```
Variable/G newVariable=1            // Do work in the new data folder
SetDataFolder savedDF               // Restore current data folder
```

See Also

Chapter II-8, **Data Folders** and **Data Folder References** on page IV-78.

SetDimLabel

SetDimLabel ***dimNumber*, *dimIndex*, *label*, *wavelist***

The SetDimLabel operation sets the dimension label or dimension element label to the specified label.

Parameters

Use *dimNumber*=0 for rows, 1 for columns, 2 for layers and 3 for chunks.

If *dimIndex* is -1, it sets the label for the entire dimension. For dimIndex ≥ 0, it sets the dimension label for that element of the dimension.

label is a name (e.g., time), not a string (e.g., "time").

label is limited to 255 bytes. If you use more than 31 bytes, your wave will require Igor Pro 8.00 or later.

Details

Dimension labels can contain up to 255 bytes and may contain spaces and other normally-illegal characters allowed in liberal names if you surround the name in single quotes or if you use the $ operator to convert a string expression to a name.

Dimension labels have the same characteristics as object names. See **Object Names** on page III-501 for a discussion of object names in general.

Prior to Igor Pro 9.00, Igor allowed you to create dimension labels containing illegal characters (double-quotes, single-quotes, colon, semicolon, and control characters). This is now flagged as an error. Existing experiments containing dimension labels with illegal names can still be loaded without error.

See Also

GetDimLabel, **FindDimLabel**, **CopyDimLabels**

Dimension Labels on page II-93 and **Example: Wave Assignment and Indexing Using Labels** on page II-82 for further usage details and examples.

SetDrawEnv

SetDrawEnv [**/W=*winName***] ***keyword*** [**=*value***][**, *keyword*** [**=*value***]]…

The SetDrawEnv operation sets properties of the drawing environment.

If one or more draw objects are selected in the top window then the SetDrawEnv command will apply only to those objects.

If no objects are selected *and* if the keyword save *is not* used *then* the command applies only to the next object drawn.

If no objects are selected *and* if the keyword "save" *is* used *then* the command sets the environment for all following objects.

Each draw layer has its own draw environment settings.

Parameters

SetDrawEnv can accept multiple *keyword=value* parameters on one line.

arrow=*arr*	Specifies the arrow head position on lines.
	arr=0: No arrowhead (default).
	arr=1: Arrowhead at end.
	arr=2: Arrowhead at start.
	arr=3: Arrowhead at start and end.
arrowfat=*afat*	Sets ratio of arrowhead width to length (default is 0.5).
arrowlen=*alen*	Sets length of arrowhead in points (default is 10).

arrowSharp=*s* Specifies the continuously variable barb sharpness between -1.0 and 1. 0.

s=1: No barb; lines only.

s=0: Blunt (default).

s=-1: Diamond.

arrowframe=*f* Specifies the stroke outline thickness of the arrow in points (default is *f*=0 for solid fill).

astyle=s Specifies which side of the line has barbs relative to a right-facing arrow.

s=0: None.

s=1: Top.

s=2: Bottom.

s=3: Both (default).

clipRect=(*left*, *top*, *right*, *bottom*)

Clips drawing to the specifed rectangle. The clipRect keyword was added in Igor Pro 8.00.

left, *top*, *right*, and *bottom* are specified in the current coordinate system of the drawing environment.

If a save is in effect, you can use values of all 0 to cancel the clipping or, better, you can a use push, clipRect, save, drawing, pop sequence. When a save is in effect, the clipping is set once and remains unchanged for the remaining objects even if the coordinate system is changed.

dash=*dsh* *dsh* is a dash pattern number between 0 and 17 (see **SetDashPattern** for patterns). 0 (solid line) is the default.

fillbgc=(*r*,*g*,*b*[,*a*]) Specifies fill background color. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is the window's background color.

fillfgc=(*r*,*g*,*b*[,*a*]) Specifies fill foreground color. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is opaque white.

fillpat=*fpatt* Specifies fill pattern density.

fpatt=-1: Erase to background color.

fpatt=0: No fill.

fpatt=1: 100% (solid pattern, default).

fpatt=2: 75% gray.

fpatt=3: 50% gray.

fpatt=4: 25% gray.

fillRule=*fr* Determines how polygons and Bezier curves with intersecting edges are filled:

fr=0: Winding rule (default)

fr=1: Even-odd rule

The fillRule keyword applies only to polygons created with **DrawPoly** and **DrawBezier**, not to those created manually.

See **Polygon and Bezier Curve Fill Rules** on page III-72 for a discussion of the rules used for filling polygons and Bezier curves with intersecting edges.

The fillRule keyword was added in Igor Pro 9.00.

fname="*fontName*" Sets font name, default is the default font or the graph font.

fsize=*size* Sets text size, default is 12 points.

fstyle=*fs*

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

Bit 0:	Bold
Bit 1:	Italic
Bit 2:	Underline
Bit 4:	Strikethrough

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

gedit= *flag*

Supplies optional edit flag for a group of objects. Use with gstart.

flag=0:	Select entire group, moveable (default).
flag=1:	Individual components editable as if not grouped. Allows objects to be grouped by name but still be editable.

gname= *name*

Supplies optional *name* for an object group. Use with gstart.

gstart

Marks the start of a group of objects.

gstop

Marks the end of a group of objects.

gradient

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

gradientExtra

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

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

Specifies line background color. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is the window's background color.

linecap=*cap*

Sets the line end cap style:

cap=0:	Flat caps (default)
cap=1:	Round caps
cap=2:	Square caps

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

linecap was added in Igor Pro 8.00.

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

Specifies line foreground color. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is opaque black.

linejoin=*j*

Sets the line join style:

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

For a miter join, you can also set the miter limit using the lineMiterLimit keyword.

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

linejoin was added in Igor Pro 8.00.

lineMiterLimit=*ml*

Applies only to a miter line join style. See the linejoin keyword.

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)

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

lineMiterLimit was added in Igor Pro 8.00.

linethick=*thick*

thick is a line thickness ≥ 0, default is 1 point.

origin= *x0,y0*
: Moves coordinate system origin to *x0,y0*. Unlike translate, rotate, and scale, this survives a change in coordinate system and is most useful that way. See **Coordinate Transformation**.

pop
: Pops a draw environment from the stack. Pops should always match pushes.

push
: Pushes the current draw environment onto a stack (limited to 10).

rotate= *deg*
: Rotates coordinate system by *deg* degrees. Only makes sense if X and Y coordinate systems are the same. See **Coordinate Transformation**.

rounding=*rnd*
: Radius for rounded rectangles in points, default is 10.

rsabout
: Redefines coordinate system rotation or scaling to occur at the translation point instead of the current origin. To use, combine rotate or scale with translate and rsabout parameters.

save
: Stores the current drawing environment as the default environment.

scale= *sx,sy*
: Scales coordinate system by *sx* and *sy*. Affects only coordinates — not line thickness or arrow head sizes. See **Coordinate Transformation**.

subpaths=*sp*
: Controls the way polygon and Bezier curve segments separated by NaN values are drawn.:

  sp=0: Each segment is drawn as a separate polygon or Bezier curve, and any arrows are added to each segment as if they are separate polygons or Bezier curves. This is the default if you omit the subpaths keyword.

  sp=1: The segments are treated as subpaths within a single polygon or Bezier curve, making it possible to define a shape with holes. Any arrows are added only to the first or last points in the entire shape.

  The subpaths keyword applies only to polygons created with **DrawPoly** and **DrawBezier**, not to those created manually.

  Also see the fillRule keyword.

  The subpaths keyword was added in Igor Pro 9.00.

textrgb=(*r,g,b*[,*a*])
: Specifies text color. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is opaque black.

textrot=*rot*
: Text rotation in degrees.

  rot is a value from -360 to 360.

  0 is normal (default) horizontal left-to-right text,

  90 is vertical bottom-to-top text, etc.

textxjust=*xj*
: Sets horizontal text alignment.

  xj=0: Left aligned text (default).

  xj=1: Center aligned text.

  xj=2: Right aligned text.

textyjust=*yj*
: Sets vertical text alignment.

  yj=0: Bottom aligned text (default).

  yj=1: Middle aligned text.

  yj=2: Top aligned text.

translate= *dx,dy*
: Shifts coordinate system by *dx* and *dy*. Units are in the current coordinate system. See **Coordinate Transformation**.

xcoord=abs	X coordinates are absolute window coordinates (default for all windows except graphs where the default is xcoord=prel). The unit of measurement is **Control Panel Units** if the window is a panel, otherwise they are points. The left edge of the window (or of the printable area in a layout) is at x=0. See **Drawing Coordinate Systems** on page III-66 for details.
xcoord=axrel	X coordinates are relative axis rectangle coordinates (graphs only). The axrel coordinate system was added in Igor Pro 9.00. The axis rectangle is the plot rectangle expanded to include any axis standoff. x=0 is at the left edge of the rectangle; x=1 is at the right edge of the rectangle. This coordinate system ideal for objects that should maintain their size and location relative to the axes when axis standoff is used. See **Axis Relative (Graphs Only)** on page III-67 for details. You can retrieve the axis rectangle's coordinates using GetWindow axSize.
xcoord=rel	X coordinates are relative window coordinates. x=0 is at the left edge of the window; x=1 is at the right edge. See **Drawing Coordinate Systems** on page III-66 for details.
xcoord=prel	X coordinates are relative plot rectangle coordinates (graphs only). x=0 is at the left edge of the rectangle; x=1 is at the right edge of the rectangle. This coordinate system ideal for objects that should maintain their size and location relative to the axes, and is the default for graphs. See **Plot Relative (Graphs Only)** on page III-67 for details. You can retrieve the axis rectangle's coordinates using GetWindow pSize.
xcoord=*axisName*	X coordinates are in terms of the named axis (graphs only).
ycoord=abs	Y coordinates are absolute window coordinates (default for all windows except graphs where the default is ycoord=prel). The unit of measurement is **Control Panel Units** if the window is a panel, otherwise they are points. The top edge of the window (or the of the printable area in a layout) is at y=0. See **Drawing Coordinate Systems** on page III-66 for details.
xcoord=axrel	Y coordinates are relative axis rectangle coordinates (graphs only). The axrel coordinate system was added in Igor Pro 9.00. The axis rectangle is the plot rectangle expanded to include any axis standoff. y=0 is at the top edge of the rectangle; y=1 is at the bottom edge of the rectangle. This coordinate system ideal for objects that should maintain their size and location relative to the axes when axis standoff is used. See **Axis Relative (Graphs Only)** on page III-67 for details. You can retrieve the axis rectangle's coordinates using GetWindow axSize.
ycoord=rel	Y coordinates are relative window coordinates. y=0 is at the top edge of the window; y=1 is at the bottom edge. See **Drawing Coordinate Systems** on page III-66 for details.
ycoord=prel	Y coordinates are relative plot rectangle coordinates (graphs only). y=0 is at the top edge of the rectangle; y=1 is at the bottom edge of the rectangle. This coordinate system ideal for objects that should maintain their size and location relative to the axes, and is the default for graphs. See **Plot Relative (Graphs Only)** on page III-67 for details. You can retrieve the axis rectangle's coordinates using GetWindow pSize.
ycoord=*axisName*	Y coordinates are in terms of the named axis (graphs only).

Flags

/W=*winName* Sets the named window or subwindow for drawing. 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.

Coordinate Transformation

The execution order for the translate, rotate, scale, and origin parameters is important. Translation followed by rotation is different than rotation followed by translation. When using multiple keywords in one SetDrawEnv operation, the order in which they are applied is origin, translate, rotate followed by scale regardless of the command order (with the exception of the rsabout parameter). Before using origin with the save keyword, you should use push to save the current draw environment and then use pop after drawing objects using the new origin.

Examples

Following is a simple example of arrow markers:

```
NewPanel
SetDrawEnv arrow= 1,arrowlen= 30,save
SetDrawEnv arrowsharp= 0.3
DrawLine 61,67,177,31
SetDrawEnv arrowsharp= 1
DrawLine 65,95,181,59
SetDrawEnv astyle= 1
DrawLine 69,123,185,87
SetDrawEnv arrowframe= 1
DrawLine 73,151,189,115
```

You can position objects in one coordinate system and then draw them in another with the origin keyword. In the following coordinate transformation example, we position arrows in axis units but size them in absolute units.

```
Make/O jack=sin(x/8)
Display jack
SetDrawEnv xcoord=bottom,ycoord=left,save
SetDrawEnv push
SetDrawEnv origin=50,0
SetDrawEnv xcoord=abs,ycoord=abs,arrow=1,arrowlen=20,arrowsharp=0.2,save
DrawLine 0,0,50,0          // arrow 50 points long pointing to the right
DrawLine 0,0,0,50          // arrow 50 points long pointing down
// now let's move over, rotate a bit and draw the same arrows:
SetDrawEnv translate=100,0
SetDrawEnv rotate=30,save
DrawLine 0,0,50,0
DrawLine 0,0,0,50
SetDrawEnv pop
```

Now try zooming in on the graph. You will see that the first pair of arrows always starts at 50 on the bottom axis and 0 on the left axis whereas the second pair is 100 points to the right of the first.

See Also

Chapter III-3, **Drawing**, and **DrawAction**.