*traceIndex*]", where *traceIndex* is zero or a positive integer denoting the trace to be modified. "[0]" denotes the first trace appended to the graph, "[1]" denotes the second trace, etc. This syntax is used for style macros, in conjunction with the /Z flag.

For certain modes and certain properties, you can set the conditions at a specific point on a trace by appending the point number in square brackets after the trace name. For more information, see the **Customize at Point** on page V-625.

The parameter descriptions below omit the optional "(*traceName*)". When using ModifyGraph from a user-defined function, be careful not to pass wave references to ModifyGraph. ModifyGraph expects trace names, not wave references. See **Trace Name Parameters** on page IV-88 for details.

arrowMarker=0

arrowMarker={*aWave*, *lineThick*, *headLen*, *headFat*, *posMode* [, *barbSharp=b*, *barbSide=s*, *frameThick=f*]}

Draws arrows instead of conventional markers at each data point in a wave. Arrows are not clipped to the plot area and will be drawn wherever a data point is within the plot area.

aWave contains arrow information for each data point. It is a two (or more) column wave containing arrow line lengths (in points) in column 0 and angles (in radians measured counterclockwise) in column 1. Zero angle is a horizontal arrow pointing to the right. If an arrow is below the minimum length of 4 points, a default marker is drawn.

You can change arrow markers into standard meteorological wind barbs by adding a column to *aWave* and giving it a column label of windBarb. Values are integers from 0 to 40 representing wind speeds up to 4 flags. Use positive integers for clockwise barbs and negative for the reverse. Use NaN to suppress the drawing. See **Wind Barb Plots** on page II-329 for an example.

Additional columns may be supplied in *aWave* to control parameters on a point by point basis. These optional columns are specified by dimension label and not by specific column numbers. The labels are *lineThick*, *headLen*, and *headFat* that correspond to the same parameters listed above.

lineThick is the line thickness in points.

headLen is the arrow head length in points.

headFat controls the arrow fatness. It is the width of the arrow head divided by the length.

posMode specifies the arrow location relative to the data point.

posMode=0:	Start at point.
posMode=1:	Middle on point.
posMode=2:	End at point.

In addition to the wave specification, *aWave* can also be the literal `_inline_` to draw lines and arrows between points on the trace (see **Examples**). If *aWave* is `_inline_`, *posMode* values are:

posMode=0:	Arrow at start.
posMode=1:	Arrow in middle.
posMode=2:	Arrow at end.
posMode=3:	Arrow in middle pointing backwards.

You can also enable inline mode even if *aWave* is not `_inline_` by setting posMode to values between 4 and 7. These are the same as modes 0-3 above.

Optional parameters must be specified using *keyword* = *value* syntax and can only be appended after *posMode* in any order.

barbSharp is the continuously variable barb sharpness between -1.0 and 1. 0:

barbSharp=1:	No barb; lines only.
barbSharp=0:	Blunt (default).
barbSharp=-1:	Diamond.

barbSide specifies which side of the line has barbs relative to a right-facing arrow:

barbSide=0:	None.
barbSide=1:	Top.
barbSide=2:	Bottom.
barbSide=3:	Both (default).

frameThick specifies the stroke outline thickness of the arrow in points. The default is *frameThick* = 0 for solid fill.

aWave can contain columns with data for each optional parameter using matching column names.

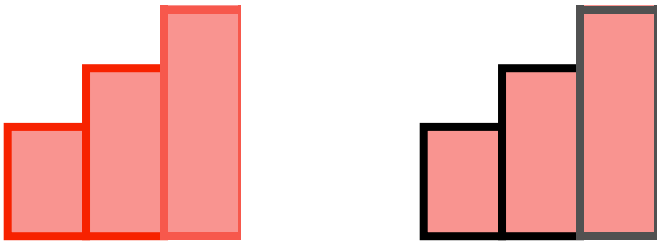
barStrokeRGB=(*r*,*g*,*b*[,*a*])

Specifies a separate color for bar strokes (outlines) if useBarStrokeRGB is 1. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is opaque black.

Applies only to Histogram Bars drawing mode (mode=5).

The bar fill color continues to be set with the rgb=(*r*,*g*,*b*[,*a*]), zColor={...}, usePlusRGB, plusRGB=(*r*,*g*,*b*[,*a*]), useNegRGB, and negRGB=(*r*,*g*,*b*[,*a*]) parameters.

Use barStrokeRGB and useBarStrokeRGB to put a differently-colored outline around Histogram Bars:



cmplxMode=*c*

Display method for complex waves.

c=0:	Default mode displays both real and imaginary parts (imaginary part offset by dx/2).
c=1:	Real part only.
c=2:	Imaginary part only.
c=3:	Magnitude.
c=4:	Phase (radians).

cmplxMode=0 does not work when the trace is a subrange of a multidimensional wave.

column=*n*

Changes the displayed column from a matrix. Out of bounds values are clipped.

gaps=*g*

Controls treatment of NaNs:

g=0:	No gaps (ignores NaNs).
g=1:	Gaps (shows NaNs as gaps).

gradient

See **Gradient Fills** on page III-498 for details.

gradientExtra

See **Gradient Fills** on page III-498 for details.

hBarNegFill=*n*

Fill kind for negative areas if useNegPat is true. *n* is the same as for the hbFill keyword.

hbFill=*n*

Sets the fill pattern.

n=0: No fill.
n=1: Erase.
n=2: Solid black.
n=3: 75% gray.
n=4: 50% gray.
n=5: 25% gray.
n>=6: See **Fill Patterns** on page III-498.

hideTrace=*h*

Removes a trace from the graph display.

h=0: Shows the trace if it is hidden.
h=1: Hides the trace and removes it from autoscale calculations.
h=2: Hides the trace.

When using *h*=1 to hide a graph trace, the hidden trace symbol and following text in annotations are also hidden. The amount of hidden text is the lesser of: the remaining text on the same line or the text up to but not including another trace symbol `"\s(traceName)"`.

lHair=*lh*

Sets the hairline factor for traces printed on a PostScript® printer.

lineJoin={*j*, *ml*}

Sets the line join style and miter limit.

Line join:

j=0 Miter joins
j=1 Round joins
j=2 Bevel joins (default)

Miter limit:

ml >= 1 Sets miter limit to *ml*
ml = INF Sets miter limit to unlimited
ml = 0 Leaves miter limit unchanged
ml = -1 Sets miter limit to default (10)

The miter limit applies only to miter joins (*j*=0) and is ignored otherwise.

See **Line Join and End Cap Styles** on page III-496 for further information.

The lineJoin keyword was added in Igor Pro 8.00.

live=*lv*

lv is a bitwise parameter defined as follows:

Bit 0: Live mode (see Live Mode below)
Bit 1: Fast line drawing (see Fast Line Drawing)

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

logZColor=*lzc*

Controls the interpretation of the zColor parameter.

lzc=0: Sets the default linearly-spaced zColors.
lzc=1: Turns on logarithmically-spaced zColors. This requires that the zWave values be greater than 0 to display correctly.

Affects trace line color only when the zColor parameter is used with a color table or color index wave - it has no effect if rgb=(*r*,*g*,*b*) parameter or zColor={...,directRGB} are used.

lOptions=cap Sets the line end cap style:

cap=0: Flat caps

cap=1: Round caps

cap=2: Square caps (default)

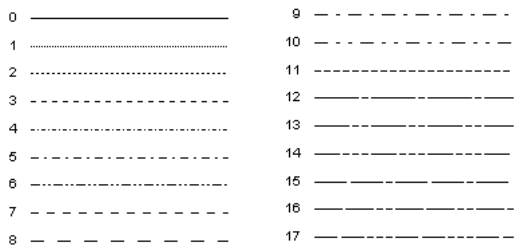
See **Line Join and End Cap Styles** on page III-496 for further information.

lSize=*l* Sets the line thickness, which can be fractional or zero, which hides the line.

lSmooth=*ls* Sets the smoothing factor for traces printed on a PostScript® printer.

lStyle=*s* Sets trace line style or dash pattern.

s=0 for solid lines. *s*=1 to s=17 for various dashed line styles.



marker=*n* n =0 to 62 designates various markers if mode=3 or 4.

You can also create custom markers. See the **SetWindow** markerHook keyword.

See **Markers** on page II-291 for a table of marker values.

mask={*maskwave*,*mode*,*value*} or 0

Specifies individual points for display by comparing values in *maskWave* with *value* as specified by *mode*.

mode=0: Exclude if equal.

mode=1: Include if equal.

mode=2: Include if bitwise AND is true.

mode=3: Include if bitwise AND is false.

maskwave can be specified using subrange notation. The length of *maskwave* (or subrange) must match the size of specified trace's wave (or subrange.) Bitwise modes should be used with integer waves with the intent of using one mask wave with multiple traces. See **Examples**.

mode=*m* Sets trace display mode.

m=0: Lines between points.

m=1: Sticks to zero.

m=2: Dots at points.

m=3: Markers.

m=4: Lines and markers.

m=5: Histogram bars.

m=6: Cityscape.

m=7: Fill to zero.

m=8: Sticks and markers.

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

Sets the background fill color for hollow markers. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

This setting takes effect only if opaque is set to non-zero. See the opaque keyword below.

The mrkFillRGB keyword was added in Igor Pro 9.00.

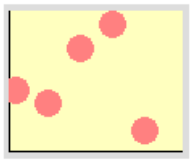
mrkStrokeRGB=(*r*,*g*,*b*[,*a*])

Specifies the color for marker stroked lines if useMrkStrokeRGB = 1. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is opaque black.

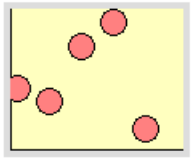
The marker fill color continues to be set with the rgb=(*r*,*g*,*b*[,*a*]) or zColor={…} parameters.

Applies only to the nontext and nonarrow marker modes.

Use mrkStrokeRGB and useMrkStrokeRGB to put a colored outline around filled markers, such as marker=19:



useMrkStrokeRGB=0



useMrkStrokeRGB=1

Note: The stroke color of unfilled markers such as marker 8 is also affected by mrkStrokeRGB, but their fill color is only affected by the opaque parameter (and the opaque fill color is always white, so if you want a color-filled marker, don't use unfilled markers).

mrkThick=*t* Sets the thickness of markers in points, which can be fractional.

msize=*m* Specifies the marker size in points.

m=0: Autosize markers.

m>0: Sets marker size.

m can be fractional, which will only make a difference when the graph is printed because fractional points can not be displayed on the screen.

mskip=*n* Puts a marker on only every *n*th data point in Lines and Markers mode (mode=4). Useful for displaying many data points when you want to identify the traces with markers. The maximum value for *n* is 32767.

mstandoff=*s* Prevents lines from touching markers in lines and markers mode if *s* is greater than zero. *s* is in units of points. This feature was added in Igor Pro 8.00.

muloffset={*mx*,*my*} Sets the display multiplier for X (*mx*) and Y (*my*). The effective value for a given X or Y data point then becomes *muloffset***data*+*offset*. A value of zero means "no multiplier" — not multiply by zero.

negRGB=(*r*,*g*,*b*[,*a*]) Specifies the color for negative values represented by the trace if useNegRGB is 1. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

offset={*x*,*y*} Sets the display offset in horizontal (X) and vertical (Y) axis units.

opaque=*o* Displays transparent (*o*=0) or opaque (*o*=1) markers.

If *o* is zero (default) hollow markers such as unfilled circles and boxes have transparent backgrounds. If *o* is non-zero, the background is filled with color. The color defaults to white, but you can change it using the mrkFillRGB keyword.

patBkgColor= 0, 1, 2 or (*r*,*g*,*b*[,*a*])

Specifies the background color for fill patterns.

0, the default, is white, 1 is graph background, 2 is transparent.

Use (*r*,*g*,*b*[,*a*]) for a specific RGB color. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

plotClip=*p* *p* =1 clips the trace by the operating system (not by Igor) to the plot rectangle. This trims overhanging markers and thick lines. On Windows, this may not be supported for certain printers or by certain applications when importing.

plusRGB=(*r*,*g*,*b*[,*a*]) Specifies the color for positive values represented by the trace if usePlusRGB is 1. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

quickdrag=*q* Controls dragging of traces.

q=0:	Normal traces.
q=1:	Traces that can be instantly dragged without the normal one second delay. See the **Quickdrag** section below.
q=2:	Causes the mouse cursor to change to 4 arrows when over the trace and a reduced search is used.

removeCustom=*r* Removes per-point trace customizations. See **Customize at Point** on page II-306 for background information. The removeCustom keyword was added in Igor Pro 9.00.

The parameter *r* is required by the ModifyGraph syntax, but its value is immaterial. We recommend using 1 for *r* but any number will work.

These examples show how to use the removeCustom keyword given a graph with a trace named wave0:

```
// Remove customizations for point 3 of trace wave0
ModifyGraph removeCustom(wave0[3]) = 1

// Remove customizations for all points of trace wave0
ModifyGraph removeCustom(wave0) = 1

// Remove customizations for all points of all traces
ModifyGraph removeCustom = 1
```

rgb=(*r*,*g*,*b*[,*a*]) Specifies the color of the trace. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

textMarker={<*char* or *wave*>,*font*,*style*,*rot*,*just*,*xOffset*,*yOffset*} or 0

Uses the specified character or text from the specified wave in place of the marker for each point in the trace.

If the first parameter is a quoted string or a string expression of the form `""+strexpr` in a user function, ModifyGraph uses the first three bytes of the string as the marker for all points. Three bytes are supported mainly for non-ASCII characters but can be used for 3 separate single-byte characters. Otherwise, it interprets the first parameter as the name of a wave. If the wave is a text wave, it uses the value of each point in the text wave as the marker for the corresponding point in the trace. If the wave is a numeric wave, the value for each point is converted into text and the result is used as the marker for the corresponding point in the trace.

xOffset and *yOffset* are offsets in fractional points. Each marker will be drawn offset from the location of the corresponding point in the trace by these amounts.

style is a font style code as used with the ModifyGraph fstyle keyword.

rot is a text rotation between -360 and 360 degrees.

just is a justification code as used in the **DrawText** operation except the X and Y codes are combined as y*4+x. Use 5 for centered.

The font size is 3*marker size. Note that marker size and color can be dynamically set via the zColor and zmrkSize keywords.

toMode=*t* Modifies the behavior of the display modes as determined by the mode parameter.

t=0: Fill to zero.

t=1: Fill to next trace. Applies to Sticks to zero (mode=1), histogram bars (mode=5), and fill to zero (mode=7).

t=2: Add the current trace's Y values to the next trace's Y values. Works with all display modes.

t=3: Stack on next and is the same as *t*=2 except that the added value is clipped to zero. Works with all display modes.

t=-1: This mode is used only with category plots and means "keep with next" (i.e., put in the same subcategory as the next trace). It is used for special effects only.

For modes 1, 2 and 3, both Y-waves must have the same number of points and must use the same X values. Igor uses the X values from the first wave for both Y-

traceName=*name* Sets, changes, or removes a custom trace name.

The traceName keyword was added in Igor Pro 9.00.

By default Igor assigns trace names based on the name of the displayed wave. You can assign a custom trace name when appending a wave to a graph using the /TN flag with the Display and AppendToGraph operations. See **Trace Names** on page II-282 for details.

The traceName keyword allows you to change the name of a trace after it has been added to a graph. For example:

```
Make/O wave0; Display wave0
ModifyGraph traceName(wave0)=Custom0
```

Use $"" for name to remove the custom name and revert to the default trace name:

```
ModifyGraph traceName(Custom0)=$""
```

Renaming a trace can change the names of other traces in the graph from the same wave. For example, if you display two instances of wave0 in one graph, the first trace name is wave0 and the second is wave0#1. If you now rename trace wave0, the name of the second instance of wave0 changes from wave0#1 to wave0. See **Instance Notation** on page IV-20 for further discussion.

useBarStrokeRGB=*u*

If *u*=1 then bar stroked lines use the color specified by the barStrokeRGB keyword.

Applies only to Histogram Bars drawing mode (mode=5).

The bar fill color continues to be set with the rgb, zColor, usePlusRGB, plusRGB, useNegRGB, and negRGB keywords.

If *u*=0 then the bar stroked line colors are set with the rgb=(*r*,*g*,*b*[,*a*]) or zColor={...} parameters, just like the bar fill color.

useMrkStrokeRGB=*u*

If *u* =1 then marker stroked lines use the color specified by the mrkStrokeRGB keyword. The marker fill color continues to be set with the rgb=(*r*,*g*,*b*[,*a*]) or zColor={…} parameters.

Applies only to the nontext and nonarrow marker modes.

If *u*=0 then the marker stroked line colors are set with the rgb=(*r*,*g*,*b*[,*a*]) or zColor={…} parameters, just like the marker fill color.

useNegPat=*u* If *u*=1, negative fills use the mode specified by the hBarNegFill keyword. Applies to the fill-to-zero, fill-to-next and histogram bar modes.

useNegRGB=*u* If *u* =1, negative fills use the color specified by the negRGB keyword. Applies to the fill-to-zero, fill-to-next and histogram bar modes.

usePlusRGB=*u*

If *u* =1, positive fills use the color specified by the plusRGB keyword. Applies to the fill-to-zero, fill-to-next and histogram bar modes.

userData={*udName*, *doAppend*, *data*}

Attaches arbitrary data to a trace. You should specify a trace name (userData(<traceName>)={...}). Otherwise copies of the data will be attached to every trace, which is most likely not what you intend.

Use the GetUserData function to retrieve the data, with the trace name as the object ID.

udName: The name of your user data. Use $"" for unnamed user data.

doAppend=0: Do not append. Any pre-existing data is replaced.

doAppend=1: Append the data. Data is added to the end of any pre-existing data.

data: A string expression containing the data you wish to attach to the trace.

zColor={*zWave*,*zMin*,*zMax*,*ctName* [,*reverseMode* [,*cWave*]]} or 0

Dynamically sets color based on the values in *zWave* and color table name or mode specified by *ctName*.

zWave may be a subrange expression such as `myZWave[2,9]` when *zWave* has more points than the trace, in which case `myZWave[2]` provides the Z value for the first point of the trace, and autoscaled *zMin* or *zMax* is determined over only the *zWave* subrange.

If a value in the *zWave* is NaN then a gap or missing marker will be observed. If a value is out of range it will be replaced with the nearest valid value. See also the zColorMax and zColorMin keywords.

ctName can be the name of a built-in color table such as returned by the **CTabList** function, such as Grays or Rainbow, for color table mode, `ctableRGB` for color table wave mode, `cindexRGB` for color index wave mode, or `directRGB` for direct color wave mode.

zColor for Built-in Color Table Mode

This mode uses *zWave* to select a color from a built-in color table specified by *ctName*. See **Image Color Tables** on page II-392 for details.

zWave contains values that are used to select a color from the built-in color table specified by *ctName*.

zMin is the *zWave* value that maps to the first entry in the color table. Use * for *zMin* to autoscale it to the smallest value in zWave.

zMax is the *zWave* value that maps to the last entry in the color table. Use * for *zMax* to autoscale it to the largest value in zWave.

ctName is the name of a built-in color table such as Grays or Rainbow. See the **CTabList** function for a list of built-in color tables.

Set *reverseMode* to 1 to reverse the color table lookup or to 0 to use the normal lookup. If you omit *reverseMode* or specify -1, the reverse mode is unchanged.

cWave must be omitted.

Normally the colors from the color table are linearly distributed between *zMin* and *zMax*. Use `logZColor=1` to distribute them logarithmically.

```
// Example zColor command using built-in color table
ModifyGraph zColor(data)={zWave,*,*,Rainbow}
```

zColor for Color Table Wave Mode

This mode is like Built-in Color Table except that the colors are stored in a color table wave that you have created. A color table wavey can be a 3 column RGB wave or a 4 column RGBA wave. See **Color Table Waves** on page II-399 for details.

zWave contains values that are used to select a color from the color table wave specified by *cWave*.

zMin is the *zWave* value that maps to the first entry in the color table wave. Use * for *zMin* to autoscale it to the smallest value in zWave.

zMax is the *zWave* value that maps to the last entry in the color table wave. Use * for *zMax* to autoscale it to the largest value in zWave.

ctName is `ctableRGB`.

Set *reverseMode* to 1 to reverse the color table lookup or to 0 to use the normal lookup. If you omit *reverseMode* or specify -1, the reverse mode is unchanged.

cWave is a reference to your color table wave.

Normally the colors from the color table are linearly distributed between *zMin* and *zMax*. Use `logZColor=1` to distribute them logarithmically.

Example `ctableRGB zColor` command:

```
ColorTab2Wave Rainbow   // Creates M_Colors wave
Rename M_Colors, MyColorTableWave
ModifyGraph zColor(data)={zWave,*,*,ctableRGB,0,MyColorTableWave}
```

zColor for Color Index Wave Mode

This mode is like Color Table Wave except that the values in *zWave* represent X indices with respect to *cWave*. You must create the RGB or RGBA color index wave and set its X scaling appropriately. See **Color Index Wave** on page II-372 for details.

zWave contains values that are used to select a color from the color index wave specified by *cWave*.

zMin and *zMax* are not used and should be set to `*`.

ctName is `cindexRGB`.

Set *reverseMode* to 1 to reverse the color table lookup or to 0 to use the normal lookup. If you omit *reverseMode* or specify -1, the reverse mode is unchanged. Normally the zWave values select the color from the row of *cWave* whose X value is closest to the zWave value. *reverseMode*=1 reverses the colors.

cWave is a reference to your color index wave.

Normally the colors from the color index wave are linearly distributed between the minimum and maximum X values of the color index wave. Use `logZColor`=1 to distribute them logarithmically.

```
// Example cindexRGB zColor command
zColor(data)={myZWave,*,*,cindexRGB,0,M_colors}
// M_colors is generated by ColorTab2Wave
```

zColor for Direct Color Wave Mode

In direct color mode, *zWave* is an RGB or RGBA wave that directly specifies the color for each point in the trace. If *zWave* is 8-bit unsigned integer, then color component values range from 0 to 255. For other numeric types, color component values range from 0 to 65535. See **ColorTab2Wave**, which generates RGB waves, and **Direct Color Details** on page II-401.

zWave is an RGB or RGBA wave that directly specifies the color for each point of the trace.

zMin and *zMax* are not used and should be set to `*`.

ctName is `directRGB`.

reverseMode is not applicable and should be omitted or set to 0.

cWave must be omitted.

```
// Example directRGB zColor command
zColor(data)={zWaveRGB,*,*,directRGB}
```

Turning zColor Off

zColor = 0 turns the zColor modes off.

zColorMax=(*red*, *green*, *blue*)

Sets the color of the trace for zColor={*zWave*, …} values greater than the zColor's *zMax*. Also turns on zColorMax mode.

The *red*, *green*, and *blue* color values are in the range of 0 to 65535.

zColorMax=1, 0, or NaN

Turns zColorMax mode off, on, or transparent. These modes affect the color of zColor={*zWave*, …} values greater than the zColor's *zMax*.

1:	Turns on zColorMax mode. The color of the affected trace pixels is black or the last color set by zColorMax=(*red*, *green*, *blue*).
0:	Turns off zColorMax mode (default). The color of the affected trace pixels is the last color in the zColor's *ctname* color table.
NaN:	Transparent zColorMax mode. Affected trace pixels are not drawn.

zColorMin=(*red*, *green*, *blue*)

Sets the color of the trace for zColor={*zWave*, …} values less than the zColor's *zMin*. Also turns zColorMin mode on.

The *red*, *green*, and *blue* color values are in the range of 0 to 65535.

zColorMin=1, 0, or NaN

Turns zColorMin mode off, on, or transparent. These modes affect the color of zColor={*zWave*, …} values less than the zColor's *zMin*.

1:	Turns on zColorMin mode. The color of the affected image pixels is black or the last color set by zColorMin=(*red*, *green*, *blue*).
0:	Turns off zColorMin mode (default). The color of the affected trace pixels is the first color in the zColor's *ctname* color table.
NaN:	Transparent zColorMin mode. Affected trace pixels are not drawn.

zmrkNum={*zWave*} or 0

Dynamically sets the marker number for each point to the corresponding value in *zWave*. The values in *zWave* are the marker numbers (as used with the marker keyword). If a value in the *zWave* is NaN then no marker will be drawn at the corresponding point. If a value is out of range it will be replaced with the nearest valid value.

zmrkNum=0 turns this mode off.

zmrkSize={*zWave*,*zMin*,*zMax*,*mrkMin*,*mrkMax*,[*axis*]} or 0

Dynamically sets marker size based on values in *zWave*. See **Marker Size as f(z)** on page II-299 for background information.

zmrkSize = 0 turns this mode off.

Use * or a missing parameter for *zMin* and *zMax* to autoscale. *mrkMin* and *mrkMax* can be fractional.

mrkMin is the marker size to use when z equals *zMin* and *mrkMax* is the marker size to use when z equals *zMax*. *mrkMin* and *mrkMax* are not limits; they just control the linear mapping of z values to marker size.

If a value in the *zWave* is NaN then the corresponding marker is not drawn.

The marker size is clipped to 400 on the high end and to 1 point on the low end.

axis is an optional parameter specifying an existing axis on the graph. It is supported in Igor Pro 9.00 and later. Passing $"" for *axis* is the same as omitting it.

If *axis* is specified, the values in *zWave* are interpreted as in units of the specified axis and specify the half-width of the marker to be drawn. For a circle marker, this is the radius of the marker.

If *axis* is specified, the *zMin*, *zMax*, *mrkMin*, and *mrkMax* parameters have no impact on the marker size unless you remove the specified axis in which case those parameters control the trace's marker sizes.

See **Marker Size as f(z) in Axis Units** on page II-300 for further information.

zpatNum={*zWave*} or 0

Dynamically sets the positive fill type/pattern number for each point to the corresponding value in *zWave*. The values in *zWave* are the pattern numbers (as used with the hbFill keyword). If a value in the *zWave* is NaN then the corresponding point will not be drawn. If a value is out of range it will be replaced with the nearest valid value.

zpatNum=0 turns this mode off.

Flags

/W=*winName* Modifies the named graph window or subwindow. When omitted, action will affect the active window or subwindow. This must be the first flag specified when used in a Proc or Macro or on the Command Line.

When identifying a subwindow with *winName*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.

/Z Does not generate an error if the indexed trace, named wave, or named axis does not exist in a style macro.

Details

Waves supplied with zmrkSize, zmrkNum, and zColor may use **Subrange Display Syntax** on page II-321.

Live Mode

Bit 0 of live=*lv* controls live mode. Live mode improves graph update performance when one or more of the waves displayed in the graph is frequently modified, for example, if the waves are being acquired from a data acquisition system. Live Mode traces do not autoscale axes.

Fast Line Drawing

Bit 1 of live=*lv* controls fast line drawing. When enabled, Igor draws traces in lines-between-points mode using custom fast code instead of the normal system drawing code. The result is not as esthetically pleasing but can be much faster and may be critical in data acquisition applications. The custom code is used only for drawing to the screen, not for exporting graphics.

Fast line drawing was added in Igor Pro 8.00.

Quickdrag

Quick drag mode (quickdrag=1) is a special purpose mode for creating cross hair cursors using a package of Igor procedures. (See the Cross Hair Demo example experiment.) Normally you would have to click and hold on a trace for one second before entering drag mode. When quickdrag is in effect, there is no delay. If a trace is in quickdrag mode it should also be set to live mode. With this combination you can click a trace and immediately drag it to a new XY offset. In addition to quick drag mode, the cross hair package relies on Igor to store information about the drag in a string variable if certain conditions are in effect. The string variable name (that you have to create) is S_TraceOffsetInfo, which must reside in a data folder that has the same name as the graph (not title!) which in turn must reside in root:WinGlobals:. If these conditions are met, then after a trace is dragged, information will be stored in the string using the following key-value format: `GRAPH:<name of graph>;XOFFSET:<x offset value>;YOFFSET:<y offset value>;TNAME:<trace name>;`

Customize at Point

You can customize the appearance of individual points on a trace in a graph for bar, marker, dot and lines to zero modes using `key(tracename[pnt])=value` syntax. The point number must be a literal number and the trace name is required.

To remove a customization, use `key(tracename[-pnt-1])=value` where value is not important but must match the syntax for the keyword. The offset of -1 is needed because point numbers start from zero.. You can also remove customizations using the removeCustom keyword described above.

Although the syntax is allowed for all trace modifiers, it has meaning only for the following: rgb, marker, msize, mrkThick, opaque, mrkFillRGB, mrkStrokeRGB, barStrokeRGB, hbFill, patBkgColor and lSize.

Note that useBarStrokeRGB and useMrkStrokeRGB are not needed. The act of using barStrokeRGB or mrkStrokeRGB is enough to customize the point. But as a convenience, since these are generated by the modify graph dialog, they are ignored if used with [pnt] syntax.

Also note that legend symbols can use [pnt] syntax like so:

```
\s(<tracename>[pnt])
```

Automatically generated legends automatically include symbols for customized points.

For example:

```
Make/O/N=10 jack=sin(x); Display jack
ModifyGraph mode=5,hbFill=6,rgb=(0,0,0)
ModifyGraph hbFill(jack[2])=7,rgb(jack[2])=(0,65535,0)
ModifyGraph rgb(jack[3])=(65535,0,0)
Legend/C/N=text1/F=0/A=MC
```

Examples

Arrow markers.

```
Make/N=10 wave1= x; Display wave1
Make/N=(10,2) awave
awave[][0]= p*5               // length
awave[][1]= pi*p/9            // angle
ModifyGraph mode=3,arrowMarker(wave1)={awave,1,10,0.3,0}

// Now add an optional column to control headLen
Redimension/N=(-1,3) awave
awave[][2]= 7+p               // will be head length

// Note: nothing changes until the following is executed
SetDimLabel 1,2,headLen,awave
```

Create meteorological wind barb symbols.

```
Make/O/N=50 jack= floor(x/10),jackx= mod(x,10)
Display jack vs jackx
Make/O/N=(50,3) jackbarb
```