Graph Library Reference

AutoGraph (AG) Functions
Standard (GS) Functions
Label Formats
Parameter Constants

AutoGraph Functions

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Function	Description
AG3DStyle	Sets style of 3D graph
A	
AGAmp	Transfers amplitude data
AGAmpError	Transfers amplitude error data
AGAxisMinorTicks	Sets the number of minor ticks for each axis
AGAux	Transfers auxiliary information
B	
AGBar2DGap	Adjusts the gap between bars in 2D bar graphs
C	
AGCageStyle	Sets 3D cage style
AGClose	Terminates AutoGraph
AGClr	Transfers color information
AGCurveStyle	Sets curve style
D	
AGDataLabels	Enables and sets text for data labels
VBAGDataLabels	Enables and sets text for data labels in Visual Basic
AGDataRange	Sets a range of data to be graphed
AGDataZ	Transfers Z data

AGDist	Transfers distance data
AGDistError	Transfers distance error data

E

AGErrorBar	Defines error bars for graph

F

AGFFT	Performs Fast Fourier Transform on array of data
AGFGColor	Sets color of foreground objects
AGFontStyle	Sets font style

G

AGGraphBG	Sets graph background
AGGridStyle	Sets style of grid lines

I

AGInfo	Gets AutoGraph drawing information

L

AGLabelFormat	Defines a format template for axis and data labels
AGLabelDateTime	Defines date/time labels for an axis
AGLabels	Defines labels for axis or pie chart
VBAGLabels	Defines labels for axis or pie chart in Visual Basic
AGLabelY	Defines labels for left or right Y axis
VBAGLabelY	Defines labels for left or right Y axis in Visual Basic
AGLabelZ	Defines labels for Z axis
VBAGLabelZ	Defines labels for Z axis in Visual Basic
AGLegend	Defines legend labels for grouped data
VBAGLegend	Defines legend labels for grouped data in Visual Basic
AGLegendStyle	Sets position and style of legend
AGLimitLines	Applies limit lines to a graph

M

AGMissingLineStyle	Selects options for bridging gaps caused by missing data

O

AGOpen	Initializes AutoGraph

P

AGPatt	Transfers pattern information

R

AGRefresh3D	Redraws True3D graph
AGReset	Resets all AutoGraph functions

S

AGSetPerspective	Sets perspective view of True3D graphs
AGShow	Shows a graph
AGSurfaceClr	Sets colors for True3D surface graph
AGSym	Transfers symbol information

T

AGTimeGraph	Begins time series graph
AGTimeUpdate	Updates time series graph with data
AGTitleBG	Sets title style and background color

AGTitleG	Defines graph title
AGTitleX	Defines bottom title for graph
AGTitleY	Defines left title for graph
AGTitleYR	Defines right title for graph
AGTrendDataSet	Applies trend lines to individual data sets

X

AGXAxisStyle	Sets X axis style

Y

AGYAxisStyle	Sets Y axis style
AGYRAxisStyle	Sets right-hand Y axis style

Z

AGZAxisStyle	Sets Z axis style

Standard Functions

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Function	Description
A	
GSArc	Draws circular arc
GSArea	Draws 2D area graph
GSArea3D	Draws 3D area graph
GSAreaLogLin	Draws a 2D area graph with semi-log scaling
GSArrow	Draws arrow
GSAxis	Draws X or Y axis
B	
GSBar2D	Draws 2D bar graph
GSBar3D	Draws 3D bar graph
GSBox2D	Draws box and fills with pattern
GSBox3D	Draws 3D box and fills with pattern
GSBoxWhisker	Draws box-whisker graph
GSBubbleChart	Draws bubble graph
C	
GSCage3D	Draws 3D cage with axes and grids
GSCircle	Draws circle
GSClearView	Clears view
GSClipRead	Reads image from Clipboard

GSClipWrite	Writes image to Clipboard
GSCloseServer	Closes connection to Graphics Server
GSClosePrn	Closes printer
GSCloseView	Closes view
GSCloseWin	Closes graphing window
GSCurveFit	Fits curve to data

D

GSDataAmp	Transfers array of amplitude data
GSDataAmpErr	Transfers array of amplitude error data
GSDataAux	Transfers array of auxiliary data
GSDataClr	Transfers array of color data
GSDataDim	Sets graph data dimensions
GSDataDist	Transfers array of distance data
GSDataDistErr	Transfers array of distance error data
GSDataGetAmp	Gets amplitude data value
GSDataGetAmpErr	Gets amplitude error data value
GSDataGetAux	Gets auxiliary data value
GSDataGetClr	Gets color data value
GSDataGetDist	Gets distance data value
GSDataGetDistErr	Gets distance error data value
GSDataGetPatt	Gets pattern data value
GSDataGetSym	Gets symbol data value
GSDataGetZ	Gets Z data value
GSDataLabels	Enables and sets text for data labels
VBGSDataLabels	Enables and sets text for data labels in Visual Basic
GSDataPatt	Transfers array of pattern data
GSDataRange	Defines range of data to graph
GSDataReset	Resets data arrays
GSDataScale	Applies scale factor to data
GSDataStoreAmp	Stores amplitude data value
GSDataStoreAmpErr	Stores amplitude error value pair
GSDataStoreAux	Stores auxiliary data value
GSDataStoreClr	Stores color data value
GSDataStoreDist	Stores distance data value
GSDataStoreDistErr	Stores distance error value pair
GSDataStorePatt	Stores pattern data value
GSDataStoreSym	Stores symbol data value
GSDataStoreZ	Stores Z data value
GSDataSym	Transfers array of symbol data
GSDataTrans	Transfers data in arrays
GSDataZ	Transfers array of Z data
GSDefPatt	Defines bit pattern for filling

E

GSEllipse	Draws ellipse
GSErrorBar	Defines error bars for graph

F

GSFixPos	Fixes current position

G

GSGantt	Draws Gantt chart
GSGetACos	Gets arccosine
GSGetALog10	Gets antilog base 10
GSGetALogE	Gets natural antilog base e
GSGetASin	Gets arcsine
GSGetATan	Gets arctangent
GSGetAXExt	Gets anchor space X extent
GSGetAYExt	Gets anchor space Y extent
GSGetBG	Gets background color
GSGetCC	Gets linear correlation coefficient
GSGetCos	Gets cosine
GSGetCurveCoeff	Gets curve coefficient
GSGetCurX	Gets current X position
GSGetCurY	Gets current Y position
GSGetE	Gets natural exponent
GSGetLog10	Gets log base 10
GSGetLogE	Gets natural log
GSGetMax	Gets maximum amplitude data value
GSGetMean	Gets mean data value of amplitude array
GSGetMF	Gets image metafile
GSGetMin	Gets minimum amplitude data value
GSGetPI	Gets value of pi
GSGetPrnHt	Gets printer paper height
GSGetPrnWid	Gets printer paper width
GSGetRTextHt	Gets raster text height
GSGetRTextWid	Gets raster text width
GSGetSD	Gets standard deviation of data set
GSGetSFHt	Gets height of system font characters
GSGetSFWid	Gets width of system font characters
GSGetSin	Gets sine
GSGetSXExt	Gets screen X extent
GSGetSYExt	Gets screen Y extent
GSGetTan	Gets tangent
GSGetVer	Gets server or DLL version number
GSGetVXExt	Gets view X extent
GSGetVYExt	Gets view Y extent
GSGetWXExt	Gets window X extent
GSGetWYExt	Gets window Y extent
GSGrid	Draws grid lines

H

GSHLC	Draws high-low-close, open-high-low-close, or candlestick graph
GSHotGraph	Enables and disables hot graphing

L

GSLabelnPie	Draws pie chart numeric labels
GSLabelnX	Draws numeric labels along X axis
GSLabelnY	Draws numeric labels along Y axis
GSLabelPie	Draws pie chart text labels

VBGSLabelPie	Draws pie chart text labels in Visual Basic
GSLabelX	Draws text labels along X axis
VBGSLabelX	Draws text labels along X axis in Visual Basic
GSLabelY	Draws text labels along Y axis
VBGSLabelY	Draws text labels along Y axis in Visual Basic
GSLegend	Draws legend
VBGSLegend	Draws legend in Visual Basic
GSLineAbs	Draws line using absolute coordinates
GSLineFit	Fits straight line to data
GSLineRel	Draws line using relative coordinates
GSLinLog	Draws lin/log graph
GSLoadRFont	Loads raster font
GSLoadVFont	Loads vector font
GSLogAxis	Draws logarithmic axis
GSLogGrid	Draws logarithmic grid
GSLogLin	Draws log/lin graph
GSLogLog	Draws log/log graph

M

GSMClrRgn	Clears mouse hot region
GSMean	Draws mean of data set
GSMGetX	Gets mouse X position
GSMGetY	Gets mouse Y position
GSMinMax	Draws minimum and maximum of data set
GSMissingLineStyle	Selects options for bridging gaps caused by missing data
GSMMotion	Reads mouse motion indicator
GSMNotify	Enables and disables notification of mouse events
GSMovePos	Moves current position
GSMPtrOff	Turns off mouse pointer
GSMPtrOn	Turns on mouse pointer
GSMPtrType	Defines mouse pointer shape
GSMSetRgn	Defines mouse hot region
GSMStatus	Reads mouse button status

O

GSOffView	Turns off view
GSOnView	Turns on view
GSOpenChildWin	Opens graphing window as child of another window
GSOpenPrn	Opens printer for current window
GSOpenServer	Opens connection to Graphics Server
GSOpenView	Opens view
GSOpenWin	Opens graphing window

P

GSPicRead	Reads image from file
GSPicWrite	Writes image to file
GSPie2D	Draws 2D pie chart
GSPie3D	Draws 3D pie chart
GSPolar	Draws polar graph
GSPolarAxes	Draws set of polar axes

GSPolyFill	Draws polygon filled with pattern
GSPolyVec	Draws polyline figure
GSPrnOut	Prints view or window
GSPrnSetup	Sets printing area

R

GSRText	Draws raster text

S

GSScatter	Draws 2D scatter graph
GSSD	Draws standard deviation lines
GSSelectPalette	Selects extended palette with 128 entries
GSSetBG	Sets background color
GSSetPal	Sets palette
GSSetRFontFace	Sets typeface used for raster font family
GSSetROP	Sets raster operation mode
GSSetVFontFace	Sets typeface used for vector font family
GSShade	Shades bounded area
GSSizeSymbol	Defines size of all symbols
GSStatsArr	Defines data for applying statistics
GSStatsWin	Defines statistics clipping region
GSSymbol	Draws symbol

T

GSTapeGraph	Draws tape graph
GSTimeGraph	Draws scrolling time series graph
GSTimeUpdate	Updates time series graph

U

GSUseView	Uses view

V

GSViewClip	Applies a clipping window within the current view
GSVText	Draws vector text

W

GSWinHandle	Returns Windows handle of graphing window
GSWinNotify	Enables and disables notification of graphing window events
GSWinPaint	Sets graphing window painting mode
GSWriteRegionFile	Creates an image map for use in an HTML page

X

GSXDataScale	Applies scale factor to distance data
GSXYGraph	Draws line graph

AG3DStyle function

Sets style of 3D graph

C/C++

```
int AG3DStyle( int nMode, int nDepth, int nXGap, int nZGap )
```

FoxPro

```
r = AG3DStyle(nMode, nDepth, nXGap, nZGap)
```

Visual Basic

```
r% = AG3DStyle(nMode%, nDepth%, nXGap%, nZGap%)
```

Parameters

`nMode`

Constant	Value	Meaning
AG3DSETDEPTH	1	Depth parameter is present
AG3DSETXGAP	2	X gap parameter is present
AG3DSETZGAP	4	Z gap parameter is present

`nDepth` Sets the depth of the graph, projected into the page, as a percentage of the default. For 3D pie charts, values can range from 10 to 200. For all other 3D graph types, any positive integer is acceptable. In all cases, a value of 100 preserves the default depth.

`nXGap and nZGap` Set the space imposed between bars as a percentage of the distance between their centers. A value of 50 imposes a space equal to the width of a bar. The default is 20.

Return values

0	Success
-1	Failure

Description

The AG3DStyle function sets the style parameters for a 3D graph.

Topic

AG3DStyle

Related

AGCageStyle
AGGridStyle
AGSetPerspective
AGSurfaceClr
AGXAxisStyle
AGYAxisStyle
AGZAxisStyle

Window initialization:
GSOpenWin
GSOpenChildWin

Graph display:
AGOpen
AGShow
AGRefresh3D
AGClose

AGAmp function

Transfers amplitude data

C/C++

```
int AGAmp( int nPts, int nGroup, double* fAmp )
```

FoxPro

```
r = AGAmp(nPts, nGroup, @fAmp(1))
```

Visual Basic

```
r% = AGAmp(nPts%, nGroup%, fAmp#(0))
```

Parameters

`nPts`	Number of points per data set. The size of the amplitude array is the number of data points per data set multiplied by the number of data sets.
`nGroup`	Number of data sets. For example, line graphs have a number of lines equal to nGroup, and stacked bar graphs have nGroup segments per bar.
`fAmp`	Pointer to array of amplitude data. Amplitude data is the principal data represented in a graph. It determines the magnitudes of pie slices, lengths of bar elements, positions of points in a line graph, and so on. Amplitude data is used in all graph types.

Return values

0	Success
-1	Failure

Description

The AGAmp function transfers amplitude data to AutoGraph and defines the number of points in a graph and the number of data sets.

Certain graph types require a specific number of data sets. A pie chart can only represent one data set at a time; a high-low-close graph always requires three data sets specifying the high, low, and close values of each point of the graph.

Example

The following example shows how to define and pass the amplitude array to draw a line graph consisting of two lines with four data points each. The first line would have points of amplitude 100, 150, 200, and 250, while the second would have points of amplitude 400, 300, 200, and 100.

```
#define   NUMPOINTS   4
#define   NUMSETS     2
double    fAmp [NUMPOINTS] [NUMSETS] = {
            /* Line 1   Line 2 */
/* Point 1 */  100.0,   400.0,
/* Point 2 */  150.0,   300.0,
/* Point 3 */  200.0,   200.0,
/* Point 4 */  250.0,   100.0
};
AGAmp(NUMPOINTS, NUMSETS, &fAmp[0][0]);
```

Topic

AGAmp

Related

AGAmpError
AGAux
AGClr
AGDataZ
AGDist
AGDistError
AGPatt
AGSym

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGAmpError function

Transfers amplitude error data

C/C++

```
int AGAmpError( int nPts, int nGroup, double* fAmpErr )
```

FoxPro

```
r = AGAmpError(nPts, nGroup, @fAmpErr(1))
```

Visual Basic

```
r% = AGAmpError(nPts%, nGroup%, fAmpErr#(0))
```

Parameters

`nPts`	Number of error values per data set
`nGroup`	Number of data sets
`fAmpErr`	Pointer to array of error data

Return values

0	Success
-1	Failure

Description

The AGAmpError function transfers amplitude error data for user-defined error bars.

Errors are passed as plus and minus error pairs stored consecutively. Both values must be passed even though the display of one or other may be suppressed.

There are two modes in which errors can be passed. First, where an error pair is supplied for every point in the graph. In this case, nPts is twice the value used in AGAmp (to account for the pair of values) and nGroup is the same as in AGAmp. Second, where an error pair is passed for each group, the same error pair being applied to each data point in the group. In this case, nPts is always 2 and nGroup is the same as in AGAmp. These modes are detected automatically from the value of nPts.

Example

The following example shows how to define and pass the amplitude error array for a line graph consisting of two lines with two data points each. The first line would have error pairs (100,200) and (150,300), and the second would have (100,30) and (50,25).

```
#define   NUMPOINTS   2
#define   NUMSETS     2
double    fAmpErr [NUMPOINTS * 2] [NUMSETS] = {
            /* L1 Err+  L1 Err- L2 Err+ L2 Err- */
/* Point 1 */  100.0,   200.0,  100.0,  30.0,
/* Point 2 */  150.0,   300.0,   50.0,  25.0,
};
AGAmpErr(NUMPOINTS * 2, NUMSETS, &fAmpErr[0][0]);
```

Topic

AGAmpError

Related

AGAmp
AGAux
AGClr
AGDataZ
AGDist
AGDistError
AGErrorBar
AGPatt
AGSym

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGAxisMinorTicks function

Sets the number of minor ticks for each axis

C/C++

```
int AGAxisMinorTicks(int nSelect, int nTicks)
```

FoxPro

```
r = AGAxisMinorTicks(nSelect, nTicks)
```

Visual Basic

```
r% = AGAxisMinorTicks(nSelect%, nTicks%)
```

Parameters

`nSelect`

Constant	Value	Meaning
AGXAXIS	0	X axis
AGYAXIS	1	Y axis (with dual axes, left-hand Y)
AGYRAXIS	2	Right-hand Y axis (dual axes)

`nTicks` Number of minor ticks

Return values

0	Success
-1	Failure

Description

The AGAxisMinorTicks function specifies the number of minor ticks to be drawn between intervals of major ticks. The nSelect parameter specifies the axis, nTicks specifies the number.

Before calling AGAxisMinorTicks, you must first call the appropriate axis style function, passing AGAXISMINORTICK in its nMode parameter. This turns on minor ticks so that a subsequent call to AGAxisMinorTicks can set their frequency.

Topic

AGAxisMinorTicks

Related

AGXAxisStyle
AGYAxisStyle
AGYRAxisStyle

AGAux function

Transfers auxiliary information

C/C++

```
int AGAux( int nSize, int* nAux )
```

FoxPro

```
r = AGAux(nSize, @nAux(1))
```

Visual Basic

```
r% = AGAux(nSize%, nAux%(0))
```

Parameters

`nSize`	Size of array
`nAux`	Pointer to array of auxiliary data

Return values

0	Success
-1	Failure

Description

The AGAux function transfers auxiliary data to AutoGraph.

Auxiliary data has four uses: to "explode" pie chart slices, to set the colors of the sides and tops of bars in 3D bar graphs, to specify statistical lines for time series graphs, and to specify "missing" data points in several graph types (line, lin/log, log/lin, log/log, polar, scatter, and tape).

The use of auxiliary data and the size of the array when used to set attributes depends on the type of graph, as shown in the following table.

Graph type	Array size	Used for
Pie	nPts	Pie segment explosion indicator. Use 0 for a normal segment and 1 for an exploded segment.
Bar	nGroup	The color of the tops and sides of 3D bars. Use normal nClr values; Graphics Server uses half-tones (shaded versions) of these colors to draw the bar sides.
Time series	nGroup	Sets statistical lines for each data set. Value 1 superimposes mean line; value 2 superimposes standard deviation.

Specifying auxiliary data is optional. If you don't specify auxiliary data for a graph type that uses it, AutoGraph creates a temporary internal array containing appropriate default values. By default, pie slices aren't exploded, the sides of 3D bar elements are drawn in the half-tone colors of the front surfaces, and time series graphs have no statistical lines.

Specifying points as "missing" in line, logarithmic, polar, 2D scatter, and tape graphs

If you have incomplete sets of data or sets in which the values of certain points are unknown, you can use the nAux array to flag

such points as missing. In this case, the marker for that point--such as a symbol--isn't drawn. If the graph uses lines (or tapes) to connect points, the connecting lines or tapes are omitted both to and from each missing point.

To flag missing data, you set an nAux value of 256 for that point. Points with nAux values of 0 are shown normally.

The size of the nAux array may be nPts or nPts nGroup. If you set the size to nPts and there's more than one group of data, the same missing points are assumed for all the groups. If the size is nPts nGroup, each point in each group has its own missing-value flag.

Topic

AGAux

Related

AGAmp
AGAmpError
AGClr
AGDataZ
AGDist
AGDistError
AGPatt
AGSym

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGBar2DGap function

Adjusts the gap between bars in 2D bar graphs

C/C++

```
int AGBar2DGap(int nGap)
```

FoxPro

```
r = AGBar2DGap(nGap)
```

Visual Basic

```
r% = AGBar2DGap(nGap%)
```

Parameters

`nGap`	Gap expressed as a percentage of the space between bars.

Return values

`0`	Success
`-1`	Failure

Description

The AGBar2DGap function defines the space between adjacent bars in a 2D bar graph or Gantt chart. The size of the space is expressed as a percentage of the maximum possible gap, which varies according to the size of the graphing window and the number of bars in the graph. A setting of 0 eliminates the gap altogether, placing the bars directly next to each other. A setting of 100 eliminates the bars altogether, making them infinitely thin.

By default a gap of 20 percent is placed between bars.

Topic

AGBar2DGap

Related

AGLabels

AGShow

AGCageStyle function

Sets 3D cage style

C/C++

```
int AGCageStyle( int nMode, int nClrWall,
                 int nClrEdge )
```

FoxPro

```
r = AGCageStyle(nMode, nClrWall, nClrEdge)
```

Visual Basic

```
r% = AGCageStyle(nMode%, nClrWall%, nClrEdge%)
```

Parameters

`nMode`

Constant	Value	Meaning
	0	Thick walls (default)
AGCAGETHIN	1	Thin side walls (zero thickness)

`nClrWall` Color of faces of walls (see Color constants).

Use a color index of -1 to leave wall faces with their default colors.

`nClrEdge` Color of edges of walls (see Color constants).

Use a color index of -1 to leave wall edges with their default colors.

Return values

0	Success
-1	Failure

Description

The AGCageStyle function sets the style of the cage that encloses a True3D graph.

Topic

AGCageStyle

Related

AG3DStyle
AGSetPerspective
AGGridStyle
AGSurfaceClr
AGXAxisStyle
AGYAxisStyle
AGZAxisStyle

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen

AGShow

AGRefresh3D

AGClose

AGClose function

Terminates AutoGraph

C/C++

```
int AGClose( )
```

FoxPro

```
r = AGClose()
```

Visual Basic

```
r% = AGClose()
```

Return values

0	Success
-1	Failure

Description

The AGClose function terminates AutoGraph, freeing the memory allocated for AutoGraph data arrays.

After this function is called, no further AutoGraph functions can be used until AutoGraph is enabled by calling AGOpen. One exception is the AGTimeUpdate function, which can be called after AGClose.

Graphics Server is always able to execute standard functions, which remain unaffected by AGOpen and AGClose.

Topic

AGClose

Related

AGOpen

AGReset

AGShow

AGClr function

Transfers color information

C/C++

```
int AGClr( int nSize, int* nClr )
```

FoxPro

```
r = AGClr(nSize, @nClr(1))
```

Visual Basic

```
r% = AGClr(nSize%, nClr%)
```

Parameters

`nSize`	Size of array
`nClr`	Pointer to color array (see Color constants)

Return values

0	Success
-1	Failure

Description

The AGClr function transfers color data to AutoGraph.

Specifying color data is optional. If you don't, AutoGraph creates a temporary internal array containing appropriate default values.

Color data is used in all graph types except surface. The same colors are used in the legend if one is present.

The use of color data and the size of the array depends on the type of graph to be drawn, as shown in the following table.

Graph type	Array size	Used for
Pie	nPts	Color of each pie segment
Bar	nGroup	Color of the bar elements of each data set
Gantt	nGroup	Color of the bar elements of each data set
Line	nGroup	Color of the line, symbols, and sticks of each data set
Log/lin, lin/log, and log/log	nGroup	Color of the line, symbols, and sticks of each data set
Area	nGroup	Color of the area of each data set
Scatter	nGroup	Color of the symbols of each data set
Polar	nGroup	Color of the line, symbols, and sticks of each data set
High-low-close	1	Color of the high-low-close symbols
Bubble	nPts	Color of each bubble
Tape	nGroup	Color of the tape of each

		data set
Surface	0	See AGSurfaceClr
Time series	nGroup	Sets color of each set
Box-whisker	nPts	Sets color of each symbol

Topic
AGClr

Related
AGAmp
AGAmpError
AGAux
AGDataZ
AGDist
AGDistError
AGPatt
AGSurfaceClr
AGSym

Window initialization:
GSOpenWin
GSOpenChildWin

Graph display:
AGOpen
AGShow
AGClose

AGCurveStyle function

Sets curve style

C/C++

```
int AGCurveStyle( int nType, int nOrder, int nSteps )
```

FoxPro

```
r = AGCurveStyle(nType, nOrder, nSteps)
```

Visual Basic

```
r% = AGCurveStyle(nType%, nOrder%, nSteps%)
```

Parameters

`nType`

Constant	Value	Meaning
CFPOLY	0	Variable-order polynomial
CFLOG	1	Logarithmic y = a + b * ln(x)
CFEXP1	2	Exponential y = a * exp(b * x)
CFEXP2	3	Exponential y = a * x * exp(-b * x)
CFPOWER	4	Power y = a * (x ^ b)
CFINV1	5	Inverse y = a + b / x
CFINV2	6	Inverse y = a / (b + x)
CFINV3	7	Inverse y = 1 / (a + b * x)
CFINV4	8	Inverse y = x / (a * x + b)
CFINV5	9	Inverse y = 1 / (a + b * x) ^ 2
CFSPLINE	10	Spline fit through all points
CFMOVINGAVEMID	11	Moving average plotted at midpoint of averaged group
CFMOVINGAVEEND	12	Moving average plotted at end point of averaged group

`nOrder`

Curve order. nOrder is only relevant to the variable-order polynomial fit and moving averages. For moving averages, nOrder defines the number of points over which the average is taken.

`nSteps`

Number of steps. nSteps defines the granularity of the drawn curve; higher nSteps values lead to smoother curves.

- *For most curves (nTypes 1-9),* an nSteps setting of 50 generally produces a smooth curve at a high drawing speed.
- *For spline curves (nType CFSPLINE),* you generally need a much larger nSteps value--typically 10 times the number of points in the graph, or higher for very irregular graphs.
- *For moving averages (nType CFMOVINGAVEMID or CFMOVINGAVEEND),* nSteps isn't relevant--the lines drawn between the plotted averages are always straight.

Return values

0	Success
-1	Failure

Description

The AGCurveStyle function specifies the style in which a curve is fitted to a graph. It applies only to 2D scatter, line, high-low-close, open-high-low-close, candlestick, and box-whisker graphs

The nStats parameter in the AGShow function should be set to AGCURVEFIT.

Moving averages may be plotted either at the midpoint of the group of averaged data or at the end point.

Topic

AGCurveStyle

Related

AGXAxisStyle
AGYAxisStyle
AGZAxisStyle
GSStatsArr

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGDataLabels function

Enables and sets text for data labels

C/C++

```
int AGDataLabels( int nMode, int nLabs,
                  char* szLabels[] )
```

FoxPro

```
r = AGDataLabels(nMode, nLabs, @szLabels(1))
```

Visual Basic

```
r% = AGDataLabels(nMode%, nLabs%, szLabels$(1))
```

Parameters

`nMode`

Constant	Value	Meaning
AGDLTEXT	0	Labels supplied in array szLabels
AGDLDATA	1	Labels derived from data
AGDLGROUPCLR	4	Color as data group

`nLabs`

Value	Meaning
0	Use if deriving labels from data (nMode AGDLDATA)
1 or greater	Use for number of labels if supplying text labels (nMode AGDLTEXT) The label array must be of size nPts nGroup to provide text labels for each data item on display. The exceptions are high-low-close, open-high-low-close, candlestick, and box-whisker graphs, which require a text array of size nPts (only one label is provided for each compound symbol, of which there are nPts).

`szLabels` Array of text labels of length nLabs

Return values

0	Success
-1	Failure

Description

The AGDataLabels function enables data labels, which are labels--either numeric or text--attached to each point of a graph. Data labels are available for all 2D graph types except pie charts (which have their own labeling scheme) and time series graphs. They aren't available for 3D graphs.

In high-low-close, open-high-low-close, box-whisker, and candlestick graphs, if you choose to have data labels derived from data (nMode AGDLDATA), they're derived from the close or

median.

J

Topic
AGDataLabels

Related
AGFontStyle
AGLabelY
AGLabelZ
AGLegend
AGTitleG
AGTitleX
AGTitleY
AGTitleYR
VBAGDataLabels

Window initialization:
GSOpenWin
GSOpenChildWin

Graph display:
AGOpen
AGShow
AGClose

AGDataRange function

Sets a range of data to be graphed

C/C++

```
int AGDataRange(int nRangeMin, int nRangeMax)
```

FoxPro

```
r = AGDataRange(nRangeMin, nRangeMax)
```

Visual Basic

```
r% = AGDataRange(nRangeMin%, nRangeMax%)
```

Parameters

`nRangeMin`	Lower index of data to be graphed.
`nRangeMax`	Higher index of data to be graphed.

Return values

`0`	Success
`-1`	Failure

Description

The AGDataRange function defines a subset of the data to be graphed. Without changing the composition of the data array passed by AGAmp, a section of the complete data can be viewed, with the graph axes adjusting appropriately.

The lower and upper bounds of the data subset are defined by RangeMin and RangeMax. By default RangeMin is the first element of the array and RangeMax is the last element.

J

Topic

AGDataRange

Related

AGAmp
AGShow

AGDataZ function

Transfers Z data

C/C++

```
int AGDataZ( int nSize, double* fZData )
```

FoxPro

```
r = AGDataZ(nSize, @fZData(1))
```

Visual Basic

```
r% = AGDataZ(nSize%, fZData#(0))
```

Parameters

`nSize`	Size of array
`fZData`	Pointer to array of Z data

Return values

0	Success
-1	Failure

Description

The AGDataZ function transfers Z data to AutoGraph.

Z data is used in True3D scatter (or point) graphs to specify the position of each data point on the Z axis projected into the page or screen.

In a graph of more than one data set, Z data may be supplied on a per-set basis, or the same Z data can be applied to all the data sets in the graph. The size of the Z data array is, accordingly, the number of points per data set multiplied by the number of data sets, or simply the number of points per data set.

J

Topic

AGDataZ

Related

AGAmp
AGAmpError
AGAux
AGClr
AGDist
AGDistError
AGPatt
AGSym

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen

AGShow
AGClose

AGDist function

Transfers distance data

C/C++

```
int AGDist( int nSize, double* fDist )
```

FoxPro

```
r = AGDist(nSize, @fDist(1))
```

Visual Basic

```
r% = AGDist(nSize%, fDist#(0))
```

Parameters

`nSize`	Size of array
`fDist`	Pointer to array of distance data

Return values

0	Success
-1	Failure

Description

The AGDist function transfers distance data to AutoGraph.

Distance data is used in axis-based graphs to specify the position of each data point on the category (usually the X) axis.

In a graph of more than one data set, distance data may be supplied on a per-set basis, or the same distance data can be applied to all the data sets in the graph. The size of the distance array is, accordingly, the number of points per data set multiplied by the number of data sets, or simply the number of points per data set.

For most graphs, distance data is optional. If you omit it, points are drawn at regular intervals (usually 0, 1, 2, and so on) along the axis. However, lin/log and log/log graphs must always have distance data, and scatter graphs generally have it (by definition, scatter graphs plot points based on independent X and Y variables).

The bubble graph also requires distance data, but it's unusual because it requires two-dimensional distance data in every case, even though the amplitude data is one-dimensional. In this case, the size of the distance array is the number of points multiplied by two and the array contains the (X,Y) coordinates of the centers of the bubbles.

J

Topic

AGDist

Related

AGAmp
AGAmpError
AGAux

AGClr
AGDataZ
AGDistError
AGPatt
AGSym

Window initialization:
GSOpenWin
GSOpenChildWin

Graph display:
AGOpen
AGShow
AGClose

AGDistError function

Transfers distance error data

C/C++

```
int AGDistError( int nPts, double* fDistErr )
```

FoxPro

```
r = AGDistError(nPts, @fDistErr(1))
```

Visual Basic

```
r% = AGDistError(nPts%, fDistErr#(0))
```

Parameters

`nPts`	Number of error values in array
`fDistErr`	Pointer to array of error data

Return values

0	Success
-1	Failure

Description

The AGDistError function transfers distance error data for user-defined error bars.

Errors are passed as plus and minus error pairs stored consecutively. Both values must be passed even though the display of one or the other may be suppressed.

There are two modes in which errors can be passed. The first mode is where an error pair is supplied for every point in the graph. In this case, nPts is twice the value used in AGDist (to account for the pair of values). The second mode is where an error pair is passed for each group, the same error pair being applied to each data point in the group. In this case, nPts is (2 nAmpGroup) where nAmpGroup is the group value passed in AGAmp. These modes are detected automatically from the value of nPts.

Example

The following example shows how to define and pass the distance error array for a scatter graph consisting of groups with two data points each. The first graphed set would have error pairs (100,200) (150,300), the second (100,30) (50,25).

```
#define   NUMPOINTS   2
#define   NUMSETS     2
double    fDistErr [NUMPOINTS * 2 * NUMSETS] = {
            /* S1 Err+  S1 Err- S2 Err+ S2 Err- */
/* Point 1 */  100.0,   200.0,  100.0,  30.0,
/* Point 2 */  150.0,   300.0,   50.0,  25.0,
};
AGDistErr(NUMPOINTS * 2 * NUMSETS,&fDistErr[0][0]);
```

J

Topic

AGDistError

Related

AGAmp
AGAmpError
AGAux
AGClr
AGDataZ
AGDist
AGPatt
AGSym

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGErrorBar function

Defines error bars for graph

C/C++

```
int AGErrorBar( int nSelect, int nSymbolStyle,
                int nColor, int nErrorSource,
                double fValue )
```

FoxPro

```
r = AGErrorBar(nSelect, nSymbolStyle, nColor,
               nErrorSource, fValue)
```

Visual Basic

```
r% = AGErrorBar(nSelect%, nSymbolStyle%, nColor%,
                nErrorSource%, fValue#)
```

Parameters

`nSelect`

Constant	Value	Meaning
AGEBY	0	Defines error bars for Y (amplitude) data
AGEBX	1	Defines error bars for X (distance) data

`nSymbolStyle`

Constant	Value	Meaning
	0	Standard style
AGEBNOPLUS	1	Omits plus bar
AGEBNOMINUS	2	Omits minus bar
AGEBNOSTEM	4	Omits stem
AGEBNOTICK	8	Omits cross tick

`nColor`

Error bar color

`nErrorSource`

Constant	Value	Meaning
AGEBFIXED	0	Error is fixed value equal to fValue
AGEBPERCENT	1	Error is fValue (here a percentage) times the data value, divided by 100
AGEBSTDDEV	2	Error is standard deviation times fValue
AGEBSTDERR	3	Error is standard error
AGEBMAXMIN	4	Errors are provided in arrays

`fValue`

Value modifier. fValue is used in a variety of ways, depending on the error source: as a fixed

value for all data points (AGEBFIXED), as a modifier when the error is expressed as a percentage of the data value (AGEBPERCENT), and finally as a multiplying factor when the error is expressed as the standard deviation of the data set (AGEBSTDDEV).

Return values

0	Success
-1	Failure

Description

The AGErrorBar function defines the format of error bars to be added to a graph.

Error bars can be applied to the following 2D graph types:

- Horizontal bar graphs (simple and clustered format)
- Vertical bar graphs (simple and clustered formats)
- Line graphs (line and symbol formats)
- Scatter graphs

For scatter graphs, you can specify both horizontal (distance data) and vertical (amplitude data) error bars. To use both, you have to call the function twice--first with nSelect AGEBY, then with nSelect AGEBX.

With mode AGEBMAXMIN, the error is supplied in the arrays fAmpErr and fDistErr, as set by the functions AGAmpError and AGDistError. Errors are supplied as paired *plus* and *minus* error values, both of which must be positive.

Topic

AGErrorBar

Related

AGAmp
AGAmpError
AGAux
AGClr
AGDataZ
AGDist
AGDistError
AGPatt
AGSym

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGFFT function

Performs Fast Fourier Transform on array of data

C/C++

```
int AGFFT( int nPts, double* fData, int nMode )
```

FoxPro

```
r = AGFFT(nPts, @fData(1), nMode)
```

Visual Basic

```
r% = AGFFT(nPts%, fData#(0), nMode%)
```

Parameters

`nPts` Number of data points in array (must be a power of 2)

`fData` Pointer to data array

`nMode`

Constant	Value	Meaning
AGFFTREALIMAG	0	Returns real and imaginary coefficients
AGFFTAMPPHASE	1	Returns amplitude and phase
AGFFTSQAMPPHASE	2	Returns squared amplitude and phase
AGFFTINTERLEAVE	16	Result vectors are interleaved

Return values

0	Success
-1	Failure

Description

The AGFFT function performs a Fast Fourier Transform on an array of data, returning the result in the same array and overwriting the source data.

The data is returned as two linear vectors each of length nPts/2 stored consecutively in the source array. The returned data may be in the form of the real and imaginary coefficients, the amplitude and phase, or the squared amplitude and phase where phase is in degrees between -180 and 180. Data is normalized to be the coefficients of the series $a0 + a1\mathrm{Cos}(f1) + b1\mathrm{Sin}(f1) +$, and so forth.

The AGFFTINTERLEAVE option returns the two result sets interleaved in the source array such that the real and imaginary values, or amplitude and phase, are stored in pairs for each data point. In the default form, the two resulting vectors can be drawn in separate graphs by passing the address of the base of the array and the mid-element of the array. In the interleaved form, the vectors can be drawn on the same graph by passing the address of the base of the array (combined graphs require the data to be interleaved).

J

Topic

AGFFT

Related

AGAmp
AGAmpError
AGAux
AGClr
AGDataZ
AGDist
AGDistError
AGPatt
AGSym

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGFGColor function

Sets color of foreground objects

C/C++

```
int AGFGColor( int nMode, int nColor )
```

FoxPro

```
r = AGFGColor(nMode, nColor)
```

Visual Basic

```
r% = AGFGColor(nMode%, nColor%)
```

Parameters

	Constant	Value	Meaning
nMode			
	AGFGALL	0	All objects
	AGFGTITLEG	1	Graph title text
	AGFGTITLEYL	2	Left title text
	AGFGTITLEYR	3	Right title text
	AGFGTITLEX	4	Bottom title text
	AGFGLABELS	5	Label text
	AGFGLEGEND	6	Legend text
	AGFGAXIS	7	Axis
	AGFGGRID	8	Grid lines
	AGFGMEAN	9	Mean lines
	AGFGMINMAX	10	Min and max lines
	AGFGSTDDEV	11	Standard deviation lines
	AGFGBESTFIT	12	Best-fit line
	AGFGCURVE	13	Curve fit line
	AGFGDATALABELS	14	Data labels
	AGFGLIMITLINES Limit lines, shading and text.	15	Limit lines, shading, and text.
nColor	Color of object		

Return values

0	Success
-1	Failure

Description

The AGFGColor function sets the color of foreground objects such as text and axes.

J

Topic

AGFGColor

Related

AGTitleBG
AGGraphBG
AGLegendStyle
AGShow

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGFontStyle function

Sets font style

C/C++

```
int AGFontStyle( int nUse, int nFamily, int nAttrib,
                 int nSize )
```

FoxPro

```
r = AGFontStyle(nUse, nFamily, nAttrib, nSize)
```

Visual Basic

```
r% = AGFontStyle(nUse%, nFamily%, nAttrib%, nSize%)
```

Parameters

`nUse`

Constant	Value	Meaning
AGFUSETITG	0	Graph title
AGFUSETITXY	1	Other titles (left, right, bottom)
AGFUSELABS	2	Labels
AGFUSELEG	3	Legend

`nFamily`

Constant	Value	Meaning
FOROMAN	1	Roman
FOSWISS	2	Swiss
FOMODERN	3	Modern
FOSCRIPT	4	Script
FODECO	5	Decorative

`nAttrib`

Constant	Value	Meaning
FOITALIC	16	Italic
FOBOLD	32	Bold
FOULINE	64	Underlined

`nSize`

Size as a percentage of the size of the system font. For example, an nSize of 400 selects a font four times the size of the system font.

AutoGraph uses nSize as a guide only. nSize is the maximum size allowed, but AutoGraph uses a smaller size if necessary to ensure good label fitting.

If nSize is set to 0, AutoGraph uses its default font size, which varies according to the value of nUse.

Return values

0	Success
-1	Failure

Description

The AGFontStyle function specifies the font style to be used by AutoGraph. A different set of style options can be applied to different components of the graph by means of the nUse parameter.

J

Topic

AGFontStyle

Related

AGLabels
AGLabelY
AGLabelZ
AGLegend
AGTitleG
AGTitleX
AGTitleY
AGTitleYR

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGGraphBG function

Sets graph background

C/C++

```
int AGGraphBG( int nMode, int nClr )
```

FoxPro

```
r = AGGraphBG(nMode, nClr)
```

Visual Basic

```
r% = AGGraphBG(nMode%, nClr%)
```

Parameters

`nMode`

Constant	Value	Meaning
AGGRFRAME	1	Black border around axes
AGGRFILL	2	Fills background with nClr
AGGRDROPSHADOW	4	Drop shadow
AGGRRAISED	8	Raised border
AGGRLOWERED	16	Lowered border

Modes 4, 8, and 16 can't be combined.

`nClr` Background color (see Color constants)

Return values

0	Success
-1	Failure

Description

The AGGraphBG function sets the background color and style of the area on which the graph and axes are drawn.

J

Topic

AGGraphBG

Related

AGFGColor
AGTitleBG
AGLegendStyle

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow

AGClose

AGGridStyle function

Sets style of grid lines

C/C++

```
int AGGridStyle( int nSelect, int nStyleMajor,
                 int nStyleMinor )
```

FoxPro

```
r = AGGridStyle(nSelect, nStyleMajor, nStyleMinor)
```

Visual Basic

```
r% = AGGridStyle(nSelect%, nStyleMajor%, nStyleMinor%)
```

Parameters

`nSelect`

Constant	Value	Meaning
AGGRIDX	0	Sets X grid
AGGRIDY	1	Sets Y grid
AGGRIDZ	2	Sets Z grid

`nStyleMajor` Line style for major grids (see Line style constants)

`nStyleMinor` Line style for minor grids (see Line style constants)

Return values

0	Success
-1	Failure

Description

The AGGridStyle function sets the style of grid lines. It must be called separately to set the styles of X, Y, and Z grids. To set the color of grids, use the AGFGColor function.

J

Topic

AGGridStyle

Related

AGGraphBG
AGInfo
AGLegendStyle
AGXAxisStyle
AGYAxisStyle
AGZAxisStyle

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen

AGShow
AGClose

AGInfo function

Gets AutoGraph drawing information

C/C++

```
double AGInfo( int nIndex )
```

FoxPro

```
r = AGInfo(nIndex)
```

Visual Basic

```
r% = AGInfo(nIndex%)
```

Parameter

`nIndex`

Value	**Meaning**
0	X axis maximum value
1	X axis minimum value
2	Y axis maximum value
3	Y axis minimum value
4	X axis length in view units
5	Y axis length in view units
6	X origin of graph in view units
7	Y origin of graph in view units
8	Label font height expressed as a percentage of the system font height
9	Right Y axis maximum value
10	Right Y axis minimum value
11	Z axis maximum value
12	Z axis minimum value
13	Number of ticks on +ve X axis
14	Number of ticks on -ve X axis
15	Number of ticks on +ve Y axis
16	Number of ticks on -ve Y axis
17	Number of ticks on +ve right Y axis
18	Number of ticks on -ve right Y axis
19	Number of ticks on +ve Z axis
20	Number of ticks on -ve Z axis

Note that the number of ticks does not include the tick at the intersection of two axes (eg. X with Y axis, Y with Z axis). In 3D graphs and graphs with axes moved from their default positions, the zero-value origin may not coincide with the intersection of the axes. In this case the tick at the zero-value is included in the count of negative ticks.

Return value

AutoGraph information on the selected item (-1 if failure)

Description

The AGInfo function returns any one of 21 useful items of information relating to where and how AutoGraph has drawn its graph and to assist with custom labelling of graphs, using the SDK functions.

The SDKInfo property in the VBX, OCX, and CGraph provides similar information. Note that the indices are one greater for SDKInfo than AGInfo owing to the (array) property being one-based.

J

Topic

AGInfo

Related

GSGetVXExt
GSGetVYExt
GSGetCurveCoeff

Window initialization:

GSOpenView
GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGLabelFormat function

Defines a template for formatting numerical axis and data labels

C/C++

```
int AGLabelFormat(int nSelect, char *szFormatString)
```

FoxPro

```
r = AGLabelFormat(nSelect, szFormatString)
```

Visual Basic

```
r% = AGLabelFormat(nSelect%, szFormatString$)
```

Parameters

`nSelect` Indicates which labels are to be formatted.

Constant	Value	Meaning
AGFORMATX	0	X labels
AGFORMATYL	1	Y left labels
AGFORMATYR	2	Y right labels
AGFORMATZ	3	Z labels
AGFORMATDLABS	4	Data labels

`szFormatString` A string providing the template for formatting.

Return values

`0`	Success
`-1`	Failure

Description

The AGLabelFormat function defines a template for formatting numeric axis and data labels. A template can also be applied to automatically generated date/time labels along the X axis. For more information, see numeric formats and date/time formats.

Topic

AGLabelFormat

Related

AGLabelDateTime
AGLabels
AGLabelY
Date/time formats
Numeric formats

AGLabelDateTime function

Defines date/time labels for an axis

C/C++

```
int AGLabelDateTime(int nSelect, int nMode,
                    char *szDTStart, char *szDTInc)
```

FoxPro

```
r = AGLabelDateTime(nSelect, nMode,
                    szDTStart, szDTInc)
```

Visual Basic

```
r% = AGLabelDateTime(nSelect%, nMode%,
                     szDTStart$, szDTInc$)
```

Parameters

`nSelect`

Constant	Value	Meaning
AGXAXIS	0	Apply to X axis labels

`nMode`

Constant	Value	Meaning
AGDTOFF	0	Disabled
AGDTDATE	1	Date labels
AGDTTIME	2	Time labels May be combined with AGDTDATE.
AGDTSKIPWKEND	4	Date labels, skip weekends Applies only when combined with AGDTDATE.

`szDTStart`

Starting date and time

The date and time must be passed in a fixed format:

"yyyy:mm:dd:hh:mm:ss" where

yyyy is the year (1900-2036)
mm is the month (01-12)
dd is the day (01-31)
hh is the hour (00-24)
mm is the minute (00-59)
ss is the second (00-59)

Example

```
'Start at 9:15 pm on 15 Nov 93
szDTStart$ = "1993:11:15:21:15:00"
```

`szDTInc`

Increment

The increment must be passed in a fixed format:

"yyyy:mm:dd:hh:mm:ss" where

yyyy is the year increment (0000-0100)
mm is the month increment (00-99)
dd is the day increment (00-99)

hh is the hour increment (00-99)
mm is the minute increment (00-99)
ss is the second increment (00-99)

Example

```
'Increment 1 year and 6 seconds
szDTInc$ = "0001:00:00:00:00:06"
```

Return values

0	Success
-1	Failure

Description

The AGLabelDateTime function defines a series automatically generated date/time labels for the X axis.

By default, X axis labels are numeric. If an array of text labels has been passed to AGLabels() , labels for the axis are text. If AGLabelDateTime() is called with nMode greater than 0, labels for the axis are date/time, regardless if an array of text labels is present. To disable date/time labeling, call the function with nMode AGDTOFF.

nMode selects the type of date/time labels. For a series of dates, pass AGDTDATE. If you want dates that skip weekends, pass AGDTDATE + AGDTSKIPWKEND. For a series of times, pass AGDTTIME. For labels that combine date and time, pass AGDTDATE + AGDTTIME.

The szDTStart parameter sets the starting date and/or time for the series. The increment for intervals along the axis is set by the szDTInc parameter.

Labels may be formatted by calling the AGLabelFormat function. If a label format string is not applied, labels will display in the default format, "mm/dd/yy" for dates and "hh:mm:ss" for times.

J

Topic

AGLabelDateTime

Related

AGLabelFormat
AGLabels
AGXAxisStyle
Date/time formats

AGLabels function

Defines labels for axis or pie chart

C/C++

```
int AGLabels( int nLabs, char* szLabs[] )
```

FoxPro

```
r = AGLabels(nLabs, @szLabs(1))
```

Visual Basic

```
r% = AGLabels(nLabs%, szLabs$(1))
```

Parameters

`nLabs`	Number of labels
`szLabs`	Pointer to array of text labels

Return values

0	Success
-1	Failure

Description

The AGLabels function transfers an array of labels for a graph's Y axis (or X axis in the case of horizontal bar graphs and Gantt charts) or the slices of a pie chart.

J

Topic

AGLabels

Related

AGFontStyle
AGLabelY
AGLabelZ
AGLegend
AGTitleG
AGTitleX
AGTitleY
AGTitleYR

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGLabelY function

Defines labels for left or right Y axis

C/C++

```
int AGLabelY( int nSelect, int nNLabs, char* szLabs[] )
```

FoxPro

```
r = AGLabelY(nSelect, nNLabs, @szLabs(1))
```

Visual Basic

```
r% = AGLabelY(nSelect%, nNLabs%, szLabs$(1))
```

Parameters

	Constant	Value	Meaning
`nSelect`	AGLABYLEFT	0	Sets left-hand Y axis labels
	AGLABYRIGHT	1	Sets right-hand Y axis labels
`nNLabs`	Number of labels		
`szLabs`	Pointer to array of text labels		

Return values

0	Success
-1	Failure

Description

The AGLabelY function transfers an array of labels for the Y axis. By default, the Y axis is labeled with numeric values according to the axis scale, which is either calculated automatically or set by the AGYAxisStyle function. This function allows arbitrary text labels to replace the numeric values.

The AGYAxisStyle function must be called to set the number of ticks on the axis and hence the number of labels to be supplied in the array.

The nSelect parameter selects between the left- and right-hand Y axes. The latter is only of relevance to combination graphs with a second Y axis drawn to a different scale.

Note that it's possible in graphs with a *single* Y axis to position that axis on the right, using the AGYAxisStyle function. However, this function still treats the axis as a left axis, and you should use nSelect = 0.

J

Topic

AGLabelY

Related

AGFontStyle
AGLabels
AGLabelZ

AGLegend
AGTitleG
AGTitleX
AGTitleY
AGTitleYR
AGYAxisStyle
VBAGLabelY

Window initialization:
GSOpenWin
GSOpenChildWin

Graph display:
AGOpen
AGShow
AGClose

AGLabelZ function

Defines labels for Z axis

C/C++

```
int AGLabelZ( int nMode, int nNLabs, char* szLabs[] )
```

FoxPro

```
r = AGLabelZ(nMode, nNLabs, @szLabs(1))
```

Visual Basic

```
r% = AGLabelZ(nMode%, nNLabs%, szLabs$(1))
```

Parameters

`nMode`

Value	**Meaning**
0	Currently no modes implemented

`nNLabs` Number of labels

`szLabs` Pointer to array of text labels

Return values

0	Success
-1	Failure

Description

The AGLabelZ function transfers an array of labels for the Z axis in True3D graphs. By default, this axis either carries no labels (for all True3D graph types except scatter) or is labeled with numeric values (for True3D scatter graphs). AGLabelZ lets you specify text labels to override these defaults.

The number of labels you need depends on the graph type:

- For True3D area (stacked style) and bar (simple, stacked, or clustered style) graphs, you need only one label.
- *For True3D area (absolute style), bar (z-clustered style), surface, and tape graphs,* you need one label for each data group. The groups are always drawn from back to front, and label array follows that order.
- *For True3D scatter graphs,* Z data values are provided in the Z data array, with the origin at the front. In this case, the Z axis is either drawn to a scale calculated automatically from the data or as specified in the AGZAxisStyle function. If you want to supply text labels, be sure to use AGZAxisStyle to set the number of ticks (and hence the number of labels) for the axis.

Topic

AGLabelZ

Related

AGFontStyle

AGLabels

AGLabelY
AGLegend
AGTitleG
AGTitleX
AGTitleY
AGTitleYR
AGZAxisStyle
VBAGLabelZ

Window initialization:
GSOpenWin
GSOpenChildWin

Graph display:
AGOpen
AGShow
AGClose

AGLegend function

Defines legend labels for grouped data

C/C++

```
int AGLegend( int nLegs, char* szLegs[] )
```

FoxPro

```
r = AGLegend(nLegs, @szLegs(1))
```

Visual Basic

```
r = AGLegend(nLegs%, szLegs$(1))
```

Parameters

`nLegs`	Number of legend labels
`szLegs`	Pointer to array of text labels

Return values

0	Success
-1	Failure

Description

The AGLegend function transfers an array of labels for the graph legend.

.

J

Topic

AGLegend

Related

AGFontStyle
AGLabels
AGLabelY
AGLabelZ
AGLegendStyle
AGTitleG
AGTitleX
AGTitleY
AGTitleYR
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGLegendStyle function

Sets position and style of legend

C/C++

```
int AGLegendStyle( int nVertical, int nHorizontal,
                   int nSize, int nClr, int nMode )
```

FoxPro

```
r = AGLegendStyle(nVertical, nHorizontal, nSize, nClr,
                  nMode)
```

Visual Basic

```
r% = AGLegendStyle(nVertical%, nHorizontal%, nSize%,
                   nClr%, nMode%)
```

Parameters

`nVertical`

Constant	Value	Meaning
AGLEGCENTRE	0	Center
AGLEGBOTTOM	1	Bottom
AGLEGTOP	2	Top

`nHorizontal`

Constant	Value	Meaning
AGLEGCENTRE	0	Center
AGLEGLEFT	1	Left
AGLEGRIGHT	2	Right

`nSize`

Percentage size relative to default size. The default size maximizes the use of the available space for the legend, expanding the gaps between adjacent legend entries either horizontally or vertically according to its placement on the graph. The value ranges from 0 to 100. At 0 the legend is reduced to the smallest line separation, and at 100 it's expanded to the largest line separation to fit the legend in the allocated space.

`nClr`

Background color (see Color constants)

`nMode`

Constant	Value	Meaning
AGLEGFRAME	1	Black border
AGLEGFILL	2	Fill background with nClr
AGLEGDROPSHADOW	4	Drop shadow
AGLEGRAISED	8	Raised border
AGLEGLOWERED	16	Lowered border

Modes 4, 8, and 16 can't be combined.

Return values

0	Success
-1	Failure

Description

The AGLegendStyle function sets the position and style of the legend. The default position is centered vertically to the right of the graph. The position parameters are combinational--for example, "top, left."

J

Topic

AGLegendStyle

Related

AGFGColor
AGFontStyle
AGGraphBG
AGTitleBG
AGLegend
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGLimitLines function

Applies limit lines to a graph

C/C++

```
int AGLimitLines(int nMode, int nLinePattern,
                 int nFillPattern, double fHighValue,
                 double fLowValue, char *szHighLabel,
                 char *szLowLabel)
```

FoxPro

```
r = AGLimitLines(nMode, nLinePattern,
                 nFillPattern, fHighValue,
                 fLowValue, szHighLabel,
                 szLowLabel)
```

Visual Basic

```
r% = AGLimitLines(nMode%, nLinePattern%,
                  nFillPattern%, fHighValue#,
                  fLowValue#, szHighLabel$,
                  szLowLabel$)
```

Parameters

`nMode`

Constant	Value	Meaning
AGLIMITHIGH	1	High value is present.
AGLIMITLOW	2	Low value is present.
AGLIMITOPPSHADE	4	Shade opposite sides.

`nLinePattern` Line pattern 0 - 5 (5 is null)

`nFillPattern` Fill pattern. 0 - Max patterns. 1 (null) means no shading.

`fHighValue` High limit line value.

`fLowValue` Low limit line value.

`szHighLabel` High limit line label.

`szLowLabel` Low limit line label.

Return values

0	Success
-1	Failure

Description

The AGLimitLines function superimposes lines on graphs to highlight data that falls outside prescribed limits. Both a high and low limit line can be shown. Text can be attached to the lines, and the areas either inside or outside the limits can be shaded with a pattern. Limit lines are not drawn on pie, polar, time-series, or any 3D graphs.

If a pattern of 1 (null) is specified, no shading is performed. If any other pattern index is provided, the areas above the high limit line and below the low limit line are filled to the extremes of the axes. If the mode is set to AGLIMITOPPSHADE, the opposite sides of the lines are shaded: with both limit lines present the area between is shaded.

Text is printed next to the line if a string is passed. If no text is required, a NULL string should be placed in the parameter list. With no shading present, text is printed immediately above the high line and immediately below the low line. With shading present, text is placed on the opposite side of the line to the shading.

The color of the lines and shading is set by AGFGColor.

J

Topic

AGLimitLines

Related

AGFGColor

AGMissingLineStyle function

Selects options for bridging gaps caused by missing data

C/C++

```
int AGMissingLineStyle(int nMode, int nSize
                       [, int *nPatt, int *nClr])
```

FoxPro

```
r = AGMissingLineStyle(nMode, nSize [, @nPatt(1),
                       @nClr(1)])
```

Visual Basic

```
r% = AGMissingLineStyle(nMode%, nSize% [, nPatt%(0),
                        nClr%(0)])
```

Parameters

`nMode`

Constant	Value	Meaning
AGMLSOMIT	0	No bridging lines (default)
AGMLSSAMESTYLE	1	Bridge with line style of the graph
AGMLSPATTERNED	2	Bridge with patterned lines
AGMLSTHICK	3	Bridge with thick lines

`nSize` If nMode is 2 or 3, size of nPatt and nClr arrays. Otherwise, zero.

`nPatt` Line pattern array. Required when nMode is 2 or 3.

`nClr` Line color array. Required when nMode is 2 or 3.

Return values

0	Success
-1	Failure

Description

If you have incomplete sets of data, or sets in which the values of certain data points are unknown, you can flag points as missing by calling the AGAux function. When the graph type is line, or any log variant of line, missing points cause a gap in the line. The AGMissingLineStyle function sets options for bridging gaps left by missing points.

nMode = 0 No bridging lines. This is the default, and it is what you get if you do not call AGMissingLineStyle(). However, if you have called the function and then later want to show gaps, this option will turn off bridging lines.

Set nSize to 0. Arrays for nPatt and nClr are ignored and can be omitted from the call.

nMode = 1 Bridge gaps by continuing the data line in the same style and color.

Set nSize to 0. Arrays for nPatt and nClr are ignored and can be omitted from the call.

nMode = 2 Bridge gaps with a line in a pattern and color of your choice. You can use the same pattern and color for all data sets, or you can select a different pattern and color for each data set.

To use the same pattern and color for all data sets, set nSize to 1.

Dimension the nPatt array for one element and store the pattern number in it. Dimension nClr for one element and store the line color.

To use different patterns for each data set, nSize must be equal to the number of data groups for the primary graph plus the number of groups for overlay graphs. Dimension nPatt and nClr to nSize elements. Store a pattern number for each group in nPatt and a line color in nClr.

Six line patterns are available. See line style constants for a list.

Colors are specified as color index numbers. See color constants for a list. If a color value of -1 is passed, the bridging line is drawn in the color of the graph line.

nMode = 3 Bridge gaps with a line in a thickness and color of your choice. You can use the same thickness and color for all data sets, or you can select a different thickness and color for each data set.

The procedure is the same as for nMode 2. Line thickness is specified in pixels. Values can range from 1 to 5.

J

Topic

AGMissingLineStyle

Related

AGAmp
AGAux
AGPatt
AGShow

AGOpen function

Initializes AutoGraph

C/C++

```
int AGOpen()
```

FoxPro

```
r = AGOpen()
```

Visual Basic

```
r% = AGOpen()
```

Return values

0	Success
-1	Failure

Description

The AGOpen function initializes the AutoGraph part of Graphics Server.

This function must be called before any other AutoGraph functions. Otherwise, those functions are rejected.

AutoGraph uses extra memory to store graph text and data arrays. When you finish using AutoGraph, use the AGClose function to release the allocated memory.

J

Topic

AGOpen

Related

AGCageStyle
AGClose
AGReset
AGShow
GSOpenServer
GSUseView

Window initialization:

GSOpenWin
GSOpenChildWin

AGPatt function

Transfers pattern information

C/C++

```
int AGPatt( int nSize, int* nPatt )
```

FoxPro

```
r = AGPatt(nSize, @nPatt(1))
```

Visual Basic

```
r% = AGPatt(nSize%, nPatt%(0))
```

Parameters

`nSize`	Size of array
`nPatt`	Pointer to pattern array (see Pattern constants)

Return values

0	Success
-1	Failure

Description

The AGPatt function transfers pattern data to AutoGraph.

The use of pattern data and the size of the array depends on the type of graph to be drawn, as shown in the following table.

Graph type	Array size	Used for
Pie	nPts	Fill pattern of each pie segment
Bar	nGroup	Fill pattern of the bar elements of each data set
Gantt	nGroup	Fill pattern of the bar elements of each data set
Line	nGroup	Line style or thickness of the line and sticks of each data set
Log/lin, lin/log, and log/log	nGroup	Line style or thickness of the line and sticks of each data set
Area	nGroup	Fill pattern of the area of each data set
Polar	nGroup	Line style or thickness of the line and sticks of each data set
Bubble	nPts	Fill pattern of each bubble
Box-whisker	nPts	Fill pattern of each box
Time series	nGroup	Defines the line style of statistical lines if they are enabled by the setting of the Aux array

Pattern data isn't used in the other graph types.

The treatment of the pattern data also depends on the type of graph, which are divided into two main categories. Treat graphs with solid surfaces, such as the pie chart and the bar graph, as the normal nPatt values of the patterns with which to draw the surfaces; treat line-based graphs as the normal nStyle or thickness values with which to draw the lines.

For line-based graphs, such as the line graph and the polar graph, the settings of the THICK and PATT options in the style parameter of AGShow determine how the pattern data is treated. For example, to draw a line graph using styled lines, the pattern array is initialized with the normal nStyle values and AGShow is called with the AGLINEPATT style option. To draw the same graph using thick lines, the pattern array is initialized with line thickness values, expressed in pixels, and AGShow is called with the AGLINETHICK style option.

Specifying pattern data is optional. If you don't specify pattern data for a graph type that uses it, AutoGraph creates a temporary internal array containing appropriate default values.

J

Topic

AGPatt

Related

AGAmp
AGAmpError
AGAux
AGClr
AGDataZ
AGDist
AGDistError
AGSym

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGRefresh3D function

Redraws True3D graph

C/C++

```
int AGRefresh3D( int nMode )
```

FoxPro

```
r = AGRefresh3D(nMode)
```

Visual Basic

```
r% = AGRefresh3D(nMode%)
```

Parameter

nMode	**Value**	**Meaning**
	0	Currently no modes implemented

Return values

0	Success
-1	Failure

Description

The AGRefresh3D function redraws a True3D graph without redrawing the titles and legends or recalculating the scale, allowing the graph to be viewed from different angles with minimum delay.

Once AGShow has been called and the graph is displayed, AGSetPerspective followed by AGRefresh3D can be called repeatedly to alter the projected view of the graph.

J

Topic

AGRefresh3D

Related

AGSetPerspective

Window initialization:

GSOpenWin

GSOpenChildWin

Graph display:

AGOpen

AGShow

AGClose

AGReset function

Resets all AutoGraph functions

C/C++

```
int AGReset()
```

FoxPro

```
r = AGReset()
```

Visual Basic

```
r% = AGReset()
```

Return values

0	Success
-1	Failure

Description

The AGReset function clears the AutoGraph data and resets all functions in preparation for a new graph.

AGReset is virtually redundant in the current edition of Graphics Server and is retained for compatibility.

J

Topic

AGReset

Related

AGOpen
AGClose
AGShow

AGSetPerspective function

Sets perspective view of True3D graphs

C/C++

```
int AGSetPerspective( int nMode, int nRotation,
                      int nElevation,
                      int nEyePosition )
```

FoxPro

```
r = AGSetPerspective(nMode, nRotation, nElevation,
                     nEyePosition)
```

Visual Basic

```
r% = AGSetPerspective(nMode%, nRotation%, nElevation%,
                      nEyePosition%)
```

Parameters

`nMode`

Constant	Value	Meaning
AG3DISO	0	Isometric projection, from infinite distance
AG3DPERSPECTIVE	1	Foreshortened perspective projection
AG3DFLIPLR	2	Enable original left wall of cage to "flip" to opposite side if data is obscured
AG3DFLIPFB	4	Enable original back wall of cage to "flip" to opposite side if data is obscured

`nRotation`

nRotation sets the angle of rotation about the Y axis. The valid range is -180 to 180 degrees (default is 0 degrees).

`nElevation`

For 3D pie charts, nElevation sets the angle of elevation relative to the default of 30 degrees. Values can range from -30 to 60. A value of 0 (default) tilts the pie 30 degrees from the horizontal. A value of -30 tilts the pie 0 degrees, resulting in a 2D view from above. A value of 60 tilts the pie 90 degrees, resulting in a 2D view of the pie's edge.

Note: 3D pie charts must be drawn with nMode = AG3DISO. Because they are not True3D, they must be redrawn with a call to AGShow(), not AGRefresh3D().

For all other 3D graphs, nElevation sets the angle

of rotation about the X axis. The valid range is -90 to 90 degrees (default is 0 degrees).

`nEyePosition` In perspective mode, nEyePosition sets the perceived viewing distance in front of the graph. The valid range is 0 to 100 in arbitrary units:

- 0 is the "furthest" viewing position, placing the viewer at a "distance" about four times the width of the viewed area.
- 100 is the "nearest" viewing position, placing the viewer at a "distance" about equal to the width of the viewed area. This distance produces the maximum distortion because of foreshortening.

In isometric mode, nEyePosition has no effect. The graph is viewed as if from an infinite distance, so all parallel lines appear parallel in the projection.

Return values

0	Success
-1	Failure

Description

The AGSetPerspective function sets the viewing position and method of projection of True3D graphs.

Topic

AGSetPerspective

Related

AGRefresh3D

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGShow function

Shows a graph

C/C++

```
int AGShow( int nGType, int nStyle, int nStats )
```

FoxPro

```
r = AGShow(nGType, nStyle, nStats)
```

Visual Basic

```
r% = AGShow(nGType%, nStyle%, nStats%)
```

Parameters

`nGType` The nGType parameter specifies the type of graph, as shown in the following table.

Constant	Value	Meaning
AGPIE2D	1	Pie chart
AGPIE3D	2	3D pie chart
AGBAR2D	3	Bar graph
AGBAR3D	4	3D bar graph
AGGANTT	5	Gantt chart
AGLINE	6	Line graph
AGLOGLIN	7	Log/lin graph
AGAREA	8	Area graph
AGSCATTER	9	Scatter graph
AGPOLAR	10	Polar graph
AGHLC	11	High-low-close graph
AGBUBBLE	12	Bubble graph
AGTAPE	13	Tape graph
AGAREA3D	14	3D area graph
AGLOGLOG	15	Log/log graph
AGLINLOG	16	Lin/log graph
AGBOXWHISKER	17	Box-whisker graph
AGSURFACE	128	True3D surface graph
AGTRUE3D	+128	True3D flag
AGTRUE3D+AGBAR3D	132	True3D bar graph
AGTRUE3D+AGSCATTER	137	True3D scatter graph
AGTRUE3D+AGTAPE	141	True3D tape graph
AGTRUE3D+AGAREA3D	142	True3D area graph
AGCOMBO	+256	Flag for overlay (combination) graph with shared Y axis
AGDUALYAXIS	+512	Flag for overlay

			(combination) graph with second Y axis

`nStyle` The nStyle parameter determines the style of the chosen graph. The options depend on the graph type, as shown in the following tables.

	Constant	Value	Meaning
Area graph	AGAREANOLABELS	1	Doesn't draw labels on axes
	AGAREALEGCLR	2	Legend text same color as symbol
	AGAREAXGRID	4	Draws X grid
	AGAREAYGRID	8	Draws y grid
	AGAREAMONO	32	Monochrome output device
	AGAREAFORECLR	64	Foreground color in high-order byte of nStats
	AGAREALOGLIN	4096	Semi-log area graph. (May be combined with AGAREAABS. Excludes AGAREAPC.)
	AGAREAPC	8192	Draw percentile area. graph
	AGAREAABS	16384	Amplitude data is absolute
Bar graph	AGBARNOLABELS	1	Doesn't draw labels on axes
	AGBARLEGCLR	2	Legend text same color as symbol
	AGBARXGRID	4	Draws X grid
	AGBARYGRID	8	Draws Y grid
	AGBARMONO	32	Monochrome output device
	AGBARFORECLR	64	Foreground color in high-order byte of nStats
	AGBARLOAT	128	Sets first element of stacked bar to be transparent
	AGBARPARETO	256	Sets Pareto format for 2D and 3D bar graphs

	AGBARLASTFIRST	512	Reverses bar order in horizontal graph
	AGBARCLUSTZ	1024	Cluster bars in the Z axis
	AGBARHORIZ	2048	Draws bars horizontally
	AGBARSTACKPC	4096	Draws stacked percentile bar graph
	AGBARSTACK	8192	Draws stacked bar graph
	AGBARCLUSTER	16384	Draws clustered bar graph
Box-whisker graph	AGBWNOLABELS	1	Doesn't draw labels on axes
	AGBWXGRID	4	Draws X grid
	AGBWYGRID	8	Draws Y grid
	AGBWMONO	32	Monochrome output device
	AGBWNOMEDIAN	512	Omits median line
	AGBWBLACKBORDER	1024	Draws a black border around box markers
	AGBWNOWHISKER	4096	Omits whisker
	AGBWNONOTCH	8192	Omits notch
	AGBWPARAMETRIC	16384	Parametric source data
Bubble graph	AGBUBNOLABELS	1	Doesn't draw labels on axes
	AGBUBLEGCLR	2	Legend text same color as bubble
	AGBUBXGRID	4	Draws X grid
	AGBUBYGRID	8	Draws Y grid
	AGBUBMONO	32	Monochrome output device
	AGBUBFORECLR	64	Foreground color in high-order byte of nStats
	AGBUBDATALABELS	16384	Adds data labels to points
Gantt chart	AGGANTTNOLABELS	1	Doesn't draw labels on axes

	AGGANTTXGRID	4	Draws X grid
	AGGANTTYGRID	8	Draws Y grid
	AGGANTTMONO	32	Monochrome output device
	AGGANTTFORECLR	64	Foreground color in high-order byte of nStats
	AGGANTTLASTFIRST	8192	Reverses bar order in horizontal graph
	AGGANTTSPACE	16384	Inserts space between adjacent bars
High-low-close	AGHLCNOLABELS	1	Doesn't draw labels on axes
	AGHLCXGRID	4	Draws X grid
	AGHLCYGRID	8	Draws Y grid
	AGHLCMONO	32	Monochrome output device
	AGHLCFORECLR	64	Foreground color in high-order byte of nStats
	AGHLCOPEN	128	Open-high-low-close
	AGHLCCANDLESTICK	256	Candlestick format
	AGHLCNOCLOSE	16384	Omits close bar
	AGHLCNOBARS	8192	Omits high and low bars
	AGHLCTHICK	4096	Uses thick lines to draw high-low-close symbol
Line graph	AGLINENOLABELS	1	Doesn't draw labels on axes
	AGLINELEGCLR	2	Legend text same color as symbol
	AGLINEXGRID	4	Draws X grid
	AGLINEYGRID	8	Draws Y grid
	AGLINESYMBOLS	16	Draws symbol at each point
	AGLINEMONO	32	Monochrome output device
	AGLINEFORECLR	64	Foreground color in high-order byte of

			nStats
	AGLINESTICK	2048	Draws vertical sticks to points
	AGLINEPATT	4096	Uses patterned lines
	AGLINETHICK	8192	Uses thick lines
	AGLINESOLID	16384	Joins points with solid lines
Log/lin, lin/log, and log/log	AGLOGLINNOLABS	1	Doesn't draw labels on axes
	AGLOGLINLEGCLR	2	Legend text same color as symbol
	AGLOGLINXGRID	4	Draws X grid
	AGLOGLINYGRID	8	Draws Y grid
	AGLOGLINSYMBOLS	16	Draws symbol at each point
	AGLOGLINMONO	32	Monochrome output device
	AGLOGLINFORECLR	64	Foreground color in high-order byte of nStats
	AGLOGLINSTICK	2048	Draws vertical sticks to points
	AGLOGLINPATT	4096	Uses patterned lines
	AGLOGLINTHICK	8192	Uses thick lines
	AGLOGLINSOLID	16384	Joins points with solid lines
Pie chart	AGPIENOLABELS	1	Doesn't draw labels on segments
	AGPIELEGCLR	2	Legend text same color as symbol
	AGPIEMONO	32	Monochrome output device
	AGPIEFORECLR	64	Foreground color in high-order byte of nStats
	AGPIESAMECLR	1024	3D pie sides same color as top
	AGPIEPCCHAR	2048	Append % symbol to pie labels
	AGPIEPERCENT	4096	Labels as % of total

			not magnitude
	AGPIESEGCLR	8192	Labels take same color as segments
	AGPIENOLINES	16384	Omits lines between labels and segment
	AGPIESMARTLABELS	32768	Auto-arranges labels to avoid overlap.
Polar graph	AGPOLARNOLABELS	1	Don't draw labels on axes
	AGPOLARLEGCLR	2	Legend text same color as symbol
	AGPOLARANGGRID	4	Draws angular grid
	AGPOLARRADGRID	8	Draw radial grid
	AGPOLARSYMBOL	16	Draws symbols at points
	AGPOLARMONO	32	Monochrome output device
	AGPOLARFORECLR	64	Foreground color in high-order byte of nStats
	AGPOLARSTICK	2048	Draws radial sticks to points
	AGPOLARPATT	4096	Uses patterned lines
	AGPOLARTHICK	8192	Uses thick lines
	AGPOLARLINE	16384	Joins points with lines
Scatter graph	AGSCATTNOLABELS	1	Doesn't draw labels on axes
	AGSCATTLEGCLR	2	Legend text same color as symbol
	AGSCATTXGRID	4	Draws X grid
	AGSCATTYGRID	8	Draws Y grid
	AGSCATTSYMBOLS	16	Draws symbol at each point
	AGSCATTMONO	32	Monochrome output device
	AGSCATTFORECLR	64	Foreground color in high-order byte of nStats
	AGSCATTSTICK	512	Draws vertical sticks (True3D only)

	AGSCATTCURVE	1024	Draws curve through each plotted data set
	AGSCATTPATT	4096	Uses patterned lines for curves (only if AGSCATTCURVE also used)
	AGSCATTTHICK	8192	Uses thick lines for curves (only if AGSCATTCURVE also used)
	AGSCATTSOLID	16384	Connects points on 3D scatter graph with solid lines.
Surface graph	AGSRFCNOLABELS	1	Doesn't draw labels on axes
	AGSRFCXGRID	4	Draws X grid
	AGSRFCYGRID	8	Draws Y grid
	AGSRFCMONO	32	Monochrome output device
	AGSRFCSIDEWALL	4096	Draws solid side walls (default none)
	AGSRFCBLACKLINES	8192	Draws connecting lines in black
	AGSRFCNET	16384	Connecting lines only (no solid fill)
Tape graph	AGTAPENOLABELS	1	Doesn't draw labels on axes
	AGTAPELEGCLR	2	Legend text same color as symbol
	AGTAPEXGRID	4	Draws X grid
	AGTAPEYGRID	8	Draws Y grid
	AGTAPEMONO	32	Monochrome output device
	AGTAPEFORECLR	64	Foreground color in high-order byte of nStats

`nStats` The nStats parameter tells AutoGraph to draw statistics lines, based on the amplitude data, overlaying the graph as shown in the following table. If nStats is zero, no statistics lines are drawn. Statistics aren't applicable to some graph types, such as the pie chart, and any specification is ignored.

Constant	Value	Meaning
AGMEAN	1	Draws mean line
AGMINMAX	2	Draws maximum and minimum lines
AGSD	4	Draws standard-deviation lines
AGLINEFIT	8	Draws best-fit line (linear regression)
AGCURVEFIT	16	Draws curve through points. You can use the AGCurveStyle function to specify one of several curve fitting algorithms.
AGCLIPGRAPH	32	Applies a clipping region to the interior of the axes in 2D graphs

Return values

0	Success
-1	Failure

Description

The AGShow function shows a graph or chart of a particular type and style, with statistical lines if specified.

Area graphs

AGAREALOGLIN in nStyle sets a semi-logarithmic (Y log, X lin) format for 2D area graphs. The default 2D style is stacked. AGAREALOGLIN can be combined with AGAREAABS to produce a semi-logarithmic area graph in absolute (unstacked) style. However, the semi-logarithmic style cannot be used with percentile area graphs. AGAREAPC and AGAREALOGLIN are mutually exclusive.

Bar graphs

The AGBARPARETO style sets Pareto format for 2D and 3D bar graphs. Bars are sorted in descending order and any attached user-defined X axis labels are sorted with the data. If there is more than one data set, bars are sorted in groups such that the first data set appears in descending order.

AGBARFLOAT in nStyle sets the first element of a stacked bar graph to be transparent, giving the appearance of the stacked bars *floating* in space. It applies only to standard stacked bar format, and there must be more than one data set.

Box-whisker graphs

The box-whisker graph is used in data analysis to illustrate the spread of values about a median. Visually each point is represented by a box with a waisted notch about the median and vertical lines or whiskers extending from the top and bottom. The notches delimit the quartiles of data. The whiskers delimit the 5th and 95th percentiles. The boxes delimit the 10th and 90th percentiles.

The data may be supplied either as an array of raw values of size nPts nGroup (with group size greater or equal to 7), which is processed to produce the percentiles across each group. Or it may be supplied as a preprocessed parametric array with a group size of 7, each group member representing one of the pre-calculated percentiles.

The boxes are patterned and colored from the nPatt and nClr arrays. Curves can be fitted to data if supplied in parametric form. By default, curves are fitted to the 50th percentile. You can select another percentile using the AGTrendDataSet function.

When you pass parametric data, you arrange groups as ascending percentiles:

Group	*%*	*Description*
0	5	5th percentile
1	10	10th percentile
2	25	25th percentile
3	50	50th percentile (median)
4	75	75th percentile
5	90	90th percentile
6	95	95th percentile

Clipping 2D graphs

The AGCLIPGRAPH mode in nStats applies a clipping region to the interior of the axes in 2D graphs. This may be advisable to mask areas of the graph that extend beyond the axes when the scale is user-defined.

Foreground colors

You can use the high-order byte of the nStats parameter to specify the foreground color in which features such as the graph axes, labels, and titles are drawn. The color number, which is a standard nClr parameter value, should be combined with the statistics options using a logical OR operation. You also have to set the foreground color option in the nStyle parameter. In the absence of a foreground color specification, AutoGraph chooses a color that contrasts with the background color. For example:

```
AGShow(AGLINE, AGLINEFORECLR, BLUE << 8 | AGMEAN | AGSD);
```

This method of setting foreground colors has been superseded by the AGFGColor function. It remains supported for users

upgrading from a previous edition of Graphics Server.

High-low-close, open-high-low-close, and candlestick graphs

High-low-close graphs normally take three groups of data, in this order: high, low, and close.

The open-high-low-close (HLCOPEN) and candlestick (HLCCANDLESTICK) forms take four groups of data, in this order: open, high, low, and close.

In candlestick format, the symbol is a rectangular box lying between the open and close values with whiskers extending to the high and low values. On ascending values (close higher than open) the box is filled with white; on descending values, it's filled with the symbol color.

Overlay graphs

A line graph can be superimposed on several graph types, using either the same Y scale as the primary graph or with a second Y axis added on the right. To draw an overlay graph, you call AGShow twice in succession.

The first call defines the primary graph. A bit must be set in the nGType parameter by adding AGCOMBO to the graph type. No graph is drawn until the second call to AGShow, which defines how the overlay graph is combined.

The primary graph can be one of the following types:

- 2D vertical bar graph: AGBAR2D
- 2D line graph: AGLINE
- 2D area graph: AGAREA
- 2D high-low-close graph: AGHLC
- 2D scatter graph: AGSCATTER

The normal styles apply, as set by nStyle.

The overlay graph must be of type AGLINE. This graph may either share the same axis and scale as the first one, or be drawn to its own scale with a separate Y axis on the right. To specify a second Y axis, add AGDUALYAXIS to the nGType parameter. (You can customize the second axis using the AGYRAxisStyle function.)

Note that the primary graph is dominant in determining the options for labels, axes, and grids. Style settings for these features are ignored in the second call to AGShow.

In creating the two graphs, you supply nGroup sets of data. The last group applies to the overlay line graph, and the preceding groups apply to the primary graph. For example, if the primary graph is a high-low-close graph, you supply four data groups: the first is the high values, the second the low values, the third the close values, and the fourth the values for the overlay line graph.

The second graph takes its symbol, color, and line pattern from

the last element in those attribute arrays (as defined by the AGSym, AGClr, and AGPatt functions). Construct the attribute arrays as if the primary graph was being drawn on its own, adding a final element for the overlay graph. If the primary graph doesn't require one of the attribute arrays, you still have to supply it, using a single element for the overlay graph.

The following AGShow calls draw a stacked bar graph with an overlay line graph. The line graph is drawn to its own scale (with a separate Y axis on the right), with symbols, and with a best-fit line superimposed:

```
AGShow(AGBAR2D+AGCOMBO+AGDUALYAXIS, AGBARSTACK, 0);
AGShow(AGLINE, AGLINESYMBOLS, AGLINEFIT);
```

Pie charts

AGPIESMARTLABELS in nStyle enables auto-arranging of labels to avoid overlapping. Note that the algorithm may cause labels to extend outside the visible area of the graph if too many are concentrated in a small sector of the pie.

AGPIE3D in nGType draws a 3D pie chart in isometric projection. The pie's depth may be adjusted with AG3Dstyle(), and the angle of elevation may be adjusted with AGSetPerspective(). However, 3D pies are not True3D graphs, and so they are always redrawn with AGShow() and not with AGRefresh3D().

True3D graphs

True3D graphs are 3D graphs capable of being rotated and viewed from any angle.

Select True3D graphs by adding the constant AGTRUE3D to the graph type in nGType. Not all graph types can be viewed in True3D. See the table for valid types.

The viewing angle and 3D projection mode is set by AGSetPerspective.

The function AGRefresh3D is used to redraw the graph with a different viewing angle without the overhead of recalculating and redrawing the titles, labels, and legend.

True3D graphs don't support superimposed statistics, curve fitting, or overlay graphs.

The same arrays are required for True3D graphs as for their pseudo-3D counterparts, the exception being the scatter graph, which requires a Z data array created using the same rules as the Dist array.

The surface graph requires multi-dimensional amplitude data and no distance or Z data, because it's always plotted at regular increments in X and Z.

True3D scatter graphs

When you draw a True3D scatter graph, you have to supply Z

data using the AGDataZ function.

Scatter graphs are the only True3D graphs that label the Z axis of the 3D cage with data values. You can use the AGZAxisStyle function to customize the axis.

You can connect points with solid lines in True3D scatter graphs by using AGSCATTSOLID in nStyle.

J

Topic

AGShow

Related

AGOpen

AGClose

AGTrendDataSet

Window initialization:

GSOpenWin

GSOpenChildWin

AGSurfaceClr function

Sets colors for True3D surface graph

C/C++

```
int AGSurfaceClr( int nColorMin, int nColorMax,
                  int nColorSide )
```

FoxPro

```
r = AGSurfaceClr(nColorMin, nColorMax, nColorSide)
```

Visual Basic

```
r% = AGSurfaceClr(nColorMin%, nColorMax%, nColorSide%)
```

Parameters

`nColorMin`	Color index at minimum data value
`nColorMax`	Color index at maximum data value
`nColorSide`	Color of side wall

Return values

0	Success
-1	Failure

Description

The AGSurfaceClr function sets the range of colors used to color the panels or lines of a surface graph, as well as the color of the optional side walls.

Surface graphs are colored according to the height of points above the origin, with nColorMin setting the color at the origin and nColorMax the color at the maximum height of the Y axis. Colors at intermediate heights are interpolated between these two values.

You should use the GSSelectPalette function to enable a 128-entry color palette and select a graded range of colors extending from 32 to 127 (0 to 31 are reserved for the standard colors). To define your own palette, you can use the GSSetPal function.

J

Topic

AGSurfaceClr

Related

AG3DStyle
AGCageStyle
AGClr
GSSelectPalette
GSSetPal

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen

AGShow

AGClose

AGSym function

Transfers symbol information

C/C++

```
int AGSym( int nSize, int* nSymbol )
```

FoxPro

```
r = AGSym(nSize, @nSymbol(1))
```

Visual Basic

```
r% = AGSym(nSize%, nSymbol%(0))
```

Parameters

`nSize`	Size of array
`nSymbol`	Pointer to array of symbol data (see Symbol constants)

Return values

0	Success
-1	Failure

Description

The AGSym function transfers symbol data to AutoGraph.

Symbol data is used only in line, logarithmic, scatter, polar, and time series graphs, and only when you specify an appropriate style in the nStyle parameter of the AGShow function.

Graph type	Array size	Used for
Line	nGroup	Symbol of the points of each data set
Log/lin, lin/log, and log/log	nGroup	Symbol of the points of each data set
Scatter	nGroup	Symbol of the points of each data set
Polar	nGroup	Symbol of the points of each data set
Time series	nGroup	Set symbol as other graphs

Specifying symbol data is optional. If you don't, AutoGraph creates a temporary internal array containing appropriate default values.

Example

You can modify the size of the symbols from their default of 2.5% of the height of the view by calling the GSSizeSymbol function prior to calling AGShow. For example, the following code will double the size of the symbols on the graph by increasing their size to 5% of the height of the view.

```
GSSizeSymbol(50);
AGShow(AGLINE, AGLINESYMBOLS, 0);
```

J

Topic
AGSym

Related
AGAmp
AGAmpError
AGAux
AGClr
AGDataZ
AGDist
AGDistError
AGPatt

Window initialization:
GSOpenWin
GSOpenChildWin

Graph display:
AGOpen
AGShow
AGClose

AGTimeGraph function

Begins time series graph

C/C++

```
int AGTimeGraph( int nPts, int nGroup, double fDataMax,
                 double fDataMin, int nStyle )
```

FoxPro

```
r = AGTimeGraph(nPts, nGroup, fDataMax, fDataMin,
                nStyle)
```

Visual Basic

```
r% = AGTimeGraph(nPts%, nGroup%, fDataMax#, fDataMin#,
                 nStyle%)
```

Parameters

`nPts` Number of points to display

`nGroup` Number of concurrent sets of data

`fDataMax` Maximum expected value of data

`fDataMin` Minimum expected value of data

`nStyle`

Constant	Value	Meaning
AGTIGNOLABELS	1	No labels
AGTIGLEGCLR	2	Legend text same color as symbols
AGTIGXGRID	4	X grid
AGTIGYGRID	8	Y grid
AGTIGLINES	16	Continuous line (no symbols)

`nAux`

Constant	Value	Meaning
AGTIGMEAN	1	Superimposed mean
AGTIGSTDDEV	2	Superimposed standard deviation

Return values

0 Success

-1 Failure

Description

The AGTimeGraph function initializes a time series graph, drawing the axes, legend, and titles.

A time series graph comprises nGroup sets of data that are displayed concurrently. At first no data is displayed. As data is added, using the AGTimeUpdate function, the new points are drawn at the origin on the right and previous points are scrolled to the left. When the number of points on display exceeds nPts, the earliest point is discarded. The result is an animated, scrolling graph that accepts continuous data.

When you use this function, you should not call AGShow. Instead, define titles, labels, and legends as usual before calling AGTimeGraph at the position in the code where AGShow would be

called, followed by the call to AGClose. AGTimeUpdate can be called at any time thereafter with the appropriate window selected.

The positive- and negative-going axes are scaled to show data in the range fDataMin to fDataMax. You have to specify these values from the outset, because no data is available for calculating a scale when the graph is first drawn.

The graph style (as would normally be defined by the AGShow function) is defined by nStyle. By default, the graph is drawn with symbols marking data points. These are subject to the AGSym function. AGTIGLINES in nStyle draws the graph with continuous lines instead of symbols. No line patterns are available.

Also by default, the graph is drawn with a main title, X and Y titles, X axis, Y axis, legend, and X and Y labels. These are subject to the normal AG text and style functions.

One time series graph can display several sets of data concurrently.

Symbol, pattern, color, aux, and optionally distance arrays must be supplied using the AG array calls prior to calling AGTimeGraph. These arrays are all of dimension nGroup and specify the attributes of each data set.

The symbol array defines the symbol drawn at each data point. The color array defines the color of the symbol and of any statistical lines. The pattern array defines the line style of the mean statistical line. The standard deviation lines are drawn with the pattern index + 1. The distance array defines the vertical offset of the data set from the origin, enabling sets to be drawn at different origins. The Aux array defines which statistical lines are to be overlaid.

The function uses XORing to produce fast animation. This has several side effects:

The color of a symbol is the result of XORing itself with whatever lies beneath. The color index must be one of the basic colors, 0-16. This color is adjusted such that the result of XORing, with the current background color, produces the color as specified. If the symbols are XORed onto any area not in the background color, the result is unpredictable.

The metafile must be turned off to prevent the whole animated history of the graph redisplaying when the window is moved or uncovered.

To overcome the latter problem, time series graphs are always drawn using blit mode. After a time series graph has concluded, it may be necessary to reset the drawing mode to metafile by calling GSWinPaint(5)

There can be only one time series graph per window.

Example

The following example shows how to create a time series graph with two concurrent data sets. Twenty points are to be displayed in the range -5 to +10 units. The mean of data is superimposed

on the first graph and the standard-deviation on the second.

```
#define   NUMPOINTS   20  // points on display
#define   NUMSETS     2   // concurrent sets
AGOpen();                 // open AG
AGTitleG("Time series graph"); // Graph title
//... set bottom and left titles
lpnClr(0) = RED;lpnClr(1) = BLUE;// set graph colors
//... set symbols and patterns for each data set
lpnAux(0) = AGTIGMEAN;    // show mean on 1st graph
lpnAux(1) = AGTIGSTDDEV;  // show std-dev on 2nd
AGLegend(NUMSETS, lpszLegend); // legend text
AGClr(NUMSETS, lpnClr);   // color array
AGSym(NUMSETS, lpnSym);   // symbol array
AGPatt(NUMSETS, lpnPatt); // line style array
AGAux(NUMSETS, lpnAux);   // aux array
AGTimeGraph(NUMPOINTS, NUMSETS, 10,-5, AGTIGXGRID);
AGClose();
//... at any time later call AGTimeUpdate
```

J

Topic
AGTimeGraph

Related
AGAmp
AGAux
AGDataZ
AGDist
AGPatt
AGSym
AGTimeUpdate
GSWinPaint

Window initialization:
GSOpenWin
GSOpenChildWin

Graph display:
AGOpen
AGShow
AGClose

AGTimeUpdate function

Updates time series graph with data

C/C++

```
int AGTimeUpdate( int nMode, int nGroup, double* fData )
```

FoxPro

```
r = AGTimeUpdate(nMode, nGroup, @fData(1))
```

Visual Basic

```
r% = AGTimeUpdate(nMode%, nGroup%, fData#(0))
```

Parameters

`nMode`

Constant	Value	Meaning
AGTIGUPDATE	0	Loads new data and redraws graph
AGTIGLOAD	1	Loads new data without redrawing graph
AGTIGHIDEDATA	2	Loads no data and hides graph
AGTIGSHOWDATA	3	Loads no data and redraws graph

Mode 0 is the normal method of loading new data and showing it, scrolling the graph to the left.

Modes 1, 2, and 3 let you perform "batch updates" to add several data points without refreshing the graph. To do a batch update, you hide the graph (mode 2), add new data (repeat mode 1), then redraw the graph (mode 3). This gives you faster updates, but there's some flickering.

`nGroup` Number of data groups

`fData` Array of data, one value for each group

Return values

0	Success
-1	Failure

Description

The AGTimeUpdate function updates a time series graph as previously created using the AGTimeGraph function.

Example

This example follows the previous example for AGTimeGraph. Here we update the graph with new data, passing values for the two data sets. Since this operation is executed at some arbitrary time after the graph is created, we must ensure that the appropriate target window is re-selected.

```
double    fData [NUMSETS];
```

```
fData[0] = fData0;   // new data for group 0
fData[1] = fData1;   // new data for group 1
GSUseView(nTIGWin, nTIGView); // select our window
AGTimeUpdate(AGTIGUPDATE, NUMSETS, fData); // update
```

J

Topic
AGTimeUpdate

Related
AGTimeGraph
GSWinPaint

AGTitleBG function

Sets title style and background color

C/C++

```
int AGTitleBG( int nMode, int nClr )
```

FoxPro

```
r = AGTitleBG(nMode, nClr)
```

Visual Basic

```
r% = AGTitleBG(nMode%, nClr%)
```

Parameters

`nMode`

Constant	Value	Meaning
AGTTLG	0	Selects graph title
AGTTLX	1	Selects bottom title
AGTTLYLEFT	2	Selects left title
AGTTLYRIGHT	3	Selects right title
AGTTLUP	4	Draws title upwards (rotated 90 degrees counterclockwise)
AGTTLDOWN	8	Draws title downwards (rotated 90 degrees clockwise)
AGTTLFRAME	16	Black border
AGTTLFILL	32	Fills background with nClr
AGTTLDROPSHADOW	64	Drop shadow
AGTTLRAISED	128	Raised border
AGTTLLOWERED	256	Lowered border

Modes 64, 128, and 256 can't be combined.

`nClr` Text color (see Color constants)

Return values

0	Success
-1	Failure

Description

The AGTitleBG function sets the background color and style of the graph title, left title, right title, and bottom title. The function must be called repeatedly to set each title style. Rotated (vertical) text is only allowed for left and right titles.

J

Topic

AGTitleBG

Related

AGFGColor
AGFontStyle
AGLabels
AGLabelY
AGLabelZ
AGLegendStyle
AGTitleG
AGTitleX
AGTitleY
AGTitleYR

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGTitleG function

Defines graph title

C/C++

```
int AGTitleG( char* szTitle )
```

FoxPro

```
r = AGTitleG(szTitle)
```

Visual Basic

```
r% = AGTitleG(szTitle$)
```

Parameters

`szTitle`	Title string

Return values

0	Success
-1	Failure

Description

The AGTitleG function defines the title to be placed centrally above the graph.

J

Topic

AGTitleG

Related

AGFontStyle
AGLabels
AGLabelY
AGLabelZ
AGLegendStyle
AGTitleBG
AGTitleX
AGTitleY
AGTitleYR
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGTitleX function

Defines bottom title for graph

C/C++

```
int AGTitleX( char* szTitle )
```

FoxPro

```
r = AGTitleX(szTitle)
```

Visual Basic

```
r% = AGTitleX(szTitle$)
```

Parameters

`szTitle`	Title string

Return values

0	Success
-1	Failure

Description

The AGTitleX function defines the graph's bottom title, which is placed at the bottom of a graphing window. The bottom title is frequently used to explain the X axis.

J

Topic

AGTitleX

Related

AGFontStyle
AGLabels
AGLabelY
AGLabelZ
AGLegendStyle
AGTitleBG
AGTitleG
AGTitleY
AGTitleYR
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGTitleY function

Defines left title for graph

C/C++

```
int AGTitleY( char* szTitle )
```

FoxPro

```
r = AGTitleY(szTitle)
```

Visual Basic

```
r% = AGTitleY(szTitle$)
```

Parameters

`szTitle`	Title string

Return values

0	Success
-1	Failure

Description

The AGTitleY function defines the graph's left title, which is placed at the left edge of a graphing window. The left title is frequently used to explain the Y axis.

J

Topic

AGTitleY

Related

AGFontStyle
AGLabels
AGLabelY
AGLabelZ
AGLegendStyle
AGTitleBG
AGTitleG
AGTitleX
AGTitleYR
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGTitleYR function

Defines right title for graph

C/C++

```
int AGTitleYR( char* szTitle )
```

FoxPro

```
r = AGTitleYR(szTitle)
```

Visual Basic

```
r% = AGTitleYR(szTitle$)
```

Parameters

`szTitle`	Title string

Return values

0	Success
-1	Failure

Description

The AGTitleYR function defines the graph's right title, which is placed at the right edge of a graphing window. The right title is frequently used to explain the right-hand Y axis used in some overlay graphs.

J

Topic

AGTitleYR

Related

AGFontStyle
AGLabels
AGLabelY
AGLabelZ
AGLegendStyle
AGTitleBG
AGTitleG
AGTitleX
AGTitleY
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGTrendDataSet function

Applies trend lines to individual data sets

C/C++

```
int AGTrendDataSet(int nSize, int *nEnable)
```

FoxPro

```
r = AGTrendDataSet(nSize, @nEnable(nSize))
```

Visual Basic

```
r% = AGTrendDataSet(nSize%, nEnable%(nSize%))
```

Parameters

`nSize` Number of elements in nEnable array. Normally there will be one element for each set of amplitude data.

`nEnable` An array of flags indicating which trend lines to draw for each data set. If an element of the array has a value of zero, no lines will be drawn for that set.

The flags listed below may be used in combination.

Constant	Value	Meaning
AGMEAN	1	Draws a mean line
AGMINMAX	2	Draws maximum and minimum lines
AGSD	4	Draws standard-deviation lines
AGLINEFIT	8	Draws a best-fit line (linear regression)
AGCURVEFIT	16	Draws a curve through points.

Return values

0	Success
-1	Failure

Description

Normally, statistical lines are drawn according to the value passed in the nStats parameter of AGShow() . The disadvantage of this method is that the same lines are drawn for all data sets. AGTrendDataSet lets you enable different lines for each set of data or enable lines for some sets while omitting them for others.

If AGTrendDataSet() is **not** called, the value of nStats in AGShow() determines which trend lines, if any, are drawn.

If AGTrendDataSet() is called, each element of the nEnable array selects one or more trend lines to be applied to the associated data set.

If you select AGCURVEFIT, you can use the AGCurveStyle function to specify one of several curve fitting algorithms. However, the selected algorithm is applied to all curves on the graph; it is not possible to derive curves differently for separate data sets.

J

Topic

AGTrendDataSet

Related

AGAmp
AGCurveStyle
AGShow

AGXAxisStyle function

Sets X axis style

C/C++

```
int AGXAxisStyle( int nMode, int nTicks, int nLabEvery,
                  double fMax, double fMin )
```

FoxPro

```
r = AGXAxisStyle(nMode, nTicks, nLabEvery, fMax, fMin)
```

Visual Basic

```
r% = AGXAxisStyle(nMode%, nTicks%, nLabEvery%, fMax#, fMin#)
```

Parameters

`nMode`

Constant	Value	Meaning
AGVARORIGIN	1	Variable origin (not necessarily 0)
AGNOLABELS	2	No labels displayed
AGUTICKS	4	Number of ticks value is present
AGUMAX	8	Axis maximum value is present
AGUMIN	16	Axis minimum value is present
AGLABEVERY	32	Labels every nth point value is present
AGNOTICKS	64	No ticks displayed
AGTICKEVERY	128	Ticks every nth point value is present
AGAXISTOP	256	Draws the X axis at the top of the graph
AGAXISBOTTOM	512	Draws the X axis at the bottom of the graph
AGAXISMINORTICK	1024	Draws minor ticks and grids
AGAXISVERTLABELS	2048	Draws X labels vertically
AGTICKIN	4096	Tick marks drawn inside the axis (default is through)
AGTICKOUT	8192	Tick marks drawn outside the axis (default is through)

`nTicks` Number of ticks

`nLabEvery` Labels every nth point

`fMax` Axis maximum value

`fMin`	Axis minimum value

Return values

0	Success
-1	Failure

Description

The AGXAxisStyle function specifies the style in which the X axis is scaled, ticked, and labeled.

There's no need to specify all the parameters when you use the AGXAxisStyle function; they can be left as 0. The AGUTICKS, AGUMAX, AGUMIN, and AGLABEVERY nMode bits specify whether the related parameters are to be used. AutoGraph will use default values for omitted parameters.

The AGUTICKS and AGTICKEVERY options are mutually exclusive and share the nTicks parameter as the means of specifying their respective values. For example, to specify 10 ticks for an axis, you set the nMode option AGUTICKS and set the parameter nTicks to 10. To specify one tick for every five points on the axis, you set the nMode option AGTICKEVERY and set nTicks to 5.

By default, tick marks are drawn through the axis. AGTICKIN in nMode causes tick marks to be drawn inside the axis. AGTICKOUT in nMode causes them to be drawn outside the axis. AGTICKIN and AGTICKOUT are mutually exclusive.

J

Topic

AGXAxisStyle

Related

AGAxisMinorTicks
AGGridStyle
AGYAxisStyle
AGYRAxisStyle
AGZAxisStyle

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGYAxisStyle function

Sets Y axis style

C/C++

```
int AGYAxisStyle( int nMode, int nTicks, int nLabEvery,
                  double fMax, double fMin )
```

FoxPro

```
r = AGYAxisStyle(nMode, nTicks, nLabEvery, fMax, fMin)
```

Visual Basic

```
r% = AGYAxisStyle(nMode%, nTicks%, nLabEvery%, fMax#,
                  fMin#)
```

Parameters

`nMode`

Constant	Value	Meaning
AGVARORIGIN	1	Variable origin (not necessarily 0)
AGNOLABELS	2	No labels displayed
AGUTICKS	4	Number of ticks required is present
AGUMAX	8	Axis maximum value is present
AGUMIN	16	Axis minimum value is present
AGNOTICKS	64	No ticks displayed
AGAXISLEFT	256	Draws the Y axis at the left of the graph
AGAXISRIGHT	512	Draws the Y axis at the right of the graph
AGAXISMINORTICK	1024	Draws minor ticks and grids
AGTICKIN	4096	Tick marks drawn inside the axis (default is through)
AGTICKOUT	8192	Tick marks drawn outside the axis (default is through)

`nTicks`	Number of ticks
`nLabEvery`	Label every nth tick (option currently not used)
`fMax`	Axis maximum value
`fMin`	Axis minimum value

Return values

0	Success
-1	Failure

Description

The AGYAxisStyle function specifies the style in which the Y axis is scaled, ticked, and labeled.

There's no need to specify all the parameters when you use the AGYAxisStyle function; they can be left as 0. The AGUTICKS, AGUMAX, and AGUMIN nMode bits specify whether the related parameters are to be used. AutoGraph will use default values for omitted parameters.

By default, tick marks are drawn through the axis. AGTICKIN in nMode causes tick marks to be drawn inside the axis. AGTICKOUT in nMode causes them to be drawn outside the axis. AGTICKIN and AGTICKOUT are mutually exclusive.

J

Topic

AGYAxisStyle

Related

AGAxisMinorTicks
AGGridStyle
AGXAxisStyle
AGYRAxisStyle
AGZAxisStyle
AGLabelY

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGYRAxisStyle function

Sets right-hand Y axis style

C/C++

```
int AGYRAxisStyle( int nMode, int nTicks,
                   int nLabEvery, double fMax,
                   double fMin )
```

FoxPro

```
r = AGYRAxisStyle(nMode, nTicks, nLabEvery, fMax, fMin)
```

Visual Basic

```
r% = AGYRAxisStyle(nMode%, nTicks%, nLabEvery%, fMax#, fMin#)
```

Parameters

`nMode`

Constant	Value	Meaning
AGVARORIGIN	1	Variable origin (not necessarily 0)
AGNOLABELS	2	No labels displayed
AGUTICKS	4	Number of ticks required is present
AGUMAX	8	Axis maximum value is present
AGUMIN	16	Axis minimum value is present
AGNOTICKS	64	No ticks displayed
AGAXISLEFT	256	Draws the Y axis at the left of the graph
AGAXISRIGHT	512	Draws the Y axis at the right of the graph
AGTICKIN	4096	Tick marks drawn inside the axis (default is through)
AGTICKOUT	8192	Tick marks drawn outside the axis (default is through)

`nTicks`	Number of ticks
`nLabEvery`	Label every nth tick (option currently not used)
`fMax`	Axis maximum value
`fMin`	Axis minimum value

Return values

0	Success
-1	Failure

Description

The AGYRAxisStyle function specifies the style in which the right-hand Y axis is scaled, ticked, and labeled in combination graphs with a dual axes.

There's no need to specify all the parameters when you use the AGYRAxisStyle function; they can be left as 0. The AGUTICKS, AGUMAX, and AGUMIN nMode bits specify whether the related parameters are to be used. AutoGraph will use default values for omitted parameters.

By default, tick marks are drawn through the axis. AGTICKIN in nMode causes tick marks to be drawn inside the axis. AGTICKOUT in nMode causes them to be drawn outside the axis. AGTICKIN and AGTICKOUT are mutually exclusive.

J

Topic

AGYRAxisStyle

Related

AGAxisMinorTicks
AGGridStyle
AGXAxisStyle
AGYAxisStyle
AGZAxisStyle
AGLabelY

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

AGZAxisStyle function

Sets Z axis style

C/C++

```
int AGZAxisStyle( int nMode, int nTicks, int nLabEvery,
                  double fMax, double fMin )
```

FoxPro

```
r = AGZAxisStyle(nMode, nTicks, nLabEvery, fMax, fMin)
```

Visual Basic

```
r% = AGZAxisStyle(nMode%, nTicks%, nLabEvery%, fMax#, fMin#)
```

Parameters

`nMode`

Constant	Value	Meaning
AGVARORIGIN	1	Variable origin (not necessarily 0)
AGNOLABELS	2	No labels displayed
AGUTICKS	4	Number of ticks value is present
AGUMAX	8	Axis maximum value is present
AGUMIN	16	Axis minimum value is present
AGLABEVERY	32	Label every nth point value is present
AGNOTICKS	64	No ticks displayed
AGTICKEVERY	128	Tick every nth point value is present
AGTICKIN	4096	Tick marks drawn inside the axis (default is through)
AGTICKOUT	8192	Tick marks drawn outside the axis (default is through)

`nTicks`	Number of ticks
`nLabEvery`	Label every nth tick
`fMax`	Axis maximum value
`fMin`	Axis minimum value

Return values

0	Success
-1	Failure

Description

The AGZAxisStyle function specifies the style in which the Z axis is scaled, ticked, and labeled in True3D graphs (currently this function is only of relevance to scatter graphs).

There's no need to specify all the parameters when you use the

AGZAxisStyle function; they can be left as 0. The AGUTICKS, AGUMAX, AGUMIN, and AGLABEVERY nMode bits specify whether the related parameters are to be used. AutoGraph will use default values for omitted parameters.

The AGUTICKS and AGTICKEVERY options are mutually exclusive and share the nTicks parameter as the means of specifying their respective values. For example, to specify that the axis will have 10 ticks, you set the nMode option AGUTICKS and set the parameter nTicks to 10. To specify that every five points on the axis will be ticked, you set the nMode option AGTICKEVERY and set the parameter nTicks to 5.

By default, tick marks are drawn through the axis. AGTICKIN in nMode causes tick marks to be drawn inside the axis. AGTICKOUT in nMode causes them to be drawn outside the axis. AGTICKIN and AGTICKOUT are mutually exclusive.

J

Topic

AGZAxisStyle

Related

AGAxisMinorTicks
AGCageStyle
AGGridStyle
AGXAxisStyle
AGYAxisStyle
AGLabelZ

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

GSArc function

Draws circular arc

C/C++

```
GSArc( double fxOrg, double fyOrg, double fRadius, double
fAng1, double fAng2, int nMode, int nStyle, int nClr )
```

FoxPro

```
r = GSArc(fxOrg, fyOrg, fRadius, fAng1, fAng2, nMode, nStyle,
nClr)
```

Visual Basic

```
r% = GSArc(fxOrg#, fyOrg#, fRadius#, fAng1#, fAng2#, nMode%,
nStyle%, nClr%)
```

Parameters

`fxOrg` X center

`fyOrg` Y center

`fRadius` Radius

`fAng1` Start angle

`fAng2` End angle

`nMode`

Constant	Value	Meaning
AARADIUS	1	Draws connecting radii at extremes of arc
AAFILL	2	Fills the arc with pattern; closing radii are automatically added to the arc to define fill area
AATHICK	4	Uses thick line style

Modes AAFILL and AATHICK are exclusive. If you enable both of them, only AAFILL is used.

`nStyle` Line thickness (nMode AATHICK) or fill pattern (mode AAFILL). See Line style constants or Pattern constants.

`nClr` Color of arc (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSArc function draws a circle or circular arc. The arc is drawn from a start angle (fAng1) to an end angle (fAng2) in a counterclockwise direction and in a continuous line.

Optionally, you can add radial lines drawn from the center (were the arc a complete circle) to the arc's end points. Also optionally, you can fill the arc with a pattern defined by nStyle.

J

Topic

GSArc

Related

GSCircle
GSEllipse
GSPolyFill
GSPolyVec

GSArea function

Draws 2D area graph

C/C++

```
GSArea( double fxOrg, double fyOrg, double fInc,
        double fHt, int nMode, int nGroup )
```

FoxPro

```
r = GSArea(fxOrg, fyOrg, fInc, fHt, nMode, nGroup)
```

Visual Basic

```
r% = GSArea(fxOrg#, fyOrg#, fInc#, fHt#, nMode%,
            nGroup%)
```

Parameters

`fxOrg`	X origin		
`fyOrg`	Y origin		
`fInc`	X increment		
`fHt`	Height of graph in percentile mode		
`nMode`	**Constant**	**Value**	**Meaning**
	ARABS	1	Data is in absolute units (default is relative)
	ARVARX	2	Uses fD array for X position
	ARPC	4	Percentile mode
`nGroup`	Number of grouped data sets		

Return values

0	Success
-1	Failure

Description

The GSArea function draws a 2D area graph from one or more sets of data.

Amplitude data may be relative or absolute. Absolute data defines the height of each dividing line above the X axis. Relative data defines the height of each line above the preceding one.

The areas are filled with the patterns and colors defined in the respective group arrays.

The data may be graphed either at fixed increments in X as defined by fInc or using the individual X values passed in the fD array.

In the percentile mode, the graph occupies the height, fHt, divided into areas on the basis of the individual amplitude values as a percentage of the sum of the data sets at each point.

GSDataTrans parameters for 2D area graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions)
fD[nPts]	Pointer to distance array (X positions)--used only with nMode ARVARX
nPatt[nGroup]	Pointer to array containing fill patterns for single area plot
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nGroup]	Pointer to array containing color for single area plot

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions)--used only with nMode ARVARX
nPatt[nGroup]	Pointer to array containing fill patterns for successive area plots
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nGroup]	Pointer to array containing colors for successive area plots

* GSDataTrans can't pass two-dimensional fD arrays. You have to use the GSDataDist function if you want to specify individual fD values for each data set. However, you can use GSDataTrans if you want to apply the same fD values to points in all sets.

J

Topic
GSArea

Related
GSArea3D
GSBar2D
GSTapeGraph

Axis/grid/legend:

GSAxis
GSGrid
GSLegend

Labels:
GSDataLabels
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:
GSDataTrans

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSArea3D function

Draws 3D area graph

C/C++

```
GSArea3D( double fxOrg, double fyOrg, double fInc,
          double fHt, double fDepth, double fAng,
          int nMode )
```

FoxPro

```
r = GSArea3D(fxOrg, fyOrg, fInc, fHt, fDepth, fAng, nMode)
```

Visual Basic

```
r% = GSArea3D(fxOrg#, fyOrg#, fInc#, fHt#, fDepth#,
              fAng#, nMode%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	X increment
`fHt`	Height of graph in percentile mode
`fDepth`	Perspective depth of graph
`fAng`	Perspective angle from the horizontal

`nMode`

Constant	Value	Meaning
ARABS	1	Data is in absolute format (default is relative)
ARVARX	2	Uses fD array for X position
ARPC	4	Percentile mode

Return values

0	Success
-1	Failure

Description

The GSArea3D function draws a 3D area graph showing one or more sets of data. This isn't True3D, but the quasi-3D of previous editions of Graphics Server. True3D graphs can be programmed only through the AutoGraph API.

Amplitude data may be relative or absolute. Absolute data defines the height of each dividing line above the X axis. Relative data defines the height of each line above the preceding one.

In relative and percentile mode, the graph is drawn in one plane, with successive data sets stacked one above the other. In absolute mode, the graph acquires Z-axis perspective, with successive data sets advancing from the back toward the front of the graph.

The areas are filled using the patterns and colors defined in the respective arrays. The side faces of the areas are solid filled in the half-tone colors of the fronts.

The data may be graphed either at fixed increments in X, as defined by fInc or using the individual X values passed in the fD array.

In the percentile mode, the graph occupies the height, fHt, divided into areas on the basis of the individual amplitude values as a percentage of the sum of the data sets at each point.

GSDataTrans parameters for 3D area graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions)
fD[nPts]	Pointer to distance array (X positions)--used only with nMode ARVARX
nPatt[nGroup]	Pointer to array containing fill patterns for single area plot
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nGroup]	Pointer to array containing color for single area plot

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions)--used only with nMode ARVARX
nPatt[nGroup]	Pointer to array containing fill patterns for successive area plots
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nGroup]	Pointer to array containing colors for successive area plots

* GSDataTrans can't pass two-dimensional fD arrays. You have to use the GSDataDist function if you want to specify individual fD values for each data set. However, you can use GSDataTrans if you want to apply the same fD values to points in all sets.

Topic

GSArea3D

Related

GSArea
GSBar2D
GSBar3D
GSTapeGraph

Axis/cage/legend:

GSCage3D
GSLegend

Labels:

GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:

GSDataTrans

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

GSAreaLogLin function

Draws a 2D area graph with semi-log scaling

C/C++

```
int GSAreaLogLin(double fxOrg, double fyOrg,
                 double fInc, double fCycleHt,
                 double fBaseVal, int nMode)
```

FoxPro

```
r = GSAreaLogLin(fxOrg, fyOrg, fInc, fCycleHt,
                 fBaseVal, nMode)
```

Visual Basic

```
r% = GSAreaLogLin(fxOrg#, fyOrg#, fInc#, fCycleHt#,
                  fBaseVal#, nMode%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	X increment
`fCycleHt`	Height of one cycle (log base 10)
`fBaseVal`	Base value of graph at y = 0
`nMode`	

Constant	Value	Meaning
ARABS	1	Data is in absolute units (default is relative)
ARVARX	2	Uses fD array for X positions

Return values

0	Success
-1	Failure

Description

The GSAreaLogLin function draws a semi-logarithmic 2D area graph.

Amplitude data may be relative or absolute. Absolute data defines the height of each dividing line above the X axis. Relative data defines the height of each line above the preceding one.

The areas are filled with the patterns and colors defined in the respective group arrays.

The data may be graphed either at fixed increments in X as defined by fInc or using the individual X values passed in the fD array.

GSDataTrans parameters

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions)
fD[nPts]	Pointer to distance array (X positions)--used only with nMode ARVARX

nPatt[nGroup]	Pointer to array containing fill patterns for single area plot
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nGroup]	Pointer to array containing color for single area plot

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions)--used only with nMode ARVARX
nPatt[nGroup]	Pointer to array containing fill patterns for successive area plots
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nGroup]	Pointer to array containing colors for successive area plots

* GSDataTrans can't pass two-dimensional fD arrays. You have to use the GSDataDist function if you want to specify individual fD values for each data set. However, you can use GSDataTrans if you want to apply the same fD values to points in all sets.

J

Topic
GSAreaLogLin

Related
GSArea
GSDataTrans
GSLogLin

GSArrow function

Draws arrow

C/C++

```
int GSArrow( double fxA, double fyA, double fxB,
             double fyB, double fHeadLen, int nMode,
             int nStyle, int nClr )
```

FoxPro

```
r = GSArrow(fxA, fyA, fxB, fyB, fHeadLen, nMode,
            nStyle, nClr)
```

Visual Basic

```
r% = GSArrow(fxA#, fyA#, fxB#, fyB#, fHeadLen#, nMode%,
             nStyle%, nClr%)
```

Parameters

`fxA`	X start		
`fyA`	Y start		
`fxB`	X end		
`fyB`	Y end		
`fHeadLen`	Length of head		
`nMode`	**Constant**	**Value**	**Meaning**
	AWTHIN	1	Thin head
	AWMEDIUM	0	Medium-width head
	AWTHICK	2	Thick head
	AWOPEN	4	Open head
	AWCLOSED	0	Closed head
`nStyle`	Line pattern or thickness (see Line style constants)		
`nClr`	Color of arrow (see Color constants)		

Return values

0	Success
-1	Failure

Description

The GSArrow function draws an arrow located by absolute view coordinates defining its start and end. The arrowhead can be drawn in various styles, as set by the nMode parameter.

J

Topic

GSArrow

Related

GSLineAbs

GSAxis function

Draws X or Y axis

C/C++

```
int GSAxis( double fxOrg, double fyOrg, double fLen,
            double fTickLen, int nMajDivs,
            int nMinDivs, int nMode, int nStyle,
            int nClr )
```

FoxPro

```
r = GSAxis(fxOrg, fyOrg, fLen, fTickLen, nMajDivs,
           nMinDivs, nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSAxis(fxOrg#, fyOrg#, fLen#, fTickLen#,
            nMajDivs%, nMinDivs%, nMode%, nStyle%,
            nClr%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fLen`	Length of axis
`fTickLen`	Length of major ticks
`nMajDivs`	Number of major ticks (divisions along axis)
`nMinDivs`	Number of minor ticks per major tick
`nMode`	

Constant	Value	Meaning
AXTICKOUT	0	Ticks on outside (left on Y axis, bottom on X axis)
AXTICKIN	1	Ticks on inside
AXTICKTHRU	2	Ticks strike through the axis
AXISX	0	Draws in X direction
AXISY	4	Draws in Y direction

`nStyle`	Line style (see Line style constants)
`nClr`	Color of axis (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSAxis function draws an axis with major and minor ticks in the X or Y direction.

Use a negative length to draw an axis in the negative X or Y direction.

J

Topic

GSAxis

Related

GSLogAxis
GSGrid
GSLogGrid
GSPolarAxes
GSCage3D
GSXYGraph

GSBar2D function

Draws 2D bar graph

C/C++

```
int GSBar2D( double fxOrg, double fyOrg, double fInc,
             double fSpace, double fStackHt, int nMode,
             int nGroup )
```

FoxPro

```
r = GSBar2D(fxOrg, fyOrg, fInc, fSpace, fStackHt,
            nMode, nGroup)
```

Visual Basic

```
r% = GSBar2D(fxOrg#, fyOrg#, fInc#, fSpace#, fStackHt#,
             nMode%, nGroup%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fInc` X or Y increment (distance in view units between adjacent data points)

`fSpace` Fractional gap between bars. The width of the space is given by the fSpace parameter, which is expressed as a fraction of the interval between points. For example, an fSpace of 0.1 gives a space of 10% of the interval.

`fStackHt` Height of graph in percentile mode

`nMode`

Constant	Value	Meaning
BARSIMPLE	0	Simple format (nGroup = 1)
BARSTACK	1	Stacked format
BARCLUST	2	Clustered format
BARSTACKPC	3	Stacked percentile format
BARFLOAT	5	Sets first element as transparent
BARHORIZ	8	Horizontal bars
BARVARPOS	16	Takes the X position from the fD array
BARLASTFIRST	64	In horizontal mode, draws bars from top to bottom of Y axis instead of bottom to top

`nGroup` Number of grouped data sets. With clustered data, the interval contains nGroup bars plus a space.

Return values

0	Success
-1	Failure

Description

The GSBar2D function draws a 2D bar graph in one of four formats: simple, stacked, stacked percentile, or clustered. The bars may extend vertically or horizontally.

Stacked percentile bars represent group data as bars of equal height, fStackHt, divided in proportion to the elements making it up. fStackHt is ignored in the other graph formats.

In stacked and clustered format, each data point is represented by a group of data that is transferred in a two-dimensional array, fA.

With simple and stacked bars, the interval between data points consists of a bar and a space.

In simple bar graphs, the number of data sets must be 1.

Optionally, the position of the left-hand corner of the bars relative to the origin may be taken from the fD array rather than at fixed intervals.

The BARFLOAT mode sets the first element of a stacked bar graph to be transparent, giving the appearance of the stacked bars *floating* in space. It only applies to standard stacked bar format, and there must be more than one data set.

GSDataTrans parameters for 2D bar graphs

Simple format

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (always 1)
fA[nPts]	Pointer to amplitude array (Y lengths for vertical bars, X lengths for horizontal bars)
fD[nPts]	Pointer to distance array (X positions for vertical bars, Y positions for horizontal bars)--used only with nMode BARVARPOS
nPatt[nPts]	Pointer to array containing one fill pattern for each bar
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nPts]	Pointer to array containing one color for each bar

Stacked or clustered format

nPts	Number of points per data set (no limit)

nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y lengths for vertical bars, X lengths for horizontal bars)
fD[nPts]	Pointer to distance array (X positions for vertical bars, Y positions for horizontal bars)--used only with nMode BARVARPOS
nPatt[nGroup]	Pointer to array containing one fill pattern for each data set
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nGroup]	Pointer to array containing one color for each data set

J

Topic

GSBar2D

Related

GSArea
GSArea3D
GSTapeGraph

Axis/grid/legend:

GSAxis
GSGrid
GSLegend

Labels:

GSDataLabels
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:

GSDataTrans

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

GSBar3D function

Draws 3D bar graph

C/C++

```
int GSBar3D( double fxOrg, double fyOrg, double fInc,
             double fSpace, double fStackHt,
             double fDepth, double fAng, int nMode,
             int nGroup )
```

FoxPro

```
r = GSBar3D(fxOrg, fyOrg, fInc, fSpace, fStackHt,
            fDepth, fAng, nMode, nGroup)
```

Visual Basic

```
r% = GSBar3D(fxOrg#, fyOrg#, fInc#, fSpace#, fStackHt#,
             fDepth#, fAng#, nMode%, nGroup%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fInc` X or Y increment (distance in view units between adjacent data points)

`fSpace` Fractional gap between bars. The width of the space is given by the fSpace parameter, which is expressed as a fraction of the interval between points. For example, an fSpace of 0.1 gives a space of 10% of the interval.

`fStackHt` Height of graph in percentile mode. Stacked percentile bars represent group data as bars of equal height, fStackHt, divided in proportion to the elements making it up. fStackHt is ignored in other graph formats.

`fDepth` Perspective depth of 3D bar

`fAng` Perspective angle to horizontal

`nMode`

Constant	Value	Meaning
BARSIMPLE	0	Simple format (nGroup = 1)
BARSTACK	1	Stacked format
BARCLUST	2	Clustered format
BARSTACKPC	3	Stacked percentile format
BARCLUSTZ	4	Clustered in the Z axis
BARFLOAT	5	Sets first element as transparent
BARHORIZ	8	Horizontal bars
BARVARPOS	16	Takes the X position from the fD array
BARHALFTONE	32	Fills side faces of bars

		using the half-tones of the nAux colors
BARLASTFIRST	64	In horizontal mode, draws bars from top to bottom of Y axis instead of bottom to top

`nGroup` Number of grouped data sets. With clustered data, the interval contains nGroup bars plus a space.

Return values

0	Success
-1	Failure

Description

The GSBar3D function draws a bar graph in one of four formats: simple (one data set), stacked, stacked percentile, or clustered format. The bars may extend vertically or horizontally. This isn't True3D, but the quasi-3D of previous editions of Graphics Server. True3D graphs can be programmed only through the AutoGraph API.

In stacked and clustered format, each data point is represented by a group of data transferred in the two-dimensional array, fA.

The bars are filled using the patterns and colors defined in the respective arrays. The side faces of the bars can also be filled with a solid in half-tone colors.

Clustered bars can be assembled in the X axis or Z axis, the latter giving a fuller 3D appearance.

With simple and stacked bars, the interval between data points consists of a bar and a space.

Optionally, the position of the bars relative to the origin may be taken from the fD array rather than at fixed intervals.

The BARFLOAT mode sets the first element of a stacked bar graph to be transparent, giving the appearance of the stacked bars *floating* in space. It applies only to standard stacked bar format, and there must be more than one data set.

Setting colors for the tops and sides of 3D bars

By default, the tops and sides of 3D bars are drawn in half-tone colors of the front faces of bars (when the front faces are drawn only in the normal 16-color palette). However, you can use the nAux array--through the GSDataAux function--to set specific colors for the tops and sides of 3D bars.

The size of the nAux array should be either nPts (for simple bar graphs) or nGroup (for clustered or stacked bar graphs).

GSDataTrans parameters for 3D bar graphs

Simple format

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (always 1)
fA[nPts]	Pointer to amplitude array (Y lengths for vertical bars, X lengths for horizontal bars)
fD[nPts]	Pointer to distance array (X positions for vertical bars, Y positions for horizontal bars)--used only with nMode BARVARPOS
nPatt[nPts]	Pointer to array containing one fill pattern for each bar
nSymbol[0]	Not used
nAux[nPts]	Pointer to array containing color for each bar's top and sides
nClr[nPts]	Pointer to array containing one color for each bar's front face

Stacked or clustered format

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y lengths for vertical bars, X lengths for horizontal bars)
fD[nPts]	Pointer to distance array (X positions for vertical bars, Y positions for horizontal bars)--used only with nMode BARVARPOS
nPatt[nGroup]	Pointer to array containing one fill pattern for each data set
nSymbol[0]	Not used
nAux[nGroup]	Pointer to array containing color for top and sides of bars (one color per data set)
nClr[nGroup]	Pointer to array containing one color for front faces of bars (one color per data set)

J

Topic
GSBar3D

Related
GSBar2D
GSArea3D

Axis/cage/legend:
GSCage3D
GSLegend

Labels:
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:
GSDataTrans
GSDataAux

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSBox2D function

Draws box and fills with pattern

C/C++

```
int GSBox2D( double fxOrg, double fyOrg, double fWid,
             double fHt, int nPatt, int nClr )
```

FoxPro

```
r = GSBox2D(fxOrg, fyOrg, fWid, fHt, nPatt, nClr)
```

Visual Basic

```
r% = GSBox2D(fxOrg#, fyOrg#, fWid#, fHt#, nPatt%,
             nClr%)
```

Parameters

`fxOrg` X origin bottom left

`fyOrg` Y origin bottom left

`fWid` Width

`fHt` Height

`nPatt` Pattern (see Pattern constants)

The following additional modes (additive with the usual pattern values) apply for this function:

Constant	Value	Meaning
BXNOBOX	128	No bounding box, just fill
BXSHADOW	256	Draws black shadow to right and down
BXBORDER	512	Draws black border
BXRAISED	1024	Sculptured raised
BXLOWERED	2048	Sculptured lowered

Modes 256, 1024, and 2048 can't be combined. For best effect with the sculptured modes, use a background color of 7.

`nClr` Color of box (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSBox2D function draws a rectangular vertical-sided box and optionally fills it with a pattern. The origin of the box is the bottom left-hand corner.

J

Topic

GSBox2D

Related

GSBox3D
GSPolyFill
GSEllipse
GSLineAbs
GSShade

GSBox3D function

Draws 3D box and fills with pattern

C/C++

```
int GSBox3D( double fxOrg, double fyOrg, double fWid,
             double fHt, double fDepth, double fAng,
             int nPatt, int nClr1, int nClr2 )
```

FoxPro

```
r = GSBox3D(fxOrg, fyOrg, fWid, fHt, fDepth, fAng,
            nPatt, nClr1, nClr2)
```

Visual Basic

```
r% = GSBox3D(fxOrg#, fyOrg#, fWid#, fHt#, fDepth#,
             fAng#, nPatt%, nClr1%, nClr2%)
```

Parameters

`fxOrg`	X origin bottom left
`fyOrg`	Y origin bottom left
`fWid`	Width
`fHt`	Height
`fDepth`	Perspective depth of 3D box
`fAng`	Perspective angle to horizontal
`nPatt`	Pattern (see Pattern constants)

The following additional modes (additive with the usual pattern values) apply for this function:

Constant	Value	Meaning
BXNOBOX	128	No bounding box, just fill
BXBORDER	512	Draws black border

`nClr1`	Front face color (see Color constants)
`nClr2`	Side face color (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSBox3D function draws a rectangular vertical-sided 3D box and optionally fills it with a pattern. The origin of the box is in the bottom left hand corner.

The box drawn isn't True3D, but the quasi-3D of previous editions of Graphics Server. True3D graphs can be programmed only through the AutoGraph API.

J

Topic

GSBox3D

Related

GSBox2D

GSBoxWhisker function

Draws box-whisker graph

C/C++

```
int GSBoxWhisker( double fxOrg, double fyOrg,
                  double fInc, double fSpace,
                  int nMode )
```

FoxPro

```
r = GSBoxWhisker(fxOrg, fyOrg, fInc, fSpace, nMode)
```

Visual Basic

```
r% = GSBoxWhisker(fxOrg#, fyOrg#, fInc#, fSpace#, nMode%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fInc` X increment (distance in view units between adjacent data points)

`fSpace` Space between bars

`nMode`

Constant	Value	Meaning
BWPARAMETRIC	1	Data is parametric (default is raw data)
BWVARX	4	Takes the X position from the fD array
BWNONOTCH	8	Omits the notch
BWNOWHISKER	16	Omits the whisker
BWNOSAMPLES	32	Omits drawing data samples (applies only to raw data, nMode 0)
BWNOMEDIAN	64	Omits median line
BWBLACKBORDER	128	Draws black border around box markers

Return values

0 Success

-1 Failure

Description

The GSBoxWhisker function draws a box-whisker graph.

Optionally, the position of points relative to the origin may be taken from the fD array rather than at fixed intervals.

This graph is used in data analysis to illustrate the spread of values about a median. Visually each point is represented by a box with a waisted notch about the median and vertical lines ("whiskers") extending from the top and bottom. The notches delimit the quartiles of data. The whiskers delimit the 5th and 95th percentiles. The boxes delimit the 10th and 90th percentiles.

The data may be supplied as an array of raw values of size nPts nGroup (with group size greater or equal to 7), which is processed

to produce the percentiles across each group. Or the data may be supplied as a preprocessed *parametric* array with a group size of 7, each group member representing one of the pre-calculated percentiles.

The boxes are patterned and colored from the nPatt and nClr arrays. Curves can be fitted to data if supplied in parametric form. The percentile that the curve is fitted to can be selected using the GSStatsArr function. Curves can't be fitted to raw data.

In mode 0 and 1, both notch and whisker are shown by default. In mode 0 the raw data samples are superimposed on each box-whisker as a column of symbols unless suppressed using the mode switch BWNOSAMPLES. Percentiles are always calculated to the nearest data point rounded up.

When you pass parametric data, you arrange groups as ascending percentiles:

Group	*%*	*Description*
0	5	5th percentile
1	10	10th percentile
2	25	25th percentile
3	50	50th percentile (median)
4	75	75th percentile
5	90	90th percentile
6	95	95th percentile

GSDataTrans parameters for box-whisker graphs

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (always 7 for parametric data and 7 or greater for raw data)
fA[nPts][nGroup]	Pointer to amplitude array (parametric or raw data)
fD[nPts]	Pointer to distance array (X positions for symbols)--used only with nMode BWVARX
nPatt[nPts]	Pointer to array containing one fill pattern for each box-whisker symbol
nSymbol[nPts]	Pointer to array containing symbols for sample points if used
nAux[0]	Not used
nClr[nPts]	Pointer to array containing one color for each box-whisker symbol

Topic
GSBoxWhisker

Related
GSHLC
GSCurveFit
GSStatsArr

Axis/grid/legend:
GSAxis
GSGrid
GSLegend

Labels:
GSDataLabels
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:
GSDataTrans

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSBubbleChart function

Draws bubble graph

C/C++

```
int GSBubbleChart( double fxOrg, double fyOrg, int nMode )
```

FoxPro

```
r = GSBubbleChart(fxOrg, fyOrg, nMode)
```

Visual Basic

```
r% = GSBubbleChart(fxOrg#, fyOrg#, nMode%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`nMode`	Function mode (no modes currently implemented)

Return values

0	Success
-1	Failure

Description

The GSBubbleChart function draws a bubble graph.

Visually the format is similar to a scatter graph, where the location of each bubble is determined by an XY pair. The radius of each bubble enables a third variable to be represented on the same graph.

The bubble graph is unlike other graphs in the way the data arrays are organized. The amplitude array, which contains the bubble radius values, is one-dimensional, while the distance array, which contains the XY coordinate values of the centers of the bubbles, is two-dimensional. This means you can't use the standard GSDataTrans function for passing all the arrays and must use a combination of the individual array passing functions instead. See the example below.

The bubbles may be individually colored and patterned.

GSDataTrans parameters for bubble graphs

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (always 1)
fA[nPts]	Pointer to array containing radius of each bubble
fD[nPts][2]	Pointer to array containing X coordinates (first set) and Y coordinates (second set) of centers of bubbles
nPatt[nPts]	Pointer to array containing one fill pattern for each bubble
nSymbol[0]	Not used
nAux[0]	Not used

nClr[nPts]	Pointer to array containing one color for each bubble

Example

The following example illustrates how you should pass the data arrays before calling the GSBubbleChart function:

```
#define NUMPTS 4
/* one-dimensional array of bubble radii */
double fAmp [NUMPTS] = { 100.0, 150.0, 200.0, 210.0 };
/* two-dimensional array of bubble centers */
double fDist [NUMPTS] [2] = {
   /* X      Y */
     50.0,  50.0,
    100.0, 100.0,
    150.0, 150.0,
    200.0, 150.0
};
int nPatt [NUMPTS] = {
   BRSOLID, BRSOLID, BRSOLID, BRSOLID
};
int nClr [NUMPTS] = { RED, BLUE, GREEN, BROWN };
/* specify the graph dimensions */
GSDataDim( NUMPTS, 1 );
/* send the amplitude array */
/* notice that the number of data sets is one */
GSDataAmp( NUMPTS, 1, fAmp );
/* now send the distance array */
/* notice how the second dimension is specified */
GSDataDist( NUMPTS * 2, &fDist[0][0] );
/* now send the other two arrays as normal */
GSDataPatt( NUMPTS, nPatt );
GSDataClr( NUMPTS, nClr );
/* draw the bubble graph at view location 100-100 */
GSBubbleChart( 100.0, 100.0, 0 );
```

J

Topic
GSBubbleChart

Related
GSPie2D
GSXYGraph

Axis/grid/legend:
GSAxis
GSGrid
GSLegend

Labels:

GSDataLabels
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:
GSDataTrans
GSDataAmp
GSDataDist

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSCage3D function

Draws 3D cage with axes and grids

C/C++

```
int GSCage3D( double fxOrg, double fyOrg, double fxLen,
              double fyLen, double fzLen, double fAng,
              double fThick, int nxGrids, int nyGrids,
              int nzGrids, int nMode, int nClr1,
              int nClr2 )
```

FoxPro

```
r = GSCage3D(fxOrg, fyOrg, fxLen, fyLen, fzLen, fAng,
             fThick, nxGrids, nyGrids, nzGrids, nMode,
             nClr1, nClr2)
```

Visual Basic

```
r% = GSCage3D(fxOrg#, fyOrg#, fxLen#, fyLen#, fzLen#,
              fAng#, fThick#, nxGrids%, nyGrids%,
              nzGrids%, nMode%, nClr1%, nClr2%)
```

Parameters

`fxOrg`	X origin bottom left
`fyOrg`	Y origin bottom left
`fxLen`	Length of X axis
`fyLen`	Length of Y axis
`fzLen`	Length of Z axis
`fAng`	Z axis angle to horizontal
`fThick`	Wall thickness
`nxGrids`	Number of X grids
`nyGrids`	Number of Y grids
`nzGrids`	Number of Z grids
`nMode`	

Constant	Value	Meaning
CGGRIDX	1	Draw X axis grids
CGGRIDY	2	Draw Y axis grids
CGGRIDZ	4	Draw Z axis grids

`nClr1`	Color of exposed cross-section (see Color constants)
`nClr2`	Color of internal faces (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSCage3D function draws the solid projection of a cage with axes and grids to accompany a 3D bar graph projected in the Z direction (BARCLUSTZ). The origin of the cage is at the front left-hand corner.

This isn't True3D but the quasi-3D of Graphics Server version 2.5. True3D graphs can be programmed only through the AutoGraph API.

The cage has height fyLen, width fxLen, and depth fzLen. The depth is the length of the projected Z axis, which is displayed obliquely on the screen.

Optionally, you can draw grids intersecting the three axes. The nxGrids parameter refers to grids intersecting the X axis in the X-Z plane, nyGrids to those intersecting the Y axis in the Y-Z plane, and nzGrids to those intersecting the Z axis in the X-Z plane.

Note that the grid count is essentially the number of divisions on the axis, including the grid at the furthest extreme of the axis. A grid count of 2 results in a single grid line at the center of the axis.

J

Topic

GSCage3D

Related

GSAxis
GSBar3D
GSArea3D
GSTapeGraph

GSCircle function

Draws circle

C/C++

```
int GSCircle( double fxOrg, double fyOrg,
              double fRadius, int nMode, int nStyle,
              int nClr )
```

FoxPro

```
r = GSCircle(fxOrg, fyOrg, fRadius, nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSCircle(fxOrg#, fyOrg#, fRadius#, nMode%, nStyle%,
              nClr%)
```

Parameters

`fxOrg` X center

`fyOrg` Y center

`fRadius` Radius

`nMode`

Constant	Value	Meaning
CCFILL	2	Fills the circle with pattern
CCTHICK	4	Uses thick line style

These modes are exclusive. If you enable both of them, only CCFILL is used.

`nStyle` Line thickness (nMode CCTHICK) or fill pattern (nMode CCFILL). See Line style constants or Pattern constants.

`nClr` Color of circle (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSCircle functions draws a circle.

Optionally, the circle may be filled with a pattern. In this case, nStyle defines the pattern.

J

Topic

GSCircle

Related

GSEllipse

GSArc

GSPolyFill

GSClearView function

Clears view

C/C++

```
int GSClearView( int nMode )
```

FoxPro

```
r = GSClearView(nMode)
```

Visual Basic

```
r% = GSClearView(nMode%)
```

Parameter `nMode`

Constant	Value	Meaning
CLTRANSP	0	Clears the contents of a view and gives it a transparent background so that the contents of other views continue to show through.
CLOPAQUE	1	Clears the contents of a view and gives it an opaque background in the current background color, which obliterates the contents of other views. The background color is controlled by calling GSSetBG before GSClearView.

Return values

0	Success
-1	Failure

Description The GSClearView function clears a view.

J

Topic
GSClearView

Related
GSSetBG
GSGetBg
GSShade
GSWinPaint

View functions:
GSCloseView
GSGetVXExt
GSGetVYExt
GSOffView
GSOnView

GSOpenView
GSUseView

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSClipRead function

Reads image from Clipboard

C/C++

```
int GSClipRead( double fxBL, double fyBL, double fWid,
                double fHt, int nFormat, int nMode )
```

FoxPro

```
r = GSClipRead(fxBL, fyBL, fWid, fHt, nFormat, nMode)
```

Visual Basic

```
r% = GSClipRead(fxBL#, fyBL#, fWid#, fHt#, nFormat%, nMode%)
```

Parameters

`fxBL` X bottom left

`fyBL` Y bottom left

`fWid` Image width

`fHt` Image height

`nFormat`

Constant	Value	Meaning
CBBMP	1	Device-dependent bitmap format
CBWMF	2	Windows metafile format
CBDIB	4	Windows device-independent bitmap format

`nMode`

Constant	Value	Meaning
	0	The image is located at the bottom left corner of the area and retains its original dimensions, with free space or clipping possible at both the top and right-hand edges.
CBCENTER	1	The image retains its original dimensions and the center of the image is located at the center of the area, with free space or clipping possible at both the horizontal and vertical edges.
CBSTRETCH	2	The image is located at the bottom left corner of the area and is stretched or compressed in either direction to give an exact fit in the area.
CBTILE	3	The image retains its original dimensions and is tiled repetitively from left to right and bottom to top of

the area.

Return values

0	Success
-1	Failure

Description

The GSClipRead function reads an image from the Windows Clipboard into the current view.

The rectangular area of the image in the view is defined by the bottom left corner and a width and height, all expressed in the current view units. If zero width and height are specified, the area is presumed to extend from the point of origin to the current width and height extents of the view.

GSClipRead handles Clipboard images in several formats. An application writing to the Clipboard can make an image available in different formats and let the receiving application decide which of them to use. The nFormat parameter specifies the format in which your application reads the image.

Format options may be combined--for example, CBBMP | CBDIB. When you do this, Graphics Server uses the first format specified in nFormat that matches a format in the Clipboard, taken in the order of preference of formats specified by the application that stored the image. Only one form of the image is actually imported into the current view. Once imported, the image from the Clipboard becomes a permanent part of the view.

The image in the Clipboard may be larger or smaller than the view area defined by the fxBL, fyBL, fWid, and fHt parameters. The nMode parameter specifies how the imported image is to fit the available area.

Example

The following example copies a device-independent bitmap image from the Clipboard into the current view. The view is divided into quadrants and the image is copied into each of the quadrants using the different modes available.

```
void OnEditPaste()
{
double fxMid, fyMid;
fxMid = GSGetVXExt() / 2; fyMid = GSGetVYExt() / 2;
GSClipRead(0, 0, fxMid, fyMid, CBDIB, 0);
GSClipRead(fxMid, 0, fxMid, fyMid, CBDIB, CBCENTER);
GSClipRead(0, fyMid, fxMid, fyMid, CBDIB, CBSTRETCH);
GSClipRead(fxMid, fyMid, fxMid, fyMid, CBDIB, CBTILE);
}
```

Topic

GSClipRead

Related

GSClipWrite
GSGetMF
GSPicRead
GSPicWrite

GSClipWrite function

Writes image to Clipboard

C/C++

```
int GSClipWrite( double fxBL, double fyBL, double fWid,
                 double fHt, int nFormat, int nMode )
```

FoxPro

```
r = GSClipWrite(fxBL, fyBL, fWid, fHt, nFormat, nMode)
```

Visual Basic

```
r% = GSClipWrite(fxBL#, fyBL#, fxWid#, fHt#, nFormat%,nMode%)
```

Parameters

`fxBL`	X bottom left
`fyBL`	Y bottom left
`fWid`	Image width
`fHt`	Image height

`nFormat`

Constant	Value	Meaning
CBBMP	1	Device-dependent bitmap
CBWMF	2	Windows metafile
CBDIB	4	Windows device-independent bitmap

`nMode`

Constant	Value	Meaning
CBMONO	256	Exports the image in monochrome mode

Return values

0	Success
-1	Failure

Description

The GSClipWrite function writes an image of the current graphing window to the Windows Clipboard.

The rectangular area of the window is defined by the bottom left corner and a width and height, all expressed in the view units of view 0, the default view. If zero width and height are specified, the area is presumed to extend from the point of origin to the current extents of width and height of the view. The option to specify an area of the window isn't supported in this release. An image of the whole window is always exported.

The image may be written in a variety of different formats. An application writing to the Clipboard can make an image available in different formats and let the receiving application decide which of them it wishes to use. The nFormat parameter enables your application to specify a set of formats from which a receiving application can choose to import the image.

Format options may be combined--for example, CBBMP | CBDIB. In this case, Graphics Server writes the image to the Clipboard in each of the formats specified. The order in which the format is

written is always as shown in the nFormat table.

You can choose to export the image in monochrome mode by means of the nMode parameter.

Example

The following example copies an image of the current window to the Clipboard in device-dependent, device-independent, and Windows metafile formats:

```
void OnEditCopy()
{
GSClipWrite( 0, 0, 0, 0, CBBMP | CBDIB | CBWMF, 0 );
}
```

J

Topic

GSClipWrite

Related

GSClipRead
GSGetMF
GSPicRead
GSPicWrite

GSCloseServer function

Closes connection to Graphics Server

C/C++

```
int GSCloseServer( )
```

FoxPro

```
r = GSCloseServer()
```

Visual Basic

```
r% = GSCloseServer()
```

Return values

0	Success
-1	Failure

Description

The GSCloseServer function closes the connection between your application and Graphics Server.

All open graphing windows and views belonging to the application are closed.

J

Topic

GSCloseServer

Related

GSOpenServer
GSOpenWin
GSOpenChildWin
GSCloseWin
GSOpenView

GSClosePrn function

Closes printer

C/C++

```
int GSClosePrn( )
```

FoxPro

```
r = GSClosePrn()
```

Visual Basic

```
r% = GSClosePrn()
```

Return values

0	Success
-1	Failure

Description

The GSClosePrn function closes the printer previously opened with GSOpenPrn.

J

Topic

GSClosePrn

Related

GSOpenPrn

GSPrnSetup

GSPrnOut

GSCloseView function

Closes view

C/C++

```
int GSCloseView( nWin, nView, nMode )
```

FoxPro

```
r = GSCloseView(nWin, nView, nMode)
```

Visual Basic

```
r% = GSCloseView(nWin%, nView%, nMode%)
```

Parameters

`nWin`	Window number		
`nView`	View number		
`nMode`	**Constant**	**Value**	**Meaning**
	CVKEEP	0	Retains view contents by copying to view 0
	CVDISCARD	1	Discards view contents

Return values

0	Success
-1	Failure

Description

The GSCloseView function closes a view.

The contents of the view may be kept or discarded. If kept, the contents are copied to view 0 and become a permanent part of that view until cleared.

If you close the active view, view 0 becomes the new active view. View 0 can't be closed.

J

Topic

GSCloseView

Related

GSClearView
GSCloseView
GSGetVXExt
GSGetVYExt
GSOffView
GSOnView
GSOpenView
GSUseView

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

GSCloseWin function

Closes graphing window

C/C++

```
int GSCloseWin( nWin )
```

FoxPro

```
r = GSCloseWin(nWin)
```

Visual Basic

```
r% = GSCloseWin(nWin%)
```

Parameter

`nWin`	Window number

Return values

0	Success
-1	Failure

Description

The GSCloseWin function closes the window identified by nWin. The window number will have been returned by a previous call to GSOpenWin or GSOpenChildWin.

J

Topic

GSCloseWin

Related

GSOpenWin
GSOpenChildWin
GSOpenView

Server:

GSCloseServer
GSOpenServer

GSCurveFit function

Fits curve to data

C/C++

```
int GSCurveFit( int nType, int nOrder, int nSteps,
                int nMode, int nStyle, int nClr )
```

FoxPro

```
r = GSCurveFit(nType, nOrder, nSteps, nMode, nStyle,
               nClr)
```

Visual Basic

```
r% = GSCurveFit(nType%, nOrder%, nSteps%, nMode%,
                nStyle%, nClr%)
```

Parameters

`nType`

Constant	Value	Meaning
CFPOLY	0	Variable-order polynomial
CFLOG	1	Logarithmic y = a + b * ln(x)
CFEXP1	2	Exponential y = a * exp(b * x)
CFEXP2	3	Exponential y = a * x * exp(-b * x)
CFPOWER	4	Power y = a * (x ^ b)
CFINV1	5	Inverse y = a + b / x
CFINV2	6	Inverse y = a / (b + x)
CFINV3	7	Inverse y = 1 / (a + b * x)
CFINV4	8	Inverse y = x / (a * x + b)
CFINV5	9	Inverse y = 1 / (a + b * x) ^ 2
CFSPLINE	10	Spline fit through all points
CFMOVINGAVEMID	11	Moving average plotted at midpoint of averaged group
CFMOVINGAVEEND	12	Moving average plotted at end point of averaged group

`nOrder` Curve order. nOrder is relevant only to the

variable-order polynomial fit and moving averages. For moving averages, nOrder defines the number of points over which the average is taken.

`nSteps` Number of steps. nSteps defines the granularity of the drawn curve; higher nSteps values lead to smoother curves.

J *For most curves (nTypes 1-9),* an nSteps setting of 50 generally produces a smooth curve at a high drawing speed.

J *For spline curves (nType CFSPLINE),* you generally need a much larger nSteps value--typically 10 times the number of points in the graph, or higher for very irregular graphs.

J *For moving averages (nType CFMOVINGAVEMID or CFMOVINGAVEEND),* nSteps isn't relevant--the lines drawn between the plotted averages are always straight.

`nMode`

Constant	Value	Meaning
	0	Draws patterned lines as specified by nStyle
LATHICK	4	Draws thick lines as specified by nStyle

`nStyle` Line pattern or thickness (see Line style constants)

`nClr` Color of curve (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSCurveFit function fits and draws a curve of the specified type through the most recent graph data. The curve is clipped within a window defined by the GSStatsWin function.

J

Topic

GSCurveFit

Related

GSBoxWhisker
GSGetCC
GSGetCurveCoeff
GSLineFit

GSStatsArr

GSDataAmp function

Transfers array of amplitude data

C/C++

```
int GSDataAmp( int nPts, int nGroup, double* fAmp )
```

FoxPro

```
r = GSDataAmp(nPts, nGroup, @fAmp(1))
```

Visual Basic

```
r% = GSDataAmp(nPts%, nGroup%, fAmp#(0))
```

Parameters

`nPts`	Number of points per data set
`nGroup`	Number of data sets
`fAmp`	Pointer to amplitude data array

Return values

0	Success
-1	Failure

Description

The GSDataAmp function transfers amplitude data to Graphics Server for use in subsequent graphing or drawing functions.

Amplitude data is used in all the graph types of Graphics Server.

The product of nPts and nGroup specifies the number of elements in the amplitude array you're transferring. The graphing functions use the number of points per data set and number of data sets--taken from the most recent call to GSDataDim or GSDataTrans--to determine the logical dimensions of the graph. You must transfer an array whose number of elements is enough to meet the logical requirements of the graph you're going to draw.

This function stores the data in the same array as GSDataTrans and a call to either one will overwrite the data stored by the other.

J

Topic

GSDataAmp

Related

GSDataAmpErr
GSDataGetAmp
GSDataStoreAmp
GSDataTrans

Array modification:

GSDataDim
GSDataRange

GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataAmpErr function

Transfers array of amplitude error data

C/C++

```
int GSDataAmpErr( int nPts, int nGroup,
                  double* fAmpErr )
```

FoxPro

```
r = GSDataAmpErr(nPts, nGroup, @fAmpErr(1))
```

Visual Basic

```
r% = GSDataAmpErr(nPts%, nGroup%, fAmpErr#(0))
```

Parameters

`nPts`	Number of error values per data set
`nGroup`	Number of data sets
`fAmpErr`	Pointer to amplitude error array

Return values

0	Success
-1	Failure

Description

The GSDataAmpErr function transfers amplitude error data to Graphics Server for use in subsequent graphing functions that add user-defined error bars.

Errors are passed as plus and minus error pairs stored consecutively. Both values must be passed even though the display of one or other may later be suppressed.

Errors can be passed in two modes. First, where an error pair is supplied for every point in the graph. In this case, nPts is twice the value used in GSDataAmp (to account for the pair of values) and nGroup is the same. Second, where an error pair is passed for each group, the same error pair being applied to each data point in the group. In this case, nPts is always 2, and nGroup is the same as in GSDataAmp. These modes are detected automatically by Graphics Server from the value of nPts.

Error values must be positive.

J

Topic

GSDataAmpErr

Related

GSDataAmp
GSDataGetAmpErr
GSDataStoreAmpErr
GSErrorBar

Array modification:

GSDataDim

GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataAux function

Transfers array of auxiliary data

C/C++

```
int GSDataAux( int nElements, int* nAux )
```

FoxPro

```
r = GSDataAux(nElements, @nAux(1))
```

Visual Basic

```
r% = GSDataAux(nElements%, nAux%(0))
```

Parameters

`nElements` Number of elements in auxiliary array

- For one-dimensional auxiliary arrays (one data set), nElements equals nPts
- For two-dimensional auxiliary arrays (more than one data set), nElements equals nPts % nGroup

`nAux` Pointer to auxiliary array

Return values

0	Success
-1	Failure

Description

The GSDataAux function transfers auxiliary data to Graphics Server for use in subsequent graphing functions.

Auxiliary data is used for four purposes: to "explode" pie chart slices, to set the colors of the sides and tops of bars in 3D bar graphs, to specify statistical lines for time series graphs, and to specify "missing" data points in several graph types (line and logarithmic, polar, 2D scatter, and tape).

The graphing functions use either the number of points or number of data sets--taken from the most recent call to the GSDataDim or GSDataTrans function--to determine how many auxiliary array elements are required. (See the entry for the specific graphing function to find out if you need to provide auxiliary values on a per-point or per-set basis.) You need enough elements in the auxiliary array to cover every point or set.

GSDataAux stores data in the same array as GSDataTrans, and a call to either function overwrites data stored by the other.

Specifying points as "missing" in line, logarithmic, polar, 2D scatter, and tape graphs

If you have incomplete sets of data or sets in which the values of certain points are unknown, you can use the nAux array to flag such points as missing. In this case, the marker for that point--such as a symbol--isn't drawn. If the graph uses lines (or tapes) to connect points, the connecting lines or tapes are omitted both

to and from each missing point.

nAux setting	**Value**	**Meaning**
	0	Point shown normally
	256	Point is "missing" and not drawn

The size of the nAux array may be nPts or nPts nGroup. If you set the size to nPts and there's more than one group of data, the same missing points are assumed for all the groups. If the size is nPts nGroup, each point in each group has its own missing-data flag.

J

Topic

GSDataAux

Related

GSDataGetAux
GSDataStoreAux
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataClr function

Transfers array of color data

C/C++

```
int GSDataClr( int nPts, int* nClr )
```

FoxPro

```
r = GSDataClr(nPts, @nClr(1))
```

Visual Basic

```
r% = GSDataClr(nPts%, nClr%(0))
```

Parameters

`nPts`	Number of elements in color array
`nClr`	Pointer to color array (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSDataClr function transfers color data to Graphics Server for use in subsequent graphing functions.

Color data is used in some of the graph types but not in others. See the specification of the data arrays of the graphing function to find out if color data is required. If the specification shows a color array dimension of zero (nClr[0]), there's no need to call the GSDataClr function.

The graphing functions use the number of points per data set or, alternatively, the number of data sets--taken from the most recent call to GSDataDim or GSDataTrans--to determine how many color array elements are required. See the specification of the data arrays of the graphing function to find out if you need to provide color values on a per-point or per-set basis. You must transfer an array whose number of elements is enough to meet the logical requirements of the graph you're going to draw.

GSDataClr stores data in the same array as GSDataTrans, and a call to either function overwrites data stored by the other.

J

Topic

GSDataClr

Related

GSDataGetClr
GSDataStoreClr
GSDataTrans

Array modification:

GSDataDim
GSDataRange

GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataDim function

Sets graph data dimensions

C/C++

```
int GSDataDim( int nPts, int nGroup )
```

FoxPro

```
r = GSDataDim(nPts, nGroup)
```

Visual Basic

```
r% = GSDataDim(nPts%, nGroup%)
```

Parameters

`nPts`	Number of points per data set in graph
`nGroup`	Number of data sets in graph

Return values

0	Success
-1	Failure

Description

The GSDataDim function deallocates all the current graph data arrays and sets the logical dimensions of the next graph to be drawn by one of the graphing functions.

Once you've called GSDataDim, you can transfer your new data arrays by calling one or more of the array-passing functions--GSDataAmp, GSDataAux, and so on.

Rather than using GSDataDim and individual array functions, you can use the GSDataTrans function to transfer all your data arrays in one step. GSDataTrans contains an implicit call to GSDataDim, so you don't have to make a separate call.

J

Topic
GSDataDim

Related
GSDataTrans
GSDataReset

Array initialization:
GSDataAmp
GSDataAux
GSDataClr
GSDataDist
GSDataPatt
GSDataSym
GSDataZ

GSDataDist function

Transfers array of distance data

C/C++

```
int GSDataDist( int nElements, double* fDist )
```

FoxPro

```
r = GSDataDist(nElements, @fDist(1))
```

Visual Basic

```
r% = GSDataDist(nElements%, fDist#(0))
```

Parameters

`nElements` Number of elements in distance array

J For one-dimensional distance arrays (one data set), nElements equals nPts

J For two-dimensional distance arrays (more than one data set), nElements equals nPts % nGroup

`fDist` Pointer to distance array

Return values

0	Success
-1	Failure

Description

The GSDataDist function transfers distance data to Graphics Server for use in subsequent graphing functions.

Distance data is accepted and used by most of the graph types. See the specification of the data arrays of the graphing function to find out if distance data is accepted. If the specification shows a distance array
dimension of zero (fDist[0]), there's no need to call the GSDataDist function.

The graphing functions normally use the number of points per data set--taken from the most recent call to GSDataDim or GSDataTrans--to determine how many distance array elements are required. You must transfer an array whose number of elements is enough to meet the logical requirements of the graph you're going to draw.

GSDataDist stores data in the same array as GSDataTrans, and a call to either function overwrites data stored by the other.

J

Topic

GSDataDist

Related

GSDataGetDist

GSDataGetDistErr

GSDataStoreDist
GSDataTrans

Array modification:
GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataDistErr function

Transfers array of distance error data

C/C++

```
int GSDataDistErr( int nPts, double* fDistErr )
```

FoxPro

```
r = GSDataDistErr(nPts, @fDistErr(1))
```

Visual Basic

```
r% = GSDataDistErr(nPts%, fDistErr#(0))
```

Parameters

`nPts`	Number of elements in distance error array
`fDistErr`	Pointer to distance error array

Return values

0	Success
-1	Failure

Description

The GSDataDistErr function transfers distance error data to Graphics Server for use in subsequent graphing functions that add user-defined error bars.

Errors are passed as plus and minus error pairs stored consecutively. Both values must be passed even though the display of one or other may later be suppressed.

Errors can be passed in two modes. First, where an error pair is supplied for every point in the graph. In this case, nPts is (2 nDistPts) where nDistPts is the value used in GSDataDist. Second, where an error pair is passed for each group, the same error pair being applied to each data point in the group. In this case, nPts is (2 nAmpGroup) where nAmpGroup is the value used in GSDataAmp. These modes are detected automatically by Graphics Server from the value of nPts.

Error values must be positive.

J

Topic

GSDataDistErr

Related

GSDataGetDistErr
GSDataStoreDistErr
GSErrorBar

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale

GSTimeUpdate
GSXDataScale

GSDataGetAmp function

Gets amplitude data value

C/C++

```
int GSDataGetAmp( int nPt, int nGroup )
```

FoxPro

```
r = GSDataGetAmp(nPt, nGroup)
```

Visual Basic

```
r# = GSDataGetAmp(nPt%, nGroup%)
```

Parameters

`nPt`	Point index
`nGroup`	Set index

Return values

0	Success
-1	Failure

Description

The GSDataGetAmp function gets the value of a single entry in the Graphics Server amplitude data array.

The dimensions of the array are set by the most recent call to the GSDataAmp or GSDataTrans function.

nPt and nGroup are the indexes of the entry in the two-dimensional array. Array indexes start at zero. A failure value is returned if either index is out of range.

J

Topic

GSDataGetAmp

Related

GSDataAmp
GSDataAmpErr
GSDataStoreAmp
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataGetAmpErr function

Gets amplitude error data value

C/C++

```
int GSDataGetAmpErr( int nPt, int nGroup, int nMode )
```

FoxPro

```
r = GSDataGetAmpErr(nPt, nGroup, nMode)
```

Visual Basic

```
r# = GSDataGetAmpErr(nPt%, nGroup%, nMode%)
```

Parameters

`nPt`	Point index		
`nGroup`	Set index		
`nMode`	**Constant**	**Value**	**Meaning**
	ERRPLUS	0	Gets plus error
	ERRMINUS	1	Gets minus error

Return values

0	Success
-1	Failure

Description

The GSDataGetAmpErr function gets the value of a single entry in the Graphics Server amplitude error data array.

The dimensions of the array are set by the most recent call to GSDataAmpErr.

nPt and nGroup are the indexes of the entry in the two-dimensional array. Array indexes start at zero. A failure value is returned if either index is out of range.

J

Topic

GSDataGetAmpErr

Related

GSDataAmpErr
GSDataStoreAmpErr
GSDataTrans
GSErrorBar

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataGetAux function

Gets auxiliary data value

C/C++

```
int GSDataGetAux( int nPt )
```

FoxPro

```
r = GSDataGetAux(nPt)
```

Visual Basic

```
r% = GSDataGetAux(nPt%)
```

Parameter

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.

Return values

0	Success
-1	Failure

Description

The GSDataGetAux function gets the value of a single entry in the Graphics Server auxiliary data array.

The dimension of the array is set by the most recent call to GSDataAux or GSDataTrans.

J

Topic

GSDataGetAux

Related

GSDataAux
GSDataStoreAux
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataGetClr function

Gets color data value

C/C++

```
int GSDataGetClr( int nPt )
```

FoxPro

```
r = GSDataGetClr(nPt)
```

Visual Basic

```
r% = GSDataGetClr(nPt%)
```

Parameter

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.

Return values

0	Success
-1	Failure

Description

The GSDataGetClr function gets the value of a single entry in the Graphics Server color data array.

The dimension of the array is set by the most recent call to GSDataClr or GSDataTrans.

J

Topic

GSDataGetClr

Related

GSDataClr
GSDataStoreClr
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataGetDist function

Gets distance data value

C/C++

```
int GSDataGetDist( int nPt )
```

FoxPro

```
r = GSDataGetDist(nPt)
```

Visual Basic

```
r# = GSDataGetDist(nPt%)
```

Parameter

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.

Return values

0	Success
-1	Failure

Description

The GSDataGetDist function gets the value of a single entry in the Graphics Server distance data array.

The dimension of the array is set by the most recent call to GSDataDist or GSDataTrans.

J

Topic

GSDataGetDist

Related

GSDataDist
GSDataStoreDist
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataGetDistErr function

Gets distance error data value

C/C++

```
double GSDataGetDistErr( int nPt, int nMode )
```

FoxPro

```
r = GSGetDataGetDistErr(nPt, nMode)
```

Visual Basic

```
r# = GSDataGetDistErr(nPt%, nMode%)
```

Parameters

`nPt` Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.

`nMode`

Constant	Value	Meaning
ERRPLUS	0	Gets plus error
ERRMINUS	1	Gets minus error

Return values

0	Success
-1	Failure

Description

The GSDataGetDistErr function gets the value of a single entry in the Graphics Server distance error data array. The dimensions of the array are set by the most recent call to GSDataDistErr.

J

Topic

GSDataGetDistErr

Related

GSDataDistErr
GSDataStoreDistErr
GSErrorBar

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataGetPatt function

Gets pattern data value

C/C++

```
int GSDataGetPatt( int nPt )
```

FoxPro

```
r = GSDataGetPatt(nPt)
```

Visual Basic

```
r% = GSDataGetPatt(nPt%)
```

Parameter

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.

Return values

0	Success
-1	Failure

Description

The GSDataGetPatt function gets the value of a single entry in the Graphics Server pattern data array.

The dimension of the array is set by the most recent call to GSDataPatt or GSDataTrans.

J

Topic

GSDataGetPatt

Related

GSDataPatt
GSDataStorePatt
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataGetSym function

Gets symbol data value

C/C++

```
int GSDataGetSym( int nPt )
```

FoxPro

```
r = GSDataGetSym(nPt)
```

Visual Basic

```
r% = GSDataGetSym(nPt%)
```

Parameter

`nPt` Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.

Return values

Symbol value (or -1 if failure)

Description

The GSDataGetSym function gets the value of a single entry in the Graphics Server symbol data array.

The dimension of the array is set by the most recent call to GSDataSym or GSDataTrans.

J

Topic

GSDataGetSym

Related

GSDataSym
GSDataStoreSym
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataGetZ function

Gets Z data value

C/C++

```
int GSDataGetZ( int nPt )
```

FoxPro

```
r = GSDataGetZ(nPt)
```

Visual Basic

```
r# = GSDataGetZ(nPt%)
```

Parameter

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.

Return values

Z data value (or -1 if failure)

Description

The GSDataGetZ function gets the value of a single entry in the Graphics Server Z data array.

The dimensions of the array are set by the most recent call to GSDataZ.

J

Topic

GSDataGetZ

Related

GSDataZ
GSDataStoreZ
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataLabels function

Enables and sets text for data labels

C/C++

```
int GSDataLabels( int nMode, int nPrec, int nCSet,
                  int nTMode, int nClr,
                  double fDataOffset, int nLabs,
                  char* szLabels[] )
```

FoxPro

```
r = GSDataLabels(nMode, nPrec, nCSet, nTMode, nClr,
                 fDataOffset, nLabs, @szLabels(1) )
```

Visual Basic

Use VBGSDataLabels function

Parameters

`nMode`

Constant	Value	Meaning
DLTEXT	0	Labels supplied in array szLabels
DLDATA	1	Labels derived from data
DLGROUPCLR	4	Colored as data group

DLGROUPCLR overrides nClr to set the color of labels to the color of the associated data group, except in graph types where the label would overprint block color and be invisible (such as bubble graphs and Gantt charts). In those cases, the DLGROUPCLR flag is ignored and the label is always rendered in nClr.

`nPrec`

Value	Meaning
0 or greater	Specific decimal precision (use 0 if you supply text labels)
-1	Precision is automatically calculated:

- If all values in array are integers from 0 to 999,999, numbers are represented in full
- *If all values in array are fractional,* each number is represented with three-digit precision
- *For arrays containing mixed values,* all numbers are scaled to the closest power of 1000 and represented with three-digit precision (for example, 3,456,000 is shown as 3.45)

`nCSet` Character set (see Character set constants)

`nTMode` Text mode (see Text mode constants)

`nClr` Color of data labels (see Color constants)

`fDataOffset`	Number to be subtracted from the data values to compensate for a nonzero origin (numeric labels only; ignored for text labels)
`nLabs`	

Value	Meaning
0	Use if deriving labels from data (nMode DLDATA)
1 or greater	Use for number of labels if supplying text labels (nMode DLTEXT) The label array must be of size nPts nGroup to provide text labels for each data item on display. The exceptions are high-low-close, open-high-low-close, candlestick, and box-whisker graphs, which require a text array of size nPts (only one label is provided for each compound symbol, of which there are nPts).

`szLabels`	Array of text labels of length nLabs (or null for derived labels)

Return values

0	Success
-1	Failure

Description

The GSDataLabels function enables data labels, which are labels--either numeric or text--attached to each point of a graph. Data labels are available for all 2D graph types except pie charts (which have their own labeling scheme) and time series graphs. They aren't available for 3D graphs.

In high-low-close, open-high-low-close, box-whisker, and candlestick graphs, if you choose to have data labels derived from data (nMode DLDATA), they're derived from the close or median.

You have to call GSDataLabels before you call the graphing function, such as GSBar2D, because labels are drawn at the same time as the graph itself.

J

Topic

GSDataLabels

Related

VBGSDataLabels

GSDataPatt function

Transfers array of pattern data

C/C++

```
int GSDataPatt( int nPts, int* nPatt )
```

FoxPro

```
r = GSDataPatt(nPts, @nPatt(1))
```

Visual Basic

```
r% = GSDataPatt(nPts%, nPatt%(0))
```

Parameters

`nPts`	Number of pattern values
`nPatt`	Pointer to pattern array (see Pattern constants)

Return values

0	Success
-1	Failure

Description

The GSDataPatt function transfers pattern data to Graphics Server for use in subsequent graphing functions.

Pattern data is used in some of the graph types but not in others. See the specification of the data arrays of the graphing function to find out if pattern data is required. If the specification shows a pattern array dimension of zero (nPatt[0]), there's no need to call the GSDataPatt function.

The graphing functions use the number of points per data set or, alternatively, the number of data sets--taken from the most recent call to GSDataDim or GSDataTrans--to determine how many pattern array elements are required. See the specification of the data arrays of the graphing function to find out if you need to provide pattern values on a per-point or per-set basis. You must transfer an array whose number of elements is enough to meet the logical requirements of the graph you're going to draw.

GSDataPatt stores data in the same array as GSDataTrans, and a call to either function overwrites data stored by the other.

J

Topic

GSDataPatt

Related

GSDataGetPatt
GSDataStorePatt
GSDataTrans

Array modification:

GSDataDim
GSDataRange

GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataRange function

Defines range of data to graph

C/C++

```
int GSDataRange( int nFirst, int nLast )
```

FoxPro

```
r = GSDataRange(nFirst, nLast)
```

Visual Basic

```
r% = GSDataRange(nFirst%, nLast%)
```

Parameters

`nFirst`	First point
`nLast`	Last point

Return values

0	Success
-1	Failure

Description

The GSDataRange function defines a subset of the current data set for use in subsequent graphing operations. Any graphing functions called after GSDataRange show only the range of data lying between the first (nFirst) and last (nLast) points you specify.

The range defaults to the complete data set following a new call to the GSDataTrans function.

Array indexes start at zero.

J

Topic

GSDataRange

Related

GSDataDim
GSDataScale
GSGetCC
GSGetMax
GSGetMean
GSGetSD

Array initialization:

GSDataAmp
GSDataAux
GSDataClr
GSDataDist
GSDataPatt
GSDataSym
GSDataTrans

GSDataZ

GSDataReset function

Resets data arrays

C/C++

```
int GSDataReset( )
```

FoxPro

```
r = GSDataReset()
```

Visual Basic

```
r% = GSDataReset()
```

Return values

0	Success
-1	Failure

Description

The GSDataReset function deallocates the Graphics Server internal data arrays, setting all data counts and indexes to zero and restoring defaults.

It can be used to reduce the memory overhead in the server when the contents of the data arrays are no longer required.

GSDataTrans automatically discards any existing data arrays before passing the new data to Graphics Server.

J

Topic

GSDataReset

Related

GSDataTrans
GSDataDim

Array initialization:

GSDataAmp
GSDataAux
GSDataClr
GSDataDist
GSDataPatt
GSDataSym
GSDataZ

GSDataScale function

Applies scale factor to data

C/C++

```
int GSDataScale( double fScale )
```

FoxPro

```
r = GSDataScale(fScale)
```

Visual Basic

```
r% = GSDataScale(fScale#)
```

Parameter

`fScale`	Data scale factor

Return values

0	Success
-1	Failure

Description

The GSDataScale function scales data as portrayed in any of the graph or chart functions.

Amplitude data in the amplitude array is multiplied by this factor before graphing.

The default factor of unity is reset whenever new data is transferred.

J

Topic

GSDataScale

Related

GSDataDim
GSDataRange
GSGetMax
GSGetMean
GSGetSD
GSXDataScale

Array initialization:

GSDataAmp
GSDataAux
GSDataClr
GSDataDist
GSDataPatt
GSDataSym
GSDataTrans
GSDataZ

GSDataStoreAmp function

Stores amplitude data value

C/C++

```
int GSDataStoreAmp( int nPt, int nGroup, double fAmp )
```

FoxPro

```
r = GSDataStoreAmp(nPt, nGroup, fAmp)
```

Visual Basic

```
r% = GSDataStoreAmp(nPt%, nGroup%, fAmp#)
```

Parameters

`nPt`	Point index
`nGroup`	Set index
`fAmp`	Amplitude value

Return values

0	Success
-1	Failure

Description

The GSDataStoreAmp function sets the value of a single entry in the Graphics Server amplitude data array. The dimensions of the array are set by the most recent call to GSDataAmp or GSDataTrans.

nPt and nGroup are the indexes of the entry in the two-dimensional array. Array indexes start at zero.

Topic

GSDataStoreAmp

Related

GSDataAmp
GSDataAmpErr
GSDataAux
GSDataClr
GSDataDist
GSDataGetAmp
GSDataPatt
GSDataStoreAmpErr
GSDataTrans
GSDataZ

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale

GSTimeUpdate
GSXDataScale

GSDataStoreAmpErr function

Stores amplitude error value pair

C/C++

```
int GSDataStoreAmpErr( int nPt, int nGroup,
                       double fErrPlus,
                       double fErrMinus )
```

FoxPro

```
r = GSDataStoreAmpErr(nPt, nGroup, fErrPlus, fErrMinus)
```

Visual Basic

```
r% = GSDataStoreAmpErr(nPt%, nGroup%, fErrPlus#,
                       fErrMinus#)
```

Parameters

`nPt`	Point index
`nGroup`	Set index
`fErrPlus`	Plus error
`fErrMinus`	Minus error

Return values

0	Success
-1	Failure

Description

The GSDataStoreAmpErr function sets the value of a single error value pair in the Graphics Server amplitude error data array.

The dimensions of the array are set by the most recent call to GSDataAmpErr.

nPt and nGroup are the indexes of the entry in the two-dimensional array. Array indexes start at zero.

J

Topic

GSDataStoreAmpErr

Related

GSDataAmpErr
GSDataGetAmpErr
GSDataTrans
GSErrorBar

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataStoreAux function

Stores auxiliary data value

C/C++

```
int GSDataStoreAux( int nPt, int nAux )
```

FoxPro

```
r = GSDataStoreAux(nPt, nAux)
```

Visual Basic

```
r% = GSDataStoreAux(nPt%, nAux%)
```

Parameters

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.
`nAux`	Auxiliary value

Return values

0	Success
-1	Failure

Description

The GSDataStoreAux function sets the value of a single entry in the Graphics Server auxiliary data array.

The dimension of the array is set by the most recent call to GSDataAux or GSDataTrans.

J

Topic

GSDataStoreAux

Related

GSDataAux
GSDataGetAux
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataStoreClr function

Stores color data value

C/C++

```
int GSDataStoreClr( int nPt, int nClr )
```

FoxPro

```
r = GSDataStoreClr(nPt, nClr)
```

Visual Basic

```
r% = GSDataStoreClr(nPt%, nClr%)
```

Parameters

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.
`nClr`	Color value (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSDataStoreClr function sets the value of a single entry in the Graphics Server color data array.

The dimension of the array is set by the most recent call to GSDataClr or GSDataTrans.

J

Topic

GSDataStoreClr

Related

GSDataClr
GSDataGetClr
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataStoreDist function

Stores distance data value

C/C++

```
int GSDataStoreDist( int nPt, double fDist )
```

FoxPro

```
r = GSDataStoreDist(nPt, fDist)
```

Visual Basic

```
r% = GSDataStoreDist(nPt%, fDist#)
```

Parameters

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.
`fDist`	Distance value

Return values

0	Success
-1	Failure

Description

The GSDataStoreDist function sets the value of a single entry in the Graphics Server distance data array.

The dimension of the array is set by the most recent call to GSDataDist or GSDataTrans.

J

Topic

GSDataStoreDist

Related

GSDataDist
GSDataGetDist
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataStoreDistErr function

Stores distance error value pair

C/C++

```
int GSDataStoreDistErr( int nPt, double fErrPlus,
                        double fErrMinus )
```

FoxPro

```
r = GSDataStoreDistErr(nPt, fErrPlus, fErrMinus)
```

Visual Basic

```
r% = GSDataStoreDistErr(nPt%, fErrPlus#, fErrMinus#)
```

Parameters

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.
`fErrPlus`	Plus error
`fErrMinus`	Minus error

Return values

0	Success
-1	Failure

Description

The GSDataStoreDistErr function sets the value of a single error value pair in the Graphics Server distance error data array. The dimensions of the array are set by the most recent call to the GSDataDistErr function.

J

Topic

GSDataStoreDistErr

Related

GSDataDistErr
GSDataGetDistErr
GSDataTrans
GSErrorBar

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataStorePatt function

Stores pattern data value

C/C++

```
int GSDataStorePatt( int nPt, int nPatt )
```

FoxPro

```
r = GSDataStorePatt(nPt, nPatt)
```

Visual Basic

```
r% = GSDataStorePatt(nPt%, nPatt%)
```

Parameters

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.
`nPatt`	Pattern (see Pattern constants)

Return values

0	Success
-1	Failure

Description

The GSDataStorePatt function sets the value of a single entry in the Graphics Server pattern data array.

The dimension of the array is set by the most recent call to GSDataPatt or GSDataTrans.

J

Topic

GSDataStorePatt

Related

GSDataPatt
GSDataGetPatt
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataStoreSym function

Stores symbol data value

C/C++

```
int GSDataStoreSym( int nPt, int nSymbol )
```

FoxPro

```
r = GSDataStoreSym(nPt, nSymbol)
```

Visual Basic

```
r% = GSDataStoreSym(nPt%, nSymbol%)
```

Parameters

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.
`nSymbol`	Symbol value (see Symbol constants)

Return values

0	Success
-1	Failure

Description

The GSDataStoreSym function sets the value of a single entry in the Graphics Server symbol data array.

The dimension of the array is set by the most recent call to the GSDataSym or GSDataTrans function.

J

Topic

GSDataStoreSym

Related

GSDataSym
GSDataGetSym
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataStoreZ function

Stores Z data value

C/C++

```
int GSDataStoreZ( int nPt, double fZ )
```

FoxPro

```
r = GSDataStoreZ(nPt, fZ)
```

Visual Basic

```
r% = GSDataStoreZ(nPt%, fZ#)
```

Parameters

`nPt`	Index of array element. Array indexes start at zero. A failure value is returned if the index is out of range.
`fZ`	Z data value

Return values

0	Success
-1	Failure

Description

The GSDataStoreZ function sets the value of a single value in the Graphics Server Z data array.

The dimensions of the array are set by the most recent call to GSDataZ.

J

Topic

GSDataStoreZ

Related

GSDataZ
GSDataGetZ
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataSym function

Transfers array of symbol data

C/C++

```
int GSDataSym( int nPts, int* nSymbol )
```

FoxPro

```
r = GSDataSym(nPts, @nSymbol(1))
```

Visual Basic

```
r% = GSDataSym(nPts%, nSymbol%(0))
```

Parameters

`nPts`	Number of elements in symbol array
`nSymbol`	Pointer to symbol data array (see Symbol constants)

Return values

0	Success
-1	Failure

Description

The GSDataSym function transfers symbol data to Graphics Server for use in subsequent graphing functions.

Symbol data is used in some of the graph types but not in others. See the specification of the data arrays of the graphing function to find out if symbol data is required. If the specification shows a symbol array dimension of zero (nSymbol[0]), there's no need to call the GSDataSym function.

The graphing functions use the number of points per data set or, alternatively, the number of data sets--taken from the most recent call to GSDataDim or GSDataTrans--to determine how many symbol array elements are required. See the specification of the data arrays of the graphing function to find out if you need to provide symbol values on a per-point or per-set basis. You must transfer an array whose number of elements is enough to meet the logical requirements of the graph you're going to draw.

GSDataSym stores data in the same array as GSDataTrans, and a call to either function overwrites data stored by the other.

J

Topic

GSDataSym

Related

GSDataPatt
GSDataGetSym
GSDataStoreSym
GSDataTrans

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataTrans function

Transfers data in arrays

C/C++

```
int GSDataTrans( int nPts, int* nGroup, double* fA,
                 double* fD, int* nPatt, int* nSymbol,
                 int* nAux, int* nClr )
```

FoxPro

```
r = GSDataTrans(nPts, nGroup, @fA(1), @fD(1),
                @nPatt(1), @nSymbol(1), @nAux(1),
                @nClr(1))
```

Visual Basic

```
r% = GSDataTrans(nPts%, nGroup%, fA#(0), fD#(0),
                 nPatt%(0), nSymbol%(0), nAux%(0),
                 nClr%(0))
```

Parameters

`nPts`	Number of data points per data set
`nGroup`	Number of data sets
`fA`	Pointer to amplitude data array
`fD`	Pointer to distance data array. GSDataTrans can pass only one-dimensional distance arrays. To pass a two-dimensional distance array, follow GSDataTrans with a call to GSDataDist.
`nPatt`	Pointer to pattern data array (see Pattern constants)
`nSymbol`	Pointer to symbol data array (see Symbol constants)
`nAux`	Pointer to auxiliary data array. GSDataTrans can pass only one-dimensional auxiliary arrays. To pass a two-dimensional auxiliary array, follow GSDataTrans with a call to GSDataAux.
`nClr`	Pointer to color data array (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSDataTrans function transfers a complete set of data arrays to Graphics Server for graphing. A single GSDataTrans function call serves the purpose of seven individual array-transfer functions (GSDataAmp, GSDataAux, GSDataClr, GSDataDist, GSDataPatt, GSDataSym, and GSDataZ).

nPts and nGroup

GSDataTrans uses the nPts and nGroup values to set the logical dimensions of the graph through an implicit call to the GSDataDim function.

Amplitude data (fA)

Amplitude data is the principal data represented in a graph. It determines the magnitudes of pie slices, lengths of bar elements, Y positions of points in a line graph, and so on. Amplitude data is used in all graph types.

The size of the amplitude array is the product of the number of points per data set (or group) and the number of data sets. Data is organized by data set within point.

Certain graph types require a specific number of data sets. For example, a pie chart can only represent one data set at a time, and a high-low-close graph always requires three data sets (high values, low values, and close values).

Distance data (fD) and auxiliary data (nAux)

GSDataTrans presumes that you're supplying one set of distance or auxiliary data to be applied equally to all the amplitude data sets. The size of the distance data array is thus taken to be nPts elements.

In some cases, you have to use multiple sets of distance or amplitude data, creating a two-dimensional fD or nAux array. For example, bubble graphs always require two sets of distance data. GSDataTrans can't pass two-dimensional distance or auxiliary arrays, so you need to call the individual array-transfer functions (GSDataDist or GSDataAmp) in those cases.

If you want to use GSDataTrans along with GSDataDist or GSDataAmp, be sure to call GSDataTrans first.

Array dimensions

The sizes of the integer data arrays are determined by the type of graph you want to draw. Some types of graph use the color, pattern, auxiliary, and symbol data on a per-point basis, while others use it on a per-set basis. Because GSDataTrans isn't aware of the type of graph you're going to draw from the data, it treats all the integer data arrays as being dimensioned to the larger of nPts and nGroup. Take care to pad your arrays appropriately or ensure that unpadded arrays aren't located on a segment high boundary.

If your graph type doesn't use a particular array of data, you can pass a null pointer instead of pointing to an actual array.

Refer to the entries for the individual GS graph functions (GSArea, GSPie2D, and so on) to find out which data arrays you need to transfer and their required dimensions.

J

Topic

GSDataTrans

Related

GSDataAmp
GSDataAux
GSDataClr
GSDataDist
GSDataPatt
GSDataSym
GSDataZ

Array modification:

GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDataZ function

Transfers array of Z data

C/C++

```
int GSDataZ( int nPts, double* fZ )
```

FoxPro

```
r = GSDataZ(nPts, @fZ(1))
```

Visual Basic

```
r% = GSDataZ(nPts%, fZ#(0))
```

Parameters

`nPts`	Number of elements in Z data array
`fZ`	Pointer to Z data array

Return values

0	Success
-1	Failure

Description

The GSDataZ function transfers Z data to Graphics Server for use in subsequent 3D graphing functions.

Z data is used by only a few True3D graph types. See the specification of the data arrays for the graph to find out if Z data is required. If the specification shows a Z array dimension of zero (fZ[0]), there's no need to call the GSDataZ function.

The graphing functions normally use the number of points per data set--taken from the most recent call to GSDataDim or GSDataTrans--to determine how many Z array elements are required. You must transfer an array whose number of elements is enough to meet the logical requirements of the graph you're going to draw.

J

Topic
GSDataZ

Related
GSDataGetZ
GSDataStoreZ
GSDataTrans

Array modification:
GSDataDim
GSDataRange
GSDataReset
GSDataScale
GSTimeUpdate
GSXDataScale

GSDefPatt function

Defines bit pattern for filling

C/C++

```
int GSDefPatt( int nBitmap, WORD wBitmap )
```

FoxPro

```
r = GSDefPatt(nBitmap, @wBitmap(1)))
```

Visual Basic

```
r% = GSDefPatt(nBitmap%, wBitmap%(0))
```

Parameters

`nBitmap`	Bitmap index in the range 0 to BRBITMAPMAX-1. The array comprises eight words. The lower eight bits of each word define the bit pattern, with the least significant bit on the right as viewed on-screen and word 0 at the top.
`wBitmap`	Pointer to array defining the bitmap

Return values

0	Success
-1	Failure

Description

The GSDefPatt function defines a bitmap pattern that can be used to fill areas in drawing or graphing operations. It replaces the internal default pattern.

Different user-defined bitmaps can be defined for each window. Note that bit patterns don't print on vector devices such as plotters.

J

Topic

GSDefPatt

Related

GSArc
GSBox2D
GSBox3D
GSCircle
GSEllipse
GSPolyFill
GSShade

GSEllipse function

Draws ellipse

C/C++

```
int GSEllipse( double fxBL, double fyBL, double fxTR,
               double fyTR, double fxa, double fya,
               double fxb, double fyb, int nMode,
               int nStyle, int nClr )
```

FoxPro

```
r = GSEllipse(fxBL, fyBL, fxTR, fyTR, fxa, nfya, fxb,
              nfyb, nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSEllipse(fxBL#, fyBL#, fxTR#, fyTR#, fxa#, fya#,
               fxb#, fyb#, nMode%, nStyle%, nClr%)
```

Parameters

`fxBL` X bottom left

`fyBL` Y bottom left

`fxTR` X top right

`fyTR` Y top right

`fxa` X start

`fya` Y start

`fxb` X end

`fyb` Y end

`nStyle` Line thickness (nMode ELTHICK) or fill pattern (nMode ELFILL). See Line style constants or Pattern constants.

`nClr` Color of ellipse (see Color constants)

`nMode`

Constant	Value	Meaning
ELRADIUS	1	Draws connecting radii at extremes of ellipse
ELFILL	2	Fills interior with pattern
ELTHICK	4	Uses thick lines

Modes ELFILL and ELTHICK are exclusive. If you enable both of them, only ELFILL is used.

Return values

0	Success
-1	Failure

Description

The GSEllipse function draws an ellipse within a bounding rectangle that is defined by the bottom left and top right corners.

Within the bounding rectangle the ellipse is drawn from point (fxa,fya) to (fxb,fyb). These points must be on or near the

ellipse.

Optionally, radii may be drawn from the center to the end points.

Optionally, the ellipse may be filled with a pattern. In this case nStyle defines the pattern. The outline is drawn with a continuous line.

J

Topic

GSEllipse

Related

GSArc

GSBox2D

GSCircle

GSDefPatt

GSErrorBar function

Defines error bars for graph

C/C++

```
int GSErrorBar( int nSelect, int nSymbolStyle,
                int nColor, int nErrorSource,
                double fValue, double fOffset )
```

FoxPro

```
r = GSErrorBar(nSelect, nSymbolStyle, nColor,
               nErrorSource, fValue, fOffset)
```

Visual Basic

```
r% = GSErrorBar(nSelect%, nSymbolStyle%, nColor%,
                nErrorSource%, fValue#, fOff#)
```

Parameters

`nSelect`

Constant	Value	Meaning
EBY	0	Defines error bars for Y (amplitude) data
EBX	1	Defines error bars for X (distance) data

`nSymbolStyle`

Constant	Value	Meaning
	0	Full symbol
EBNOPLUS	1	Omits plus bar
EBNOMINUS	2	Omits minus bar
EBNOSTEM	4	Omits stem
EBNOTICK	8	Omits cross tick

`nColor`

Error bar color (see Color constants)

`nErrorSource`

Constant	Value	Meaning
EBFIXED	0	Error is fixed value equal to fValue
EBPERCENT	1	Error is fValue (here a percentage) times the data value, divided by 100
EBSTDDEV	2	Error is standard deviation times fValue
EBSTDERR	3	Error is standard error
EBMAXMIN	4	Errors are provided in arrays

`fValue`

Value modifier. fValue is used in a variety of ways, depending on the error source: as a fixed value for all data points (EBFIXED), as a modifier when the error is expressed as a percentage of

	the data value (EBPERCENT), and as a multiplying factor when the error is expressed as the standard deviation of the data set (EBSTDDEV).
`fOffset`	Value to be added to data when using the EBPERCENT mode. This is only relevant when the data has been pre-adjusted for a non-zero graph origin by subtracting an offset. fOffset is used to restore the data to its unadjusted form before calculating the percentage of data value.

Return values

0	Success
-1	Failure

Description

The GSErrorBar function defines the format of error bars to be added to a graph. You have to call it before calling the graphing function for the graph type, such as GSBar2D.

You can specify both horizontal and vertical error bars. To use both, you have to call the function twice--first with nSelect EBY, then with nSelect EBX.

Error bars can be applied to the following 2D graph types:

- Horizontal bar graphs (simple and clustered format)
- Vertical bar graphs (simple and clustered formats)
- Line graphs (line and symbol formats)
- Scatter graphs

X (distance data) error bars can be added only to scatter graphs.

You can use error bars along the linear axis in lin/log and log/lin graphs, but never along the logarithmic axis.

Error bars automatically accommodate a change of scale made by the function GSDataScale or GSXDataScale.

With mode EBMAXMIN, the error is supplied in the arrays fAmpErr and fDistErr, as set by the functions GSDataAmpErr and GSDataDistErr. Errors are supplied as paired *plus* and *minus* error values, both of which must be positive.

Topic

GSErrorBar

Related

GSDataAmpErr
GSDataDistErr
GSDataScale
GSXDataScale
GSDataGetDistErr
GSDataStoreAmpErr

GSDataStoreDistErr

GSFixPos function

Fixes current position

C/C++

```
int GSFixPos( double fx, double fy )
```

FoxPro

```
r = GSFixPos(fx, fy)
```

Visual Basic

```
r% = GSFixPos(fx#, fy#)
```

Parameters

`fx`	Current X
`fy`	Current Y

Return values

0	Success
-1	Failure

Description

The GSFixPos function fixes the current position in terms of (X,Y) view coordinates.

Certain drawing functions are able to draw relative to the current position and optionally move the current position to their drawing end point. The current position is set to (0,0) when a view is opened.

J

Topic

GSFixPos

Related

GSMovePos
GSLineRel
GSGetCurX
GSGetCurY
GSPolyFill
GSPolyVec

GSGantt function

Draws Gantt chart

C/C++

```
int GSGantt( double fxOrg, double fyOrg, double fInc,
             int nMode, int nGroup )
```

FoxPro

```
r = GSGantt(fxOrg, fyOrg, fInc, nMode, nGroup)
```

Visual Basic

```
r% = GSGantt(fxOrg#, fyOrg#, fInc#, nMode%, nGroup%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fInc` Y increment (vertical distance in view units between adjacent data points)

`nMode`

Constant	Value	Meaning
GAVARY	1	Takes the Y position from the fD array
GASPACE	2	Inserts a space between adjacent bars

`nGroup` Group size

Return values

0	Success
-1	Failure

Description

The GSGantt function draws a Gantt chart. The Gantt chart is similar in format to a horizontal 2D stacked bar graph, except that the bars are free to be positioned away from the axis. The position and length of the bars is determined by the amplitude array.

The amplitude array has an additional data set compared to the pattern and color arrays. The first data set defines the height of the base of the bar stack above the axis. This data set has no corresponding entries in the pattern and color arrays.

The second and subsequent amplitude data sets define the height of each bar end above the axis. The first entries in the pattern and color data arrays are applied to the second amplitude data set and so on.

Optionally the Y position of the bars relative to the origin may be taken from the fD array rather than at fixed intervals.

GSDataTrans parameters for Gantt charts

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (always at least 2)

fA[nPts][nGroup]	Pointer to amplitude array containing X positions of bars (first set has starting points of first bars, second set has ending points of first bars, subsequent sets have ending points of subsequent bars)
fD[nPts]	Pointer to distance array (Y positions of bars)--used only with nMode GAVARY
nPatt[nGroup-1]	Pointer to array containing fill patterns of bar segments
nSymbol[0]	Not used
nAux[0]	Not used
nClr[nGroup-1]	Pointer to array containing colors of bar segments

J

Topic

GSGantt

Related

GSBar2D

Axis/grid/legend:

GSAxis
GSGrid
GSLegend

Labels:

GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:

GSDataTrans

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

GSGetACos function

Gets arccosine

C/C++

```
double GSGetACos( double fVal )
```

FoxPro

```
r = GSGetACos(fVal)
```

Visual Basic

```
r# = GSGetACos(fVal#)
```

Parameter

`fVal`	Angle (in degrees) for which to determine arccosine

Return value

Arccosine of angle

Description

The GSGetACos function returns the arccosine of the given angle.

J

Topic

GSGetACos

Related

GSGetCos

GSGetALog10 function

Gets antilog base 10

C/C++

```
double GSGetALog10( double fVal )
```

FoxPro

```
r = GSGetALog10(fVal)
```

Visual Basic

```
r# = GSGetALog10(fVal#)
```

Parameter

`fVal`	Number for which to determine antilog

Return value

Antilog base 10 of number

Description

The GSGetALog10 function returns the antilog base 10 of the given number.

J

Topic

GSGetALog10

Related

GSGetLog10
GSGetALogE
GSGetLogE

GSGetALogE function

Gets natural antilog base e

C/C++

```
double GSGetALogE( double fVal )
```

FoxPro

```
r = GSGetALogE(fVal)
```

Visual Basic

```
r# = GSGetALogE(fVal#)
```

Parameter

`fVal`	Number for which to determine natural antilog

Return value

Natural antilog base e of number (-1 if failure)

Description

The GSGetALogE function returns the natural antilog of the given number.

J

Topic

GSGetALogE

Related

GSGetLogE
GSGetALog10
GSGetLog10

GSGetASin function

Gets arcsine

C/C++

```
double GSGetASin( double fVal )
```

FoxPro

```
r = GSGetASin(fVal)
```

Visual Basic

```
r# = GSGetASin(fVal#)
```

Parameter

`fVal`	Angle (in degrees) for which to determine arcsine

Return value

Arcsine of angle

Description

The GSGetASin function returns the arcsine of the given angle.

J

Topic

GSGetASin

Related

GSGetSin

GSGetATan function

Gets arctangent

C/C++

```
double GSGetATan( double fVal )
```

FoxPro

```
r = GSGetATan(fVal)
```

Visual Basic

```
r# = GSGetATan(fVal#)
```

Parameter

`fVal`	Angle (in degrees) for which to determine arctangent

Return value

Arctangent of angle

Description

The GSGetATan function returns the arctangent of the given angle.

J

Topic

GSGetATan

Related

GSGetTan

GSGetAXExt function

Gets anchor space X extent

C/C++

```
double GSGetAXExt( HWND hWnd )
```

FoxPro

```
r = GSGetAXExt(hWnd)
```

Visual Basic

```
r# = GSGetAXExt(hWnd%)
```

Parameter

`hWnd`	Windows handle of the window to be measured. If you use an hWnd of zero, the anchor space is mapped into the whole screen, and the width of the whole screen in anchor units is returned. In this mode, the value returned by GSGetAXExt is the same as that returned by GSGetSXExt.

Return values

0 or greater	X extent of the anchor space
-1	Failure

Description

The GSGetAXExt function maps a local anchor space into the client area of a window and returns the X extent (width) in anchor units. Anchor units are device-independent units used to specify the origin and size of a new graphing window.

GSGetAXExt, together with the complementary GSGetAYExt, is typically used to measure the client area of a parent window in anchor units prior to opening a new graphing child window.

J

Topic

GSGetAXExt

Related

GSGetSXExt
GSGetWXExt
GSGetAYExt
GSOpenChildWin

GSGetAYExt function

Gets anchor space Y extent

C/C++

```
double GSGetAYExt( HWND hWnd )
```

FoxPro

```
r = GSGetAYExt(hWnd)
```

Visual Basic

```
r# = GSGetAYExt(hWnd%)
```

Parameter

`hWnd`	Windows handle of the window to be measured. If you use an hWnd of zero, the anchor space is mapped into the whole screen and the width of the whole screen in anchor units is returned. In this mode, the value returned by GSGetAYExt is the same as that returned by GSGetSYExt.

Return values

0 or greater	Y extent of the anchor space
-1	Failure

Description

The GSGetAYExt function maps a local anchor space into the client area of a window and returns the Y extent (height) in anchor units. Anchor units are device-independent units used to specify the origin and size of a new graphing window.

GSGetAYExt, together with the complementary GSGetAXExt, is typically used to measure the client area of a parent window in anchor units prior to opening a new graphing child window.

J

Topic

GSGetAYExt

Related

GSGetSYExt
GSGetWYExt
GSGetAXExt
GSOpenChildWin

GSGetBG function

Gets background color

C/C++

```
int GSGetBG( )
```

FoxPro

```
r = GSGetBG()
```

Visual Basic

```
r% = GSGetBG()
```

Return values

0 or greater	Current background color index
-1	Failure

Description

The GSGetBG function returns the color index of the current background color.

J

Topic

GSGetBG

Related

GSSetBg

GSClearView

GSGetCC function

Gets linear correlation coefficient

C/C++

```
double GSGetCC( )
```

FoxPro

```
r = GSGetCC()
```

Visual Basic

```
r# = GSGetCC()
```

Return value

Linear regression correlation coefficient in the range -1.0000 to 1.0000 (-1 if failure)

Description

The GSGetCC function returns the correlation coefficient of the least-squares regression of Y over X of the current data set.

The value is calculated from the complete set of points within the data array unless modified by GSDataRange, in which case it applies to the selected subset.

Since the correlation coefficient is dependent on both X and Y, it's affected by the scale factor applied to X and Y that should be the same for consistency.

Note that if data is fixed increment, the graphing function must be called first to establish the X increment.

J

Topic

GSGetCC

Related

GSLineFit

GSCurveFit

GSDataRange

GSGetCos function

Gets cosine

C/C++

```
double GSGetCos( double fVal )
```

FoxPro

```
r = GSGetCos(nVal)
```

Visual Basic

```
r# = GSGetCos(fVal#)
```

Parameter

`fVal`	Angle (in degrees) for which to determine cosine

Return value

Cosine of angle

Description

The GSGetCos function returns the cosine of the given angle.

J

Topic

GSGetCos

Related

GSGetACos

GSGetCurveCoeff function

Gets curve coefficient

C/C++

```
double GSGetCurveCoeff( )
```

FoxPro

```
r =  GSGetCurveCoeff()
```

Visual Basic

```
r# = GSGetCurveCoeff()
```

Return value

Next curve coefficient in sequence

Description

The GSGetCurveCoeff function returns the coefficients of the equation of the curve most recently drawn by GSCurveFit.

The function is called repeatedly to return all the coefficients. For example, in the quadratic *y = a + bx + cx2,* the first time you call the function after GSCurveFit, it returns the value of *a,* the second time *b,* and the third time *c.* If you call it again, it returns *a* again, then *b,* and so on. In other words, the function cycles repeatedly around the coefficients of the selected curve.

You can use GSGetCurveCoeff to get the curve coefficients after calling AGShow for a scatter graph with a fitted curve. In this case, beware that the returned coefficients are expressed in view units, rather than the natural units of the data you supplied through AGAmp. This is due to the automatic scaling of data in AutoGraph.

Coefficients returned in view units may be converted back into the units of your data. *y = a + bx + cx2* in view coordinates is equivalent to *yscale * y = a + b(xscale * x) + c(xscale * x)2* or *Y = a / yscale + b(xscale * x) / yscale + c(xscale * x)2 / yscale.*

To obtain the scaling factors used to draw the graph from AGInfo, set *yscale* = AGInfo(5) / AGInfo(2) - AGInfo(3) and set *xscale* = AGInfo(4) / AGInfo(0) - AGInfo(1). Now you can calculate your A, B, and C: *A = a / yscale; B = b * xscale / yscale; C = c * xscale2 / yscale.*

J

Topic

GSGetCurveCoeff

Related

GSCurveFit

AGShow

GSGetCurX function

Gets current X position

C/C++

```
double GSGetCurX( )
```

FoxPro

```
r = GSGetCurX()
```

Visual Basic

```
r# = GSGetCurX()
```

Return value

Current X position in view units

Description

The GSGetCurX function returns the current X position in the view.

J

Topic

GSGetCurX

Related

GSGetCurY
GSFixPos
GSMovePos
GSLineRel
GSPolyFill
GSPolyVec

GSGetCurY function

Gets current Y position

C/C++

```
double GSGetCurY( )
```

FoxPro

```
r = GSGetCurY()
```

Visual Basic

```
r# = GSGetCurY()
```

Return value

Current Y position in view units (-1 if failure)

Description

The GSGetCurY function returns the current Y position in the view.

J

Topic

GSGetCurY

Related

GSGetCurX
GSFixPos
GSMovePos
GSLineRel
GSPolyFill
GSPolyVec

GSGetE function

Gets natural exponent

C/C++

```
double GSGetE( )
```

FoxPro

```
r = GSGetE()
```

Visual Basic

```
r# = GSGetE()
```

Return value

Natural exponent

Description

The GSGetE function returns the natural exponent e.

J

Topic

GSGetE

Related

GSGetPi

GSGetLog10 function

Gets log base 10

C/C++

```
double GSGetLog10( double fVal )
```

FoxPro

```
r = GSGetLog10(fVal)
```

Visual Basic

```
r# = GSGetLog10(fVal#)
```

Parameters

`fVal`	Number for which to determine base 10 logarithm

Return value

Log base 10 of number

Description

The GSGetLog10 function returns the base 10 logarithm of the given number.

J

Topic

GSGetLog10

Related

GSGetALog10
GSGetALogE
GSGetLogE

GSGetLogE function

Gets natural log

C/C++

```
double GSGetLogE( double fVal )
```

FoxPro

```
r = GSGetLogE(fVal)
```

Visual Basic

```
r# = GSGetLogE(fVal#)
```

Parameter

`fVal`	Number for which to determine natural logarithm

Return value

Natural logarithm number (-1 if failure)

Description

The GSGetLogE function returns the natural logarithm of the given number.

J

Topic

GSGetLogE

Related

GSGetALogE

GSGetLog10

GSGetALogE

GSGetMax function

Gets maximum amplitude data value

C/C++

```
double GSGetMax( )
```

FoxPro

```
r = GSGetMax()
```

Visual Basic

```
r# = GSGetMax()
```

Return value

Maximum data value in amplitude array

Description

The GSGetMax function returns the maximum data value in the amplitude array.

In arrays with more than one set of data, the maximum is for all the sets.

The value is calculated from the complete data set unless reduced to a subset by the GSDataRange function. It's also modified by any prevailing GSDataScale function.

J

Topic

GSGetMax

Related

GSGetMin

GSDataRange

GSDataScale

GSGetMean function

Gets mean data value of amplitude array

C/C++

```
double GSGetMean( )
```

FoxPro

```
r = GSGetMean()
```

Visual Basic

```
r# = GSGetMean()
```

Return value

Mean data value of amplitude array

Description

The GSGetMean function returns the mean data value of the amplitude array.

In arrays with more than one set of data, the mean is for all the sets.

The value is calculated from the complete data set unless reduced to a subset by the GSDataRange function. It's also modified by any prevailing GSDataScale function.

J

Topic

GSGetMean

Related

GSGetSD
GSDataRange
GSDataScale
GSMean

GSGetMF function

Gets image metafile

C/C++

```
HANDLE GSGetMF( nMode )
```

FoxPro

```
r = GSGetMF(nMode)
```

Visual Basic

```
r% = GSGetMF(nMode%)
```

Parameter

`nMode`	Function mode

Return values

Greater than 0	Windows handle to a memory metafile
-1	Failure

Description

The GSGetMF function returns a handle to a memory metafile of the image in the current window.

All dimensions in the metafile are expressed in units of 0.05 of a millimeter. The image is ISOTROPIC, which means that a line of *n* units drawn along the X axis will appear on any output device the same length as a similar line drawn along the Y axis. The natural dimensions of the image may be obtained using the GSGetWXExt and GSGetWYExt functions in the GWWHOLE mode.

Graphics Server uses a traditional Cartesian coordinate system, which is inverted in the Y (vertical) dimension relative to the Windows device coordinate system. In Graphics Server, the origin (X0,Y0) is at the bottom left, with Y coordinates increasing toward the top of the page; Windows devices have their origin at the top left, with Y coordinates increasing toward the bottom. Graphics Server transfers coordinates between the two systems by changing the sign of the logical Y extent, using a call to the Windows SetWindowExt function in the body of the metafile returned by GSGetMF.

Example

The following example displays a memory metafile obtained from GSGetMF:

```
hMF = GSGetMF(0);

SetMapMode(hDC, MM_ISOTROPIC);
SetViewportExt(hDC, ViewportWidth, ViewportHeight);
SetViewportOrg(hDC, ViewportOriginX,
               ViewportOriginY);
PlayMetaFile(hDC, hMF);
```

The metafile obtained from GSGetMF belongs to your application, so remember to delete it when you're through with it:

```
DeleteMetaFile(hMF);
```

J

Topic

GSGetMF

Related

GSPicRead
GSPicWrite
GSClipRead
GSClipWrite
GSWinPaint

GSGetMin function

Gets minimum amplitude data value

C/C++

```
double GSGetMin( )
```

FoxPro

```
r = GSGetMin()
```

Visual Basic

```
r# = GSGetMin()
```

Return value

Minimum data value in amplitude array

Description

The GSGetMin function returns the minimum data value in the amplitude array.

In arrays with more than one set of data, the minimum is for all the sets.

The value is calculated from the complete data set unless reduced to a subset by the GSDataRange function. It's also modified by any prevailing GSDataScale function.

J

Topic

GSGetMin

Related

GSGetMax

GSGetPI function

Gets value of pi

C/C++ `double GSGetPI( )`

FoxPro `r = GSGetPI()`

Visual Basic `r# = GSGetPI()`

Return value Pi (or -1 if failure)

Description The GSGetPI function returns the value of pi (3.1416).

J

Topic
GSGetPI

Related
GSGetE

GSGetPrnHt function

Gets printer paper height

C/C++

```
double GSGetPrnHt( int nUnits )
```

FoxPro

```
r = GSGetPrnHt(nUnits)
```

Visual Basic

```
r# = GSGetPrnHt(nUnits%)
```

Parameter

`nUnits`

Constant	**Value**	**Meaning**
UNMM	1	Millimeters
UNINCH	2	Inches

Return values

0 or greater	Printer paper height in the selected units
-1	Failure

Description

The GSGetPrnHt function returns the paper height of the printer currently selected using the GSOpenPrn function. The height is returned in physical units of inches or millimeters.

J

Topic

GSGetPrnHt

Related

GSGetPrnWid
GSPrnSetup
GSPrnOut
GSOpenPrn

GSGetPrnWid function

Gets printer paper width

C/C++

```
double GSGetPrnWid(int nUnits )
```

FoxPro

```
r = GSGetPrnWid(nUnits)
```

Visual Basic

```
r# = GSGetPrnWid(nUnits%)
```

Parameter

nUnits	**Constant**	**Value**	**Meaning**
	UNMM	1	Millimeters
	UNINCH	2	Inches

Return values

0 or greater	Printer paper width in the selected units
-1	Failure

Description

The GSGetPrnWid function returns the paper width of the printer currently selected using the GSOpenPrn function. The width is returned in physical units of inches or millimeters.

J

Topic

GSGetPrnWid

Related

GSGetPrnHt
GSPrnSetup
GSPrnOut
GSOpenPrn

GSGetRTextHt function

Gets raster text height

C/C++

```
double GSGetRTextHt( int nCSet, int nTMode,
                     char szString )
```

FoxPro

```
r = GSGetRTextHt(nCSet, nTMode, szString)
```

Visual Basic

```
r# = GSGetRTextHt(nCSet%, nTMode%, szString$)
```

Parameters

`nCSet`	Character set (see Character set constants)
`nTMode`	Text mode (see Text mode constants)
`szString`	Text string

Return values

0 or greater	Height of text string in view units
-1	Failure

Description

The GSGetRTextHt function returns the height of a text string in view units.

J

Topic

GSGetRTextHt

Related

GSGetRTextWid
GSRText
GSGetSFHt

GSGetRTextWid function

Gets raster text width

C/C++

```
double GSGetRTextWid( int nCSet, int nTMode,
                      char szString )
```

FoxPro

```
r = GSGetRTextWid(nCSet, nTMode, szString)
```

Visual Basic

```
r# = GSGetRTextWid(nCSet%, nTMode%, szString$)
```

Parameters

`nCSet`	Character set (see Character set constants)
`nTMode`	Text mode (see Text mode constants)
`szString`	Text string

Return values

0 or greater	Width of text string in view units
-1	Failure

Description

The GSGetRTextWid function returns the width of a text string in view units.

J

Topic

GSGetRTextWid

Related

GSGetRTextHt
GSRText
GSGetSFWid

GSGetSD function

Gets standard deviation of data set

C/C++

```
double GSGetSD( )
```

FoxPro

```
r = GSGetSD()
```

Visual Basic

```
r# = GSGetSD()
```

Return values

Standard deviation of data set

Description

The GSGetSD function returns the standard deviation of the current data set.

The value is calculated from the complete data set unless reduced to a subset by the GSDataRange function. It's also modified by any prevailing GSDataScale function.

J

Topic

GSGetSD

Related

GSGetMean
GSDataRange
GSDataScale
GSSD

GSGetSFHt function

Gets height of system font characters

C/C++

```
double GSGetSFHt( )
```

FoxPro

```
r = GSGetSFHt()
```

Visual Basic

```
r# = GSGetSFHt()
```

Return values

0 or greater	Height of system font characters in view units
-1	Failure

Description

The GSGetSFHt function returns the height of the system font characters in view units. This height can be used to calculate line spacing.

J

Topic

GSGetSFHt

Related

GSGetSFWid
GSGetRTextHt
GSRText

GSGetSFWid function

Gets width of system font characters

C/C++

```
double GSGetSFWid( )
```

FoxPro

```
GSGetSFWid()
```

Visual Basic

```
r# = GSGetSFWid()
```

Return values

0 or greater	Width of system font characters in view units
-1	Failure

Description

The GSGetSFWid function returns the width of the system font characters in view units. This width can be used to calculate text lengths.

J

Topic

GSGetSFWid

Related

GSGetSFHt
GSGetRTextWid
GSRText

GSGetSin function

Gets sine

C/C++ `double GSGetSin( fVal )`

FoxPro `r = GSGetSin(fVal)`

Visual Basic `r# = GSGetSin(fVal#)`

Parameter `fVal` Angle (in degrees) for which to determine sine

Return value Sine of angle

Description The GSGetSin function returns the sine of the given angle.

J

Topic
GSGetSin

Related
GSGetASin

GSGetSXExt function

Gets screen X extent

C/C++

```
double GSGetSXExt( )
```

FoxPro

```
r = GSGetSXExt()
```

Visual Basic

```
r# = GSGetSXExt()
```

Return values

0 or greater	X extent of the screen
-1	Failure

Description

The GSGetSXExt function maps an anchor space into the whole screen and returns the X extent in anchor units.

Anchor units are device-independent units used to specify the origin and size of a new graphing window. This function, together with the complementary GSGetSYExt, is typically used to measure the screen in anchor units prior to opening a new graphing window.

J

Topic

GSGetSXExt

Related

GSGetSYExt
GSGetAXExt
GSOpenWin

GSGetSYExt function

Gets screen Y extent

C/C++

```
double GSGetSYExt( )
```

FoxPro

```
r = GSGetSYExt()
```

Visual Basic

```
r# = GSGetSYExt()
```

Return values

0 or greater	Y extent of the screen
-1	Failure

Description

TheGSGetSYExt function maps an anchor space into the whole screen and returns the Y extent in anchor units.

Anchor units are device-independent units used to specify the origin and size of a new graphing window. This function, together with the complementary GSGetSXExt, is typically used to measure the screen in anchor units prior to opening a new graphing window.

J

Topic

GSGetSYExt

Related

GSGetSXExt
GSGetAYExt
GSOpenWin

GSGetTan function

Gets tangent

C/C++

```
double GSGetTan( double fVal )
```

FoxPro

```
r = GSGetTan(fVal)
```

Visual Basic

```
r# = GSGetTan(fVal#)
```

Parameter

`fVal`	Angle (in degrees) for which to determine tangent

Return values

Tangent of angle

Description

The GSGetTan function returns the tangent of the given angle.

J

Topic

GSGetTan

Related

GSGetATan

GSGetVer function

Gets server or DLL version number

C/C++

```
int GSGetVer( nVer )
```

FoxPro

```
r = GSGetVer(nVer)
```

Visual Basic

```
r% = GSGetVer(nVer%)
```

Parameter `nVer`

Constant	**Value**	**Meaning**
GVSERVER	0	Gets Graphics Server EXE version number
GVDLL	1	Gets Graphics Server DLL version number

Return values

0 or greater	Version number of the specified component
-1	Failure

Description

The GSGetVer function returns the version number of the Graphics Server EXE or DLL module.

The high-order byte of the return value contains the minor version number, and the low-order byte contains the major version number.

GSGetVXExt function

Gets view X extent

C/C++

```
double GSGetVXExt( )
```

FoxPro

```
r = GSGetVXExt()
```

Visual Basic

```
r# = GSGetVXExt()
```

Return values

0 or greater	X extent of the current view in view units
-1	Failure

Description

The GSGetVXExt function returns the X extent of the current view in view units.

J

Topic

GSGetVXExt

Related

GSClearView
GSCloseView
GSGetVYExt
GSGetWXExt
GSOffView
GSOnView
GSOpenView
GSUseView

GSGetVYExt function

Gets view Y extent

C/C++

```
double GSGetVYExt( )
```

FoxPro

```
r = GSGetVYExt()
```

Visual Basic

```
r# = GSGetVYExt()
```

Return values

0 or greater	Y extent of the current view in view units
-1	Failure

Description

The GSGetVYExt function returns the Y extent of the current view in view units.

J

Topic

GSGetVYExt

Related

GSClearView
GSCloseView
GSGetVXExt
GSGetWYExt
GSOffView
GSOnView
GSOpenView
GSUseView

GSGetWXExt function

Gets window X extent

C/C++

```
double GSGetWXExt( int nMode, int nUnits )
```

FoxPro

```
r = GSGetWXExt(nMode, nUnits)
```

Visual Basic

```
r# = GSGetWXExt(nMode%, nUnits%)
```

Parameters

nMode	Constant	Value	Meaning
	GWWHOLE	0	Returns the whole extent
	GWCLIPPED	1	Returns visible extent

nUnits	Constant	Value	Meaning
	UNLOG	0	Anchor units
	UNMM	1	Millimeters
	UNINCH	2	Inches
	UNDEV	3	Device units (pixels)

Return values

0 or greater	X extent of window
-1	Failure

Description

The GSGetWXExt function returns the X extent of the current window. You can have it return either the whole extent (including invisible portions) or just the visible part.

Topic

GSGetWXExt

Related

GSGetWYExt
GSGetVXExt
GSGetAXExt
GSOpenView

GSGetWYExt function

Gets window Y extent

C/C++

```
double GSGetWYExt( int nMode, int nUnits )
```

FoxPro

```
r = GSGetWYExt(nMode, nUnits)
```

Visual Basic

```
r# = GSGetWYExt(nMode%, nUnits%)
```

Parameters

`nMode`

Constant	Value	Meaning
GWWHOLE	0	Returns the whole extent
GWCLIPPED	1	Returns visible extent

`nUnits`

Constant	Value	Meaning
UNLOG	0	Anchor units
UNMM	1	Millimeters
UNINCH	2	Inches
UNDEV	3	Device units (pixels)

Return values

0 or greater	Y extent of window
-1	Failure

Description

The GSGetWYExt function returns the Y extent of the current window. You can have it return either the whole extent (including invisible portions) or just the visible part.

Topic

GSGetWYExt

Related

GSGetWXExt
GSGetVYExt
GSGetAYExt
GSOpenView

GSGrid function

Draws grid lines

C/C++

```
int GSGrid( double fxOrg, double fyOrg,
            double fAxisLen, double fGridLen,
            int nDivs, int nMode, int nStyle,
            int nClr )
```

FoxPro

```
r = GSGrid(fxOrg, fyOrg, fAxisLen, fGridLen, nDivs,
           nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSGrid(fxOrg#, fyOrg#, fAxisLen#, fGridLen#,
            nDivs%, nMode%, nStyle%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fAxisLen` Length of axis

`fGridLen` Length of grid

`nDivs` Number of divisions along length of axis

`nMode`

Constant	Value	Meaning
GRX	0	Draws X grids (parallel to Y axis)
GRY	1	Draws Y grids (parallel to X axis)
GRNOFIRST	2	Omits the first grid line
GRNOLAST	4	Omits the last grid line

`nStyle` Line style (see Line style constants)

`nClr` Color of grid lines (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSGrid function draws grid lines at regular intervals along the X axis (grid lines parallel to the Y axis) or along the Y axis (grid lines parallel to the X axis).

You can omit the first and last grid lines to avoid overdrawing axes or frames.

J

Topic

GSGrid

Related

GSAxis
GSLogGrid
GSXYGraph

GSHLC function

Draws high-low-close, open-high-low-close, or candlestick graph

C/C++

```
int GSHLC( double fxOrg, double fyOrg, double fInc,
           int nMode, int nClr )
```

FoxPro

```
r = GSHLC(fxOrg, fyOrg, fInc, nMode, nClr)
```

Visual Basic

```
r% = GSHLC(fxOrg#, fyOrg#, fInc#, nMode%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fInc` X increment (distance in view units between adjacent data points). Optionally, you can take the position of points relative to the origin from the fD array rather than placing points at fixed intervals.

`nMode`

Constant	Value	Meaning
HLCVARX	1	Takes X position from the fD array
HLCNOCLOSE	2	Omits close bar
HLCNOBARS	4	Omits high, low, and close bars, drawing vertical line only
HLCTHICK	8	Draws symbol with thick lines
HLCOPEN	16	Open-high-low-close format
HLCCANDLESTICK	32	Candlestick format (special version of open-high-low-close)

`nClr` Color of symbols (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSHLC function draws a high-low-close, open-high-low-close, or candlestick graph.

Each high-low-close symbol consists of a vertical line intersected

by horizontal high, low, and close bars. Open-high-low-close graphs add a fourth bar for the open value.

You can omit horizontal bars from the symbol, leaving just a vertical line. You can also omit the close bars (and open bars, if used) from the symbols, but you still have to provide closing (and opening) values in the data array.

Candlestick graphs

The candlestick graph is a special case of the open-high-low-close graph. It consists of a series of boxes with lines extending up and down from the ends. The top and bottom of each box indicate the open and close values. If the open value is higher, the box is filled with a color; if the close value is higher, the box is filled with white. The ascending and descending lines indicate the high and low values for that point.

GSDataTrans parameters for high-low-close, open-high-low-close, and candlestick graphs

nPts	Number of symbols (no limit)
nGroup	Number of data sets (always 3 for high-low-close and 4 for open-high-low-close and candlestick)
fA[nPts][nGroup]	Pointer to amplitude array J *For high-low-close,* first set is high values, second set is low values, third set is close values J *For open-high-low-close and candlestick,* first set is open values, second set is high values, third set is low values, fourth set is close values
fD[nPts]	Pointer to distance array (X positions of symbols)--used only with nMode HLCVARX
nPatt[0]	Not used
nSymbol[0]	Not used
nAux[0]	Not used
nClr[0]	Not used

J

Topic

GSHLC

Related

GSBoxWhisker

GSStatsArr

Axis/grid/legend:
GSAxis
GSGrid
GSLegend

Labels:
GSDataLabels
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:
GSDataTrans

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSHotGraph function

Enables and disables hot graphing

C/C++

```
int GSHotGraph( int nMode )
```

FoxPro

```
r = GSHotGraph(nMode)
```

Visual Basic

```
r% = GSHotGraph(nMode%)
```

Parameter

nMode	**Constant**	**Value**	**Meaning**
	FALSE	0	Disables hot graphing
	TRUE	1	Enables hot graphing

Return values

0	Success
-1	Failure

Description

The GSHotGraph function enables and disables the drawing of "hot graphs" by Graphics Server. A hot graph is one in which a mouse click on any of the points of the graph will feed a notification message--identifying the clicked point and data set number--to the client application.

Hot graphing is applied on a per-window basis. One window can have only one hot graph in it at a time. However, new hot graphs can be drawn successively into the same window, with each new graph automatically canceling the hot regions of the previous one. You can have different hot graphs active in different windows at the same time.

GSHotGraph always acts on the current window. Use the GSUseView function to change the current window.

nMode is set non-zero to enable hot graphing and zero to disable it. The hot-graph mode is applied to all the graphs drawn subsequently in the window, both by standard graph functions and by AutoGraph. Call it with 1+16 to enable it for the primary graph when you've created an AutoGraph overlay graph; otherwise, the hot events will only come from the second (overlay) graph.

Hot graphs and Graphics Server's mouse event notification system are closely related. In addition to calling GSHotGraph to enable the drawing of hot graphs, you must call GSMNotify to enable the hot-graph event notification to your application.

Example

The following example illustrates how to open a graphing window and enable it for hot-graph operation:

```
#define WM_HITPOINT  (WM_USER + 1)

double ScreenWid, ScreenHt;
int WinNum;
```

```
ScreenWid = GSGetSXExt();
ScreenHt = GSGetSYExt();
WinNum = GSOpenWin( 0.10 * ScreenWid,
                    0.10 * ScreenHt,
                    0.50 * ScreenWid,
                    0.50 * ScreenHt,
                    1000, 0, OWMFIXED,
                    "Hot-graph Window" );
if ( WinNum < 0 ) {
   /* GSOpenWin failed */
}
/* enable hot graphs and event notification */
GSHotGraph( TRUE );
GSMNotify( hWndHdlr, WM_HITPOINT, MNHITPT );
/* collect some data and eventually draw a graph
this will draw a hot graph */
AGShow( AGBAR2D, 0, 0 );
```

The window procedure for the window hWndHdlr might contain the following:

```
LONG WndHdlr_WndProc( HWND hWnd, UINT uMsg, UINT
                      wParam, LONG lParam )
{
switch ( uMsg ) {
case WM_HITPOINT:
/* the user clicked a point in the graph so get the point and
data set number from the message */
PointNum = LOWORD( lParam );
SetNum = HIWORD( lParam );
/* etc. */
}
}
```

J

Topic
GSHotGraph

Related
GSMNotify
GSUseView

GSLabelnPie function

Draws pie chart numeric labels

C/C++

```
int GSLabelnPie( double fxOff, double fRad,
                 double fWid, double fHt, int nPrec,
                 int nMode, int nCSet, int nTMode,
                 int nClr )
```

FoxPro

```
r = GSLabelnPie(fxOff, fRad, fWid, fHt, nPrec, nMode,
                nCSet, nTMode, nClr)
```

Visual Basic

```
r% = GSLabelnPie(fxOff#, fRad#, fWid#, fHt#, nPrec%,
                 nMode%, nCSet%, nTMode%, nClr%)
```

Parameters

`fxOff` Horizontal offset

`fRad` Radius of the arc on which the labels are drawn. This radius must be at least 1.1 times greater than the pie radius (or 1.35 times greater if any segments are exploded). The pie radius is taken from the preceding GSPieChart function call.

`fWid` Width of labels

`fHt` Height of labels

`nPrec` Decimal precision

`nMode`

Constant	Value	Meaning
LPSEGCLR	1	Colors labels the same as segments
LPNOLINES	2	Omits pointing lines from pie to label
LPPC	4	Shows percentage rather than magnitude
LPPCCHAR	8	Appends percentage character to each label
LPSMART	16	Auto-arranges labels to avoid overlapping

`nCSet` Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.

`nTMode`	Text mode (see Text mode constants)
`nClr`	Color of labels (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSLabelnPie function draws a sequence of numeric labels to complement a pie chart. The label may be either the value passed as data or the value expressed as a percentage of the whole.

The pie chart must be drawn first since this function adopts certain parameters from the preceding pie chart function call.

The angular position of each label is calculated from the data for the pie chart. The labels are drawn in an arc on each side of the pie, connected to their respective segments by pointing lines.

These lines are drawn from the label horizontally a distance fXOff, then radially toward the center of the pie, in a direction bisecting the segment.

By default, this function chooses a vector character set and each label is drawn to fit a rectangle fWid wide and fHt high.

The color of the labels may either correspond to those of their respective pie slices or be uniformly the same.

The LPSMART mode enables auto-arranging of labels to avoid overlapping. The algorithm is not infallible and may cause labels to extend outside the visible area of the graph if too many are concentrated in a small sector of the pie.

J

Topic

GSLabelnPie

Related

GSLabelPie
GSLoadRFont
GSLoadVFont
GSPie2D
GSPie3D

GSLabelnX function

Draws numeric labels along X axis

C/C++

```
int GSLabelnX( double fxOrg, double fyOrg, double fInc,
               double fWid, double fHt,
               double fBaseVal, double fStepVal,
               int nPrec, int nLabs, int nCSet,
               int nTMode, int nClr )
```

FoxPro

```
r = GSLabelnX(fxOrg, fyOrg, fInc, fWid, fHt, fBaseVal,
              fStepVal, nPrec, nLabs, nCSet, nTMode,
              nClr)
```

Visual Basic

```
r% = GSLabelnX(fxOrg#, fyOrg#, fInc#, fWid#, fHt#,
               fBaseVal#, fStepVal#, nPrec%, nLabs%,
               nCSet%, nTMode%, nClr%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	X increment
`fWid`	Width of label
`fHt`	Height of label
`fBaseVal`	Starting value
`fStepVal`	Incrementing value
`nPrec`	Decimal precision
`nLabs`	Number of labels
`nCSet`	Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.
`nTMode`	Text mode (see Text mode constants)
`nClr`	Color of labels (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSLabelnX function draws a horizontal sequence of text labels on screen starting at fxOrg, fyOrg and at intervals of fInc to the right of this point. The labels start with the value fBaseVal and increase in steps of fStepVal. nPrec determines the number of decimals shown.

By default, this function chooses a vector character set and each

label is drawn to fit a rectangle fWid wide and fHt high.

J

Topic
GSLabelnX

Related
GSLabelX
GSLoadRFont
GSLoadVFont

GSLabelnY function

Draws numeric labels along Y axis

C/C++

```
int GSLabelnY( double fxOrg, double fyOrg, double fInc,
               double fWid, double fHt,
               double fBaseVal, double fStepVal,
               int nPrec, int nLabs, int nCSet,
               int nTMode, int nClr )
```

FoxPro

```
r = GSLabelnY(fxOrg, fyOrg, fInc, fWid, fHt, fBaseVal,
              fStepVal, nPrec, nLabs, nCSet, nTMode,
              nClr)
```

Visual Basic

```
r% = GSLabelnY(fxOrg#, fyOrg#, fInc#, fWid#, fHt#,
               fBaseVal#, fStepVal#, nPrec%, nLabs%,
               nCSet%, nTMode%, nClr%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	X increment
`fWid`	Width of label
`fHt`	Height of label
`fBaseVal`	Starting value
`fStepVal`	Incrementing value
`nPrec`	Decimal precision
`nLabs`	Number of labels
`nCSet`	Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.
`nTMode`	Text mode (see Text mode constants)
`nClr`	Color of labels (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSLabelnY function draws a vertical sequence of text labels on screen starting at (fxOrg,fyOrg) and at intervals of fInc above this point. The labels start with the value fBaseVal and increase in steps of fStepVal. nPrec determines the number of decimals shown.

By default, this function chooses a vector character set and each

label is drawn to fit a rectangle fWid wide and fHt high.

J

Topic
GSLabelnY

Related
GSLabelY
GSLoadRFont
GSLoadVFont

GSLabelPie function

Draws pie chart text labels

C/C++

```
int GSLabelPie( double fxOff, double fRad, double fWid,
                double fHt, int nLabs, int nMode,
                int nCSet, int nTMode, int nClr,
                char* szLabs[] )
```

FoxPro

```
r = GSLabelPie(fxOff, fRad, fWid, fHt, nLabs, nMode,
               nCSet, nTMode, nClr, @szLabs(1))
```

Visual Basic

Use VBGSLabelPie function

Parameters

`fxOff` Horizontal offset

`fRad` Radius of the arc on which the labels are drawn. This radius must be at least 1.1 times greater than the pie radius (or 1.35 times greater if any segments are exploded). The pie radius is taken from the preceding GSPieChart function call.

`fWid` Width of label

`fHt` Height of label

`nLabs` Number of labels in array

`nMode`

Constant	Value	Meaning
LPSEGCLR	1	Colors labels the same as segments
LPNOLINES	2	Omits pointing lines from pie to labels
LPSMART	16	Auto-arranges labels to avoid overlapping

`nCSet` Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.

`nTMode` Text mode (see Text mode constants)

`nClr` Color of labels (see Color constants)

`szLabs` Pointer to array of text labels

Return values

0 Success

-1 Failure

Description

The GSLabelPie function draws a sequence of text labels to complement a pie chart. The pie chart must be drawn first because this function adopts certain parameters from the

preceding pie chart function call.

There must be the same number of labels as pie slices. The angular position of each label is calculated from the data for the pie chart.

The labels are drawn in an arc on each side of the pie, connected to their respective segments by pointing lines. These lines are drawn from the label horizontally a distance fXOff, then radially toward the center of the pie in a direction bisecting the segment.

By default, this function chooses a vector character set and each label is drawn to fit a rectangle fWid wide and fHt high. Since vector text is shaped to fit this area, the strings should all be of the same length to give a uniform character appearance. This may require padding of some strings with spaces.

The color of the labels may either correspond to those of their respective pie slices or be uniformly the same.

The LPSMART mode enables auto-arranging of labels to avoid overlapping. The algorithm is not infallible, and may cause labels to extend outside the visible area of the graph if too many are concentrated in a small sector of the pie.

J

Topic

GSLabelPie

Related

GSLabelnPie
GSLoadRFont
GSLoadVFont
VBGSLabelPie
GSPie2D
GSPie3D

GSLabelX function

Draws text labels along X axis

C/C++

```
int GSLabelX( double fxOrg, double fyOrg, double fInc,
              double fWid, double fHt, int nLabs,
              int nCSet, int nTMode, int nClr,
              char* szLabs )
```

FoxPro

```
r = GSLabelX(fxOrg, fyOrg, fInc, fWid, fHt, nLabs,
             nCSet, nTMode, nClr, @szLabs(1))
```

Visual Basic

Use VBGSLabelX function

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	X increment
`fWid`	Width of label
`fHt`	Height of label
`nLabs`	Number of labels
`nCSet`	Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.
`nTMode`	Text mode (see Text mode constants)
`nClr`	Color of labels (see Color constants)
`szLabs`	Pointer to array of text labels

Return values

0	Success
-1	Failure

Description

The GSLabelX function draws a horizontal sequence of text labels starting on screen at (fxOrg,fyOrg) and at intervals of fInc to the right of this point. The labels must be passed in a string array of dimension nLabs.

By default, this function chooses a vector character set and each label is drawn to fit a rectangle fWid wide and fHt high. Since vector text is shaped to fit this area, the strings should all be of the same length to give a uniform character appearance. This may require padding of some strings with spaces.

J

Topic

GSLabelX

Related

GSLabelnX
GSLoadRFont
GSLoadVFont
GSXYGraph
VBGSLabelX

GSLabelY function

Draws text labels along Y axis

C/C++

```
int GSLabelY( double fxOrg, double fyOrg, double fInc,
              double fWid, double fHt, int nLabs,
              int nCSet, int nMode, int nClr,
              char* szLabs[] )
```

FoxPro

```
r = GSLabelY(fxOrg, fyOrg, fInc, fWid, fHt, nLabs,
             nCSet, nMode, nClr, @szLabs(1))
```

Visual Basic

Use VBGSLabelY function

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	Y increment
`fWid`	Width of label
`fHt`	Height of label
`nLabs`	Number of labels
`nCSet`	Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.
`nTMode`	Text mode (see Text mode constants)
`nClr`	Color of labels (see Color constants)
`szLabs`	Pointer to array of text labels

Return values

0	Success
-1	Failure

Description

The GSLabelY function draws a vertical sequence of text labels starting on screen at (fxOrg,fyOrg) and at intervals of fInc above this point. The labels must be passed in a string array of dimension nLabs.

By default, this function chooses a vector character set and each label is drawn to fit a rectangle fWid wide and fHt high. Since vector text is shaped to fit this area, the strings should all be of the same length to give a uniform character appearance. This may require padding of some strings with spaces.

J

Topic

GSLabelY

Related

GSLabelnY
GSLoadRFont
GSLoadVFont
GSXYGraph
VBGSLabelY

GSLegend function

Draws legend

C/C++

```
int GSLegend( double fxOrg, double fyOrg, double fWid,
              double fHt, int nNLeg, int nRows,
              int nMode, int nCSet, int nTMode,
              int nClr, int* nBClr, int* nBPatt,
              char* szLegs[] )
```

FoxPro

```
r = GSLegend(fxOrg, fyOrg, fWid, fHt, nNLeg, nRows,
             nMode, nCSet, nTMode, nClr, @nBClr(1),
             @nBPatt(1), @szLegs(1))
```

Visual Basic

Use VBGSLegend function

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fWid` Width of bounding area

`fHt` Height of bounding area

`nNLeg` Number of legend entries

`nRows` Number of rows in legend

`nMode`

Constant	Value	Meaning
LGBOX	1	Draws black box around legend area
LGTXCLR	2	Text takes its color from the associated legend box
LGBG	4	Fills the bounding area with the current background color
LGLINE	8	Shows line patterns
LGSYMBOL	16	Shows symbols

`nCSet` Character set (see Character set constants)

`nTMode` Text mode (see Text mode constants)

`nClr` Text color (see Color constants)

`nBClr` Pointer to array of legend box colors (see Color constants)

`nBPatt` Pointer to array of legend box patterns (see Pattern constants)

`szLegs` Pointer to array of text labels

Return values

0	Success
-1	Failure

Description

The GSLegend function draws a legend to accompany a graph or chart.

The legend, consisting of a stack or row of patterned and colored boxes associated with text strings, is drawn within a bounding rectangle defined by its width and height and located by the origin at its bottom left. If you choose, you can have the legend show line patterns rather than fill patterns (use nMode LGLINE).

Each legend entry is defined by elements in three arrays: color, pattern, and text.

J

Topic

GSLegend

Related

GSLoadRFont

GSLoadVFont

VBGSLegend

GSLineAbs function

Draws line using absolute coordinates

C/C++

```
int GSLineAbs( double fxA, double fyA, double fxB,
               double fyB, int nMode, int nStyle,
               int nClr )
```

FoxPro

```
r = GSLineAbs(fxA, fyA, fxB, fyB, nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSLineAbs(fxA#, fyA#, fxB#, fyB#, nMode%, nStyle%,
               nClr%)
```

Parameters

`fxA` X start

`fyA` Y start

`fxB` X end

`fyB` Y end

`nMode`

Constant	Value	Meaning
LAUPDATE	1	Updates the current position
LATHICK	4	Draws a thick line
LACONT	16	Continues from current position

`nStyle` Line pattern or thickness (see Line style constants)

`nClr` Color of line (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSLineAbs function draws a straight line from point (fxA,fyA) to point (fxB,fyB).

Optionally, you can have the current position be updated to the end point of the line you draw (fxB,fyB). To do this, include LACONT in the nMode parameter.

J

Topic

GSLineAbs

Related

GSArrow

GSBox2D

GSLineRel

GSPolyFill
GSPolyVec

GSLineFit function

Fits straight line to data

C/C++

```
int GSLineFit( int nStyle, int nClr )
```

FoxPro

```
r = GSLineFit(nStyle, nClr)
```

Visual Basic

```
r% = GSLineFit(nStyle%, nClr%)
```

Parameters

`nStyle`	Line style (see Line style constants)
`nClr`	Line color (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSLineFit function fits a straight line through graphed data using a least-squares linear regression of Y over X.

The line is clipped within a window defined by the GSStatsWin function.

J

Topic

GSLineFit

Related

GSStatsWin
GSStatsArr
GSCurveFit
GSGetCC

GSLineRel function

Draws line using relative coordinates

C/C++

```
int GSLineRel( double fxr, double fya, int nMode,
               int nStyle, int nClr )
```

FoxPro

```
r = GSLineRel(fxr, fya, nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSLineRel(fxr#, fya#, nMode%, nStyle%, nClr%)
```

Parameters

`fxr` X relative or radius

`fya` Y relative or angle

`nMode`

Constant	Value	Meaning
LRNOUPDATE	1	Doesn't update the current position
LRPOLAR	2	fxr, fya are polar radius and angle
LRTHICK	4	Draws a thick line

`nStyle` Line pattern or thickness (see Line style constants)

`nClr` Line color (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSLineRel function draws a straight line from the current position to an end point specified relative to the start.

The relative position may be specified either in Cartesian (X,Y) coordinates or in polar (radius and angle) notation.

By default the current position is updated to the end point of the line. This may be overridden.

J

Topic

GSLineRel

Related

GSFixPos
GSMovePos
GSGetCurX
GSGetCurY
GSLineAbs

GSLinLog function

Draws lin/log graph

C/C++

```
int GSLinLog( double fxOrg, double fyOrg,
              double fCycleWid, double fBaseX,
              int nMode, int nClr )
```

FoxPro

```
r = GSLinLog(fxOrg, fyOrg, fCycleWid, fBaseX, nMode,
             nClr)
```

Visual Basic

```
r% = GSLinLog(fxOrg#, fyOrg#, fCycleWid, fBase#,
              nMode%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fCycleWid` Width of one cycle (log base 10)

`fBaseX` Horizontal base of the graph at x = 0

`nMode`

Constant	Value	Meaning
XYGLINE	1	Connects points with lines
XYGSYMBOL	2	Draws symbols at points
XYGSTICK	4	Draws vertical sticks to points
XYGTHICK	16	Uses thick lines
XYGPATT	32	Uses patterned lines
XYGGROUPED	64	Multiple data-set mode

`nClr` Color of markers if you're graphing only one data set. With multiple data sets, you have to create a color array. (See Color constants.)

Return values

0 Success

-1 Failure

Description

The GSLinLog function draws a lin/log graph.

The Y position of each point is the value in the fA array. The X position of each point is the log of the value in the fD array.

The graph may appear as lines, symbols, vertical sticks, or any combination of these.

In single data-set mode, the values in the appropriate arrays are applied on a per-point basis. This means that a single data set line graph can use a symbol or line style to differentiate each

point. In multiple data-set mode, the arrays are applied on a per-set basis to enable you to differentiate the data sets on the graph.

Lines may be specified as patterned or thickened by means of the nMode parameter. In patterned mode the nPatt array is presumed to contain a series of line style values specified from the set LSSOLID, LSDOT, and so forth. In thickened mode, the values are presumed to specify the approximate thicknesses of the lines in pixel units.

Specifying missing-data points

You can use the nAux array--through the GSDataAux function--to flag points of a lin/log graph as "missing." Missing points aren't shown, whether or not you've provided values for them. If you use lines to connect points (nMode XYGLINE), the connecting lines are omitted both to and from each missing point.

nAux setting	**Value**	**Meaning**
	0	Point shown normally
	256	Point is "missing" and not drawn

The size of the nAux array may be nPts or nPts nGroup. If you set the size to nPts and there's more than one group of data, the same missing points are assumed for all the groups. If the size is nPts nGroup, each point in each group has its own missing-data flag.

GSDataTrans parameters for lin/log graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions of plotted points)
fD[nPts]	Pointer to distance array (X positions of plotted points)
nPatt[nPts]	Pointer to array containing line style or thickness for each line element--used only with nMode XYGTHICK or XYGPATT
nSymbol[nPts]	Pointer to array containing symbol design for each point
nAux[nPts]	Pointer to array containing missing-data flag for each point
nClr[0]	Not used

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions of plotted points)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions of plotted points)
nPatt[nGroup]	Pointer to array containing line style or thickness for each data set--used only with nMode XYGTHICK or XYGPATT
nSymbol[nGroup]	Pointer to array containing symbol design for each data set
nAux[nPts] *or* nAux[nPts][nGroup]*	Pointer to array containing missing-data flag for each point
nClr[nGroup]	Pointer to array containing color for each data set

* GSDataTrans can't pass two-dimensional fD or nAux arrays. You have to use the GSDataDist or GSDataAux function if you want to specify individual fD or nAux values for each data set. However, you can use GSDataTrans if you want to apply the same fD or nAux values to points in all sets.

J

Topic

GSLinLog

Related

GSLogLog
GSLogLin
GSXYGraph

Log functions:

GSGetALog10
GSGetLog10
GSGetALogE
GSGetLogE

Axis/grid/legend:

GSLogAxis
GSLogGrid
GSLegend

Labels:

GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:

GSDataTrans
GSDataAux
GSDataDist

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

GSLoadRFont function

Loads raster font

C/C++

```
int GSLoadRFont( int nFamily, int nAttrib, int nSize,
                 int nPitch )
```

FoxPro

```
r = GSLoadRFont(nFamily, nAttrib, nSize, nPitch)
```

Visual Basic

```
r% = GSLoadRFont(nFamily%, nAttrib%, nSize%, nPitch%)
```

Parameters

`nFamily`

Constant	Value	Meaning
FOROMAN	1	Roman
FOSWISS	2	Swiss (Helvetica)
FOMODERN	3	Modern
FOSCRIPT	4	Script
FODECO	5	Decorative

`nAttrib`

Constant	Value	Meaning
FOITALIC	16	Italic
FOBOLD	32	Bold
FOULINE	64	Underlined

`nSize`

Size as a percentage of the size of the system font. For example, an nSize of 400 selects a font four times the size of the system font.

`nPitch`

Constant	Value	Meaning
FOFIXED	128	Fixed pitch (default is variable)

Return values

0	Success
-1	Failure

Description

The GSLoadRFont function loads a new raster character set into the user buffer. That set is used for any text whose character set is specified as CSRASTER + CSUSER.

Windows itself selects the character set that best matches the specification. GSLoadRFont has no control over the matching process.

Topic

GSLoadRFont

Related

GSRText
GSLabelnPie
GSLabelnX
GSLabelnY
GSLabelPie
GSLabelX
GSLabelY
GSLegend
GSSetRFontFace

GSLoadVFont function

Loads vector font

C/C++

```
int GSLoadVFont( int nFamily, int nAttrib, int nPitch )
```

FoxPro

```
r = GSLoadVFont(nFamily, nAttrib, nPitch)
```

Visual Basic

```
r% = GSLoadVFont(nFamily%, nAttrib%, nPitch%)
```

Parameters

`nFamily`

Constant	Value	Meaning
FOROMAN	1	Roman
FOSWISS	2	Swiss (Helvetica)
FOMODERN	3	Modern
FOSCRIPT	4	Script
FODECO	5	Decorative

`nAttrib`

Constant	Value	Meaning
FOITALIC	16	Italic
FOBOLD	32	Bold
FOULINE	64	Underlined

`nPitch`

Constant	Value	Meaning
FOFIXED	128	Fixed pitch (default is variable)

Description

The GSLoadVFont function loads a new vector character set into the User buffer. It will be used in any text function that specifies CSUSER as its character set.

Windows itself selects the character set that best matches the specification. This function has no control over the matching process.

Vector fonts are infinitely scalable. The size of the printed text is determined by a parameter in the function that draws it, not when the font is loaded.

J

Topic

GSLoadVFont

Related

GSVText

GSLabelnPie

GSLabelnX
GSLabelnY
GSLabelPie
GSLabelX
GSLabelY
GSLegend
GSSetVFontFace

GSLogAxis function

Draws logarithmic axis

C/C++

```
int GSLogAxis( double fxOrg, double fyOrg, double fLen,
               double fTickLen, int nCycles, int nMode,
               int nStyle, int nClr )
```

FoxPro

```
r = GSLogAxis(fxOrg, fyOrg, fLen, fTickLen, nCycles,
              nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSLogAxis(fxOrg#, fyOrg#, fLen#, fTickLen#,
               nCycles%, nMode%, nStyle%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fLen` Length of axis

`fTickLen` Length of ticks

`nCycles` Number of log cycles

`nMode`

Constant	Value	Meaning
AXTICKOUT	0	Ticks on outside (left on Y axis, bottom on X axis)
AXTICKIN	1	Ticks strike through the axis
AXTICKTHRU	2	Ticks on inside (right on Y axis, top on X axis)
AXISX	0	Draws in X direction
AXISY	4	Draws in Y direction

`nStyle` Line style (see Line style constants)

`nClr` Color of axis (see Color constants)

Return values

0	Success
-1	Failure

Description The GSLogAxis function draws an axis in the X or Y direction with ticks at logarithmic distances, 9 per cycle.

J

Topic

GSLogAxis

Related

GSAxis
GSLogGrid
GSGetALog10
GSGetLog10
GSGetALogE
GSGetLogE

GSLogGrid function

Draws logarithmic grid

C/C++

```
int GSLogGrid( double fxOrg, double fyOrg,
               double fAxisLen, double fGridLen,
               int nCycles, int nMode, int nStyle,
               int nClr )
```

FoxPro

```
r = GSLogGrid(fxOrg, fyOrg, fAxisLen, fGridLen,
              nCycles, nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSLogGrid(fxOrg#, fyOrg#, fAxisLen#, fGridLen#,
               nCycles%, nMode%, nStyle%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fAxisLen` Length of axis

`fGridLen` Length of grid

`nCycles` Number of log cycles

`nMode`

Constant	Value	Meaning
GRX	0	Draws X grids (parallel to Y axis)
GRY	1	Draws Y grids (parallel to X axis)
GRNOFIRST	2	Omits the first grid line
GRNOLAST	3	Omits the last grid line

`nStyle` Line style (see Line style constants)

`nClr` Color of grid lines (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSLogGrid function draws grid lines at logarithmic intervals (9 grids per cycle) along the X or Y axes, parallel to the Y or X axis.

You can omit the first and last grid lines to avoid overdrawing axes or frames.

J

Topic

GSLogGrid

Related

GSAxis
GSGrid
GSLogAxis
GSGetALog10
GSGetLog10
GSGetALogE
GSGetLogE

GSLogLin function

Draws log/lin graph

C/C++

```
int GSLogLin( double fxOrg, double fyOrg, double fInc,
              double fCycleHt, double fBaseVal,
              int nMode, int nClr )
```

FoxPro

```
r = GSLogLin(fxOrg, fyOrg, fInc, fCycleHt, fBaseVal,
             nMode, nClr)
```

Visual Basic

```
r% = GSLogLin(fxOrg#, fyOrg#, fInc#, fCycleHt#,
              fBaseVal#, nMode%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fInc` X or Y increment

`fCycleHt` Height of one cycle (log base 10)

`fBaseVal` Base value of the graph at y = 0

`nMode`

Constant	Value	Meaning
XYGLINE	1	Connects points with lines
XYGSYMBOL	2	Draws symbols at points
XYGSTICK	4	Draws vertical sticks to points
XYGVARX	8	Uses fD for X position
XYGTHICK	16	Uses thick lines
XYGPATT	32	Uses patterned lines
XYGGROUPED	64	Multiple data-set mode

`nClr` Color of markers if you're graphing only one data set. With multiple data sets, you have to create a color array. (See Color constants.)

Return values

0 Success

-1 Failure

Description

The GSLogLin function draws a log/lin graph.

The graph may appear as lines, symbols, vertical sticks, or any combination of these.

In single data-set mode, the values in the appropriate arrays are applied on a per-point basis. This means that a single data set

line graph can use a symbol or line style to differentiate each point. In multiple data-set mode, the arrays are applied on a per-set basis to enable you to differentiate the data sets on the graph.

Lines may be specified as patterned or thickened by means of the nMode parameter. In patterned mode, the nPatt array is presumed to contain a series of line style values specified from the set LSSOLID, LSDOT, and so forth. In thickened mode, the values are presumed to specify the approximate thicknesses of the lines in pixel units.

The data may be graphed either at fixed increments in X, as defined by fInc, or using the individual X values passed in the fD array.

Specifying missing-data points

You can use the nAux array--through the GSDataAux function--to flag points of a log/lin graph as "missing." Missing points aren't shown, whether or not you've provided values for them. If you use lines to connect points (nMode XYGLINE), the connecting lines are omitted both to and from each missing point.

nAux setting	**Value**	**Meaning**
	0	Point shown normally
	256	Point is "missing" and not drawn

The size of the nAux array may be nPts or nPts nGroup. If you set the size to nPts and there's more than one group of data, the same missing points are assumed for all the groups. If the size is nPts nGroup, each point in each group has its own missing-data flag.

GSDataTrans parameters for log/lin graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions of plotted points)
fD[nPts]	Pointer to distance array (X positions of plotted points)--used only with nMode XYGVARX
nPatt[nPts]	Pointer to array containing line style or thickness for each line element--used only with nMode XYGTHICK or XYGPATT
nSymbol[nPts]	Pointer to array containing symbol design for each point
nAux[nPts]	Pointer to array containing missing-data flag

	for each point
nClr[0]	Not used
Multiple data sets	
nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions of plotted points)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions of plotted points)--used only with nMode XYGVARX
nPatt[nGroup]	Pointer to array containing line style or thickness for each data set--used only with nMode XYGTHICK or XYGPATT
nSymbol[nGroup]	Pointer to array containing symbol design for each data set
nAux[nPts] *or* nAux[nPts][nGroup]*	Pointer to array containing missing-data flag for each point
nClr[nGroup]	Pointer to array containing color for each data set

* GSDataTrans can't pass two-dimensional fD or nAux arrays. You have to use the GSDataDist or GSDataAux function if you want to specify individual fD or nAux values for each data set. However, you can use GSDataTrans if you want to apply the same fD or nAux values to points in all sets.

J

Topic
GSLogLin

Related
GSLogLog
GSLinLog
GSXYGraph

Log functions:
GSGetALog10
GSGetLog10
GSGetALogE
GSGetLogE

Axis/grid/legend:
GSLogAxis
GSLogGrid

GSLegend

Labels:
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:
GSDataTrans
GSDataAux
GSDataDist

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSLogLog function

Draws log/log graph

C/C++

```
int GSLogLog( double fxOrg, double fyOrg,
              double fCycleHt, double fBaseY,
              double fCycleWid, double fBaseX,
              int nMode, int nClr )
```

FoxPro

```
r = GSLogLog(fxOrg, fyOrg, fCycleHt, fBaseY, fCycleWid,
             fBaseX, nMode, nClr)
```

Visual Basic

```
r% = GSLogLog(fxOrg#, fyOrg#, fCycleHt#, fBaseY#,
              fCycleWid#, fBaseX#, nMode%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fCycleHt` Height of one cycle (log base 10)

`fBaseY` Vertical base value of the graph at y = 0

`fCycleWid` Width of one cycle (log base 10)

`fBaseX` Horizontal base value of the graph at x = 0

`nMode`

Constant	Value	Meaning
XYGLINE	1	Connects points with lines
XYGSYMBOL	2	Draws symbols at points
XYGSTICK	4	Draws vertical sticks to points
XYGTHICK	16	Uses thick lines
XYGPATT	32	Uses patterned lines
XYGGROUPED	64	Multiple data-set mode

`nClr` Color of markers if you're graphing only one data set. With multiple data sets, you have to create a color array. (See Color constants.)

Return values

0 Success

-1 Failure

Description

The GSLogLog function draws a log/log graph.

The Y position of each point is the log of the value in the fA array. The X position of each point is the log of the value in the fD array.

The graph may appear as lines, symbols, vertical sticks, or any

combination of these.

In single data-set mode, the values in the appropriate arrays are applied on a per-point basis. This means that a single data set line graph can use a symbol or line style to differentiate each point. In multiple data-set mode, the arrays are applied on a per-set basis to enable you to differentiate the data sets on the graph.

Lines may be specified as patterned or thickened by means of the nMode parameter. In patterned mode, the nPatt array is presumed to contain a series of line style values specified from the set LSSOLID, LSDOT, and so forth. In thickened mode, the values are presumed to specify the approximate thicknesses of the lines in pixel units.

The data may be graphed either at fixed increments in X, as defined by fInc, or using the individual X values passed in the fD array.

Specifying missing-data points

You can use the nAux array--through the GSDataAux function--to flag points of a log/log graph as "missing." Missing points aren't shown, whether or not you've provided values for them. If you use lines to connect points (nMode XYGLINE), the connecting lines are omitted both to and from each missing point.

nAux setting	**Value**	**Meaning**
	0	Point shown normally
	256	Point is "missing" and not drawn

The size of the nAux array may be nPts or nPts nGroup. If you set the size to nPts and there's more than one group of data, the same missing points are assumed for all the groups. If the size is nPts nGroup, each point in each group has its own missing-data flag.

GSDataTrans parameters for log/log graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions of points)
fD[nPts]	Pointer to distance array (X positions of points)
nPatt[nPts]	Pointer to array containing line style or thickness for each line element--used only with nMode XYGTHICK or XYGPATT
nSymbol[nPts]	Pointer to array containing symbol design

	for each point
nAux[nPts]	Pointer to array containing missing-data flag for each point
nClr[0]	Not used

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions of points)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions of points)
nPatt[nGroup]	Pointer to array containing line style or thickness for each data set--used only with nMode XYGTHICK or XYGPATT
nSymbol[nGroup]	Pointer to array containing symbol design for each data set
nAux[nPts] *or* nAux[nPts][nGroup]*	Pointer to array containing missing-data flag for each point
nClr[nGroup]	Pointer to array containing color for each data set

* GSDataTrans can't pass two-dimensional fD or nAux arrays. You have to use the GSDataDist or GSDataAux function if you want to specify individual fD or nAux values for each data set. However, you can use GSDataTrans if you want to apply the same fD or nAux values to points in all sets.

J

Topic

GSLogLog

Related

GSLinLog
GSLogLin
GSXYGraph

Log functions:

GSGetALog10
GSGetLog10
GSGetALogE
GSGetLogE

Axis/grid/legend:

GSLogAxis

GSLogGrid
GSLegend

Labels:
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:
GSDataTrans
GSDataAux
GSDataDist

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSMClrRgn function

Clears mouse hot region

C/C++

```
int GSMClrRgn( int nRgn )
```

FoxPro

```
r = GSMClrRgn(nRgn)
```

Visual Basic

```
r% = GSMClrRgn(nRgn%)
```

Parameter

`nRgn`	Hot region number

Return values

0	Success
-1	Failure

Description

The GSMClrRgn function clears the specified mouse hot region, previously defined in the current window.

The function doesn't clear any visible representation of the hot region produced by GSPolyFill.

J

Topic

GSMClrRgn

Related

GSMSetRgn
GSPolyFill
GSMNotify
GSMStatus

GSMean function

Draws mean of data set

C/C++

```
int GSMean( int nStyle, int nClr )
```

FoxPro

```
r = GSMean(nStyle, nClr)
```

Visual Basic

```
r% = GSMean(nStyle%, nClr%)
```

Parameters

`nStyle`	Line style (see Line style constants)
`nClr`	Line color (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSMean function draws a line representing the mean of a data set. The mean is drawn relative to the origin of the immediately preceding graphing function.

The line is clipped within a window defined by the GSStatsWin function.

J

Topic

GSMean

Related

GSSD
GSGetMean
GSStatsWin
GSStatsArr
GSMinMax

GSMGetX function

Gets mouse X position

C/C++

```
double GSMGetX( )
```

FoxPro

```
r = GSMGetX()
```

Visual Basic

```
r# = GSMGetX()
```

Return values

0 or greater	X position of the mouse pointer in view units
-1	Failure

Description

The GSMGetX function returns the X position of the mouse pointer as frozen by a preceding GSMStatus call.

The mouse position is returned in the view units of the current view. The option of overlapping views means that a single position of the mouse pointer may relate to a different logical position in each view. To obtain the mouse position in the units of another view, make a preceding GSUseView call to select the view. This procedure may be repeated to obtain the position in as many views as required.

A negative position indicates that the mouse pointer is outside the logical bounds of the current view.

J

Topic

GSMGetX

Related

GSMGetY
GSMStatus
GSUseView
GSMNotify

GSMGetY function

Gets mouse Y position

C/C++

```
double GSMGetY( )
```

FoxPro

```
r = GSMGetY()
```

Visual Basic

```
r# = GSMGetY()
```

Return values

0 or greater	Y position of the mouse pointer in view units
-1	Failure

Description

TheGSMGetY function returns the Y position of the mouse pointer as frozen by a preceding GSMStatus call.

The mouse position is returned in the view units of the current view. The option of overlapping views means that a single position of the mouse pointer may relate to a different logical position in each view. To obtain the mouse position in the units of another view, make a preceding GSUseView call to select the view. This procedure may be repeated to obtain the position in as many views as required.

A negative position indicates that the mouse pointer is outside the logical bounds of the current view.

J

Topic

GSMGetY

Related

GSMGetX
GSMStatus
GSUseView
GSMNotify

GSMinMax function

Draws minimum and maximum of data set

C/C++

```
int GSMinMax( int nStyle, int nClr )
```

FoxPro

```
r = GSMinMax(nStyle, nClr)
```

Visual Basic

```
r% = GSMinMax(nStyle%, nClr%)
```

Parameters

`nStyle`	Line style (see Line style constants)
`nClr`	Line color (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSMinMax function draws lines representing the minimum and maximum of a data set. These lines are calculated relative to the origin of the immediately preceding graphing function.

The lines are clipped within a window defined by the GSStatsWin function.

J

Topic

GSMinMax

Related

GSMean
GSSD
GSStatsWin

GSMissingLineStyle function

Selects options for bridging gaps caused by missing data

C/C++

```
int GSMissingLineStyle(int nMode, int nSize
                       [,int *nPatt, int *nClr])
```

FoxPro

```
r = GSMissingLineStyle(nMode, nSize [, @nPatt(1),
                       @nClr(1)])
```

Visual Basic

```
r% = GSMissingLineStyle(nMode%, nSize% [, nPatt%(0),
                        nClr%(0)])
```

Parameters

`nMode`

Constant	Value	Meaning
MLSOMIT	0	No bridging lines (default)
MLSSAMESTYLE	1	Bridge with line style of the graph
MLSPATTERNED	2	Bridge with patterned lines
MLSTHICK	3	Bridge with thick lines

`nSize` If nMode is 2 or 3, size of nPatt and nClr arrays. Otherwise, zero.

`nPatt` Line pattern array. Required when nMode is 2 or 3.

`nClr` Line color array. Required when nMode is 2 or 3.

Return values

0	Success
-1	Failure

Description

If you have incomplete sets of data, or sets in which the values of certain data points are unknown, you can flag points as missing by calling the GSDataAux function. When the graph type is line, or any log variant of line, missing points cause a gap in the line. The GSMissingLineStyle function sets options for bridging gaps left by missing points.

nMode = 0 No bridging lines. This is the default, and it is what you get if you do not call GSMissingLineStyle(). However, if you have called the function and then later want to show gaps, this option will turn off bridging lines.

Set nSize to 0. Arrays for nPatt and nClr are ignored and can be omitted from the call.

nMode = 1 Bridge gaps by continuing the data line in the same style and color.

Set nSize to 0. Arrays for nPatt and nClr are ignored and can be omitted from the call.

nMode = 2 Bridge gaps with a line in a pattern and color of your choice. You can use the same pattern and color for all data sets, or you can select a different pattern and color for each data set.

To use the same pattern and color for all data sets, set nSize to 1.

Dimension the nPatt array for one element and store the pattern number in it. Dimension nClr for one element and store the line color.

To use different patterns for each data set, nSize must be equal to the number of data groups for the primary graph plus the number of groups for overlay graphs. Dimension nPatt and nClr to nSize elements. Store a pattern number for each group in nPatt and a line color in nClr.

Six line patterns are available. See line style constants for a list.

Colors are specified as color index numbers. See color constants for a list. If a color value of -1 is passed, the bridging line is drawn in the color of the graph line.

nMode = 3 Bridge gaps with a line in a thickness and color of your choice. You can use the same thickness and color for all data sets, or you can select a different thickness and color for each data set.

The procedure is the same as for nMode 2. Line thickness is specified in pixels. Values can range from 1 to 5.

J

Topic

GSMissingLineStyle

Related

GSDataAux

GSDataTrans

GSMMotion function

Reads mouse motion indicator

C/C++

```
int GSMMotion( )
```

FoxPro

```
r = GSMMotion()
```

Visual Basic

```
r% = GSMMotion()
```

Return values

0	False--no mouse movement since last call
nonzero	True--mouse has moved since last call

Description

The GSMMotion function returns a non-zero value if the mouse has moved within the client area of the current window since the last time it was called.

Movement of the mouse outside the client area of a window can't be detected.

GSMStatus must be called before reading back actual position information.

You shouldn't call GSMMotion in a loop waiting for the mouse to move. Calling GSMNotify with mode MNTRACK provides the means of monitoring mouse movement asynchronously through the application message queue. GSMStatus must continue to be called to freeze mouse position information but only in response to mouse MNTRACK event messages.

J

Topic

GSMMotion

Related

GSMStatus

GSMNotify

GSMNotify function

Enables and disables notification of mouse events

C/C++

```
int GSMNotify( HWND hWnd, int nMsg, int nMode )
```

FoxPro

```
r = GSMNotify(hWnd, nMsg, nMode)
```

Visual Basic

```
r% = GSMNotify(hWnd%, nMsg%, nMode%)
```

Parameters

`hWnd` Client notification window handle (in FoxPro, always a null string)

`nMsg` Client window message number (in FoxPro, always a null string)

`nMode`

Constant	Value	Meaning
MNTRACK	1	Mouse cursor movement
MNPRESS	2	Button press and release
MNHIT	4	Button press in predefined region
MNHITPT	8	Button press on point in hot graph

Return values

0	False--no mouse present
nonzero	True--a mouse is present

Description

The GSMNotify function enables and disables asynchronous notification of mouse events for a particular window. Once you call this function (assuming a mouse is present), the application is ready to receive mouse event messages in the window procedure for that window.

The first two parameters specify the Windows handle of the window where mouse event messages should be posted and the unique window-message number. The message number should be identified within the window procedure for the notified window.

The nMode settings are additive--for example, MNTRACK | MNPRESS notifies the window of movement as well as button actions. Each call to GSMNotify for a particular window replaces the modes previously in effect. To end event notification completely, call GSMNotify with all three parameters set to zero.

Each graphing window can have different event notifications in force. GSMNotify always acts on the current window. To change the current window, use the GSUseView function.

A mouse event message is posted in the following form:

Parameter	*Type*	*Meaning*
hWnd	HWND	The Windows handle of the notified window as

		originally specified to GSMNotify
uMsg	UINT	The Windows message number as originally specified to GSMNotify
wParam	UINT	The type of window event, which is *one* of the nEvents settings originally specified to GSMNotify. Note that although events may be enabled together, they are always notified individually.
lParam	LONG	Event qualifying information

Constant	Value	Meaning
MNTRACK	1	No information is supplied.
MNPRESS	2	The low-order word contains a numeric value indicating which mouse buttons, if any, are pressed. Values are the same as for GSMStatus and may be combined. Zero indicates that all the buttons are released. The high-order word isn't used.
MNHIT	4	The low-order word contains a numeric value indicating which mouse button was pressed in the region. Values are the same as for GSMStatus but aren't combined. The high-order word contains a number identifying the region in which the button was pressed. The same region number will have been returned in an earlier GSMSetRgn call.
MNHITPT	8	The low-order word contains the zero-based point index number of the point clicked on the graph. The high-order word contains the zero-based set index number.

Example

The following example illustrates how to open a graphing window and enable it for mouse event notification:

```
#define WM_MOUSEEVENT    (WM_USER + 1)

double ScreenWid, ScreenHt;
```

```
int WinNum;
ScreenWid = GSGetSXExt();
ScreenHt = GSGetSYExt();
WinNum = GSOpenWin( 0.10 * ScreenWid,
                    0.10 * ScreenHt,
                    0.50 * ScreenWid,
                    0.50 * ScreenHt,
                    1000, 0, OWMFIXED,
                    "Graphing window" );
if ( WinNum < 0 ) {
   /* GSOpenWin failed */
}
/* enable mouse event notification */
GSMNotify( hWndHdlr, WM_MOUSEEVENT, MNTRACK |
           MNPRESS );
```

The window procedure for the window hWndHdlr might contain this:

```
LONG WndHdlr_WndProc( HWND hWnd, UINT uMsg, UINT
                      wParam, LONG lParam )
{
switch ( uMsg ) {
case WM_MOUSEEVENT:
/* the user caused a mouse event so see what type
   it is */
   switch ( wParam ) {
case MNTRACK:
/* a tracking event etc. */
case MNPRESS:
   /* a button press or release event etc. */
}
}
}
```

J

Topic

GSMNotify

Related

GSMStatus
GSMMotion
GSMGetX
GSMGetY
GSMSetRgn
GSMClrRgn
GSUseView
GSHotGraph

GSMovePos function

Moves current position

C/C++

```
int GSMovePos( double fxr, double fya, int nMode )
```

FoxPro

```
r = GSMovePos(fxr, fya, nMode)
```

Visual Basic

```
r% = GSMovePos(fxr#, fya#, nMode%)
```

Parameters

`fxr`	X relative or radius		
`fya`	Y relative or angle		
`nMode`	**Constant**	**Value**	**Meaning**
		0	fxr, fya are X and Y distances
	MPPOLAR	1	fxr, fya are polar radius and angle

Return values

0	Success
-1	Failure

Description

The GSMovePos function moves the current position by a relative amount. You can specify the relative position either in Cartesian (X,Y) coordinates or in polar (radius and angle) notation.

J

Topic

GSMovePos

Related

GSFixPos
GSLineRel
GSGetCurX
GSGetCurY
GSPolyFill
GSPolyVec

GSMPtrOff function

Turns off mouse pointer

C/C++

```
int GSMPtrOff( )
```

FoxPro

```
r = GSMPtrOff()
```

Visual Basic

```
r% = GSMPtrOff()
```

Return values

0	Success
-1	Failure

Description

The GSMPtrOff function turns off the mouse pointer used for the client area of the current window. The pointer can later be restored using GSMPtrOn.

The status of the mouse pointer used anywhere outside the client area of a window can't be changed.

J

Topic

GSMPtrOff

Related

GSMPtrOn

GSMPtrType

GSMPtrOn function

Turns on mouse pointer

C/C++

```
int GSMPtrOn( )
```

FoxPro

```
r = GSMPtrOn()
```

Visual Basic

```
r% = GSMPtrOn()
```

Return values

0	Success
-1	Failure

Description

The GSMPtrOn function turns on the mouse pointer used for the client area of the current window.

The status of the mouse pointer used anywhere outside the client area of a window can't be changed.

The mouse pointer is always turned on for a new window when it's opened.

J

Topic

GSMPtrOn

Related

GSMPtrOff

GSMPtrType

GSMPtrType function

Defines mouse pointer shape

C/C++

```
int GSMPtrType( int nType )
```

FoxPro

```
r = GSMPtrType(nType)
```

Visual Basic

```
r% = GSMPtrType(nType%)
```

Parameter

`nType`

Constant	Value	Meaning
MCARROW	0	Standard oblique arrow
MCIBEAM	1	Text I-beam
MCWAIT	2	Hourglass
MCCROSS	3	Crosshair
MCUPARROW	4	Vertical arrow
MCSIZE	5	Four-pointed horizontal/vertical arrow
MCICON	6	Empty icon
MCSIZENWSE	7	Two-pointed oblique arrow
MCSIZENESW	8	Two-pointed oblique arrow
MCSIZEWE	9	Two-pointed horizontal arrow
MCSIZENS	10	Two-pointed vertical arrow

Return values

0	Success
-1	Failure

Description

The GSMPtrType function sets the shape of the mouse pointer used for the client area of the current window to one of a range of system pointer shapes.

The shape of the mouse pointer used anywhere outside the client area of a window can't be changed.

A standard oblique arrow pointer is automatically selected for a new window when it's opened.

Topic

GSMPtrType

Related

GSMPtrOn

GSMPtrOff

GSMSetRgn function

Defines mouse hot region

C/C++

```
int GSMSetRgn( double fxr, double fya, int nMode,
               double fAng )
```

FoxPro

```
r = GSMSetRgn(fxr, fya, nMode, fAng)
```

Visual Basic

```
r% = GSMSetRgn(fxr#, fya#, nMode%, fAng#)
```

Parameters

`fxr`	X origin or radius
`fya`	Y origin or angle
`nMode`	Function mode
`fAng`	Angle of rotation

Return values

0 or greater	Mouse hot region number
-1	Failure

Description

The GSMSetRgn function defines a mouse hot region in the current window from a number of connected points defined in the amplitude array, fA.

Frequently, you'll want your application to be aware of mouse clicks only when they occur in selected regions of a graphing window. These mouse-sensitive regions are called "hot regions."

GSMSetRgn defines a region of the current window as a mouse hot region. A hot region number is returned that uniquely identifies the new region in the window. Once a region is defined, it's automatically monitored and the application is notified with an MNHIT mouse event message whenever a mouse click occurs within the boundary. To enable mouse notification, you have to call the GSMNotify function with a mode of MNHIT.

Hot regions can overlap in a window. When a button is pressed in an area of overlapping regions, an MNHIT event is notified for each region.

If you want a hot region to be visible, you can follow the GSMSetRgn call with a call to the GSPolyFill function, using the same first four parameters and the same point data.

To clear a hot region, use the GSMClrRgn function.

GSDataTrans parameters for hot regions

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (always 2)
fA[nPts][nGroup]	Pointer to amplitude array (first set contains X positions of points or vector lengths, second

	set contains Y positions of points or vector angles)
fD[0]	Not used
nPatt[0]	Not used
nSymbol[0]	Not used
nAux[0]	Not used
nClr[0]	Not used

J

Topic

GSMSetRgn

Related

GSMClrRgn
GSMNotify
GSPolyFill
GSMStatus

GSMStatus function

Reads mouse button status

C/C++

```
int GSMStatus( )
```

FoxPro

```
r = GSMStatus()
```

Visual Basic

```
r% = GSMStatus()
```

Return values

0	FALSE--no buttons pressed
1	MBLEFT--left button pressed
2	MBMIDDLE--middle button pressed
3	MBRIGHT--right button pressed

Description

The GSMStatus function returns a numeric value indicating which mouse buttons are pressed. Values may be combined. For example, MBLEFT + MBRIGHT means the left and right buttons are pressed together.

Concurrent with this it freezes the position information returned using GSMGetX and GSMGetY. This is to achieve synchronism between the time a button is pressed and the position that is recorded.

The status of a mouse button pressed outside the client area of a window can't be read.

GSMStatus must be called before reading back actual position information.

You shouldn't call GSMStatus in a loop waiting for button status to change. Calling GSMNotify with mode MNPRESS provides the means of monitoring button status changes asynchronously through the application message queue. GSMStatus must continue to be called to freeze mouse position information but only in response to mouse MNPRESS event messages.

For the means of monitoring button presses in precise regions of the client area, refer to GSMNotify with mode MNHIT and functions GSMSetRgn and GSMClrRgn.

J

Topic

GSMStatus

Related

GSMNotify
GSMSetRgn
GSMClrRgn
GSMGetX

GSMGetY
GSMMotion

GSOffView function

Turns off view

C/C++

```
int GSOffView( int nWin, int nView )
```

FoxPro

```
r = GSOffView(nWin, nView)
```

Visual Basic

```
r% = GSOffView(nWin%, nView%)
```

Parameters

`nWin`	Window number
`nView`	View number

Return values

0	Success
-1	Failure

Description

The GSOffView function temporarily turns off a view, making it invisible. It may later be made visible using GSOnView.

Any view may be made invisible. This may be useful, for example, when printing a portion of a window on a hard copy device.

J

Topic

GSOffView

Related

GSOnView
GSUseView
GSCloseView
GSPrnOut

GSOnView function

Turns on view

C/C++

```
int GSOnView( int nWin, int nView )
```

FoxPro

```
r = GSOnView(nWin, nView)
```

Visual Basic

```
r% = GSOnView(nWin%, nView%)
```

Parameters

`nWin`	Window number
`nView`	View number

Return values

0	Success
-1	Failure

Description

The GSOnView function turns on a view that has previously been turned off using GSOffView.

J

Topic

GSOnView

Related

GSCloseView
GSOffView
GSUseView

GSOpenChildWin function

Opens graphing window as child of another window

C/C++

```
int GSOpenChildWin( HWND hWndParent, double fxOrg,
                    double fyOrg, double fWid,
                    double fHt, double fyExt,
                    int nStyle, int nMode,
                    char szTitle )
```

FoxPro

```
r = GSOpenChild(hWndParent, fxOrg, fyOrg, fWid, fHt,
                fyExt, nStyle, nMode, szTitle)
```

Visual Basic

```
r% = GSOpenChildWin(hWndParent%, fxOrg#, fyOrg#, fWid#,
                    fHt#, fyExt#, nStyle%, nMode%,
                    szTitle$)
```

Parameters

`hWndParent` Parent window handle

`fxOrg` Bottom left X origin

`fyOrg` Bottom left Y origin

`fWid` External width of window

`fHt` External height of window

`fyExt` Logical Y extent of view 0

`nStyle` Style of window

Constant	Value	Meaning
OWSTHICKFRAME	1	Includes a thick sizing frame around the window. By default, the window has no sizing frame and can't be resized by the user. Specifying OWSTHICKFRAME implies style OWSBORDER.
OWSBORDER	2	Includes a single-line black border around the window
OWSHSCROLL	4	Includes a horizontal scroll bar
OWSVSCROLL	8	Includes a vertical scroll bar
OWSMAXIMIZEBOX	16	Includes a maximize box in the window caption bar
OWSMINIMIZEBOX	32	Includes a minimize box in the window

		caption bar
OWSSETFOCUS	64	Sets the input focus to the graphing window after it has opened. By default, the input focus is left with the window that currently has it.
OWSHIDDEN	128	Opens a hidden window
OWSSYSMENU	256	Includes a system menu in the window caption bar
OWSCAPTION	512	Includes the window caption bar. Without this option, the system menu, minimize box, and maximize box are omitted and the user can't move the window.
OWSCLIPCHILDREN	1024	When drawing in the client area of this graphing window, clips any output that falls in an area occupied by a child window. Applicable to a window if you subsequently intend to make it the parent of another window.
OWSCLIPSIBLINGS	2048	When drawing in the client area of this graphing window, clips any output that falls in an area occupied by another child window.
OWSTRANSPARENT	4096	Opens a transparent window. Graphics Server won't draw a background in the window, so any detail underneath shows through.

`nMode` Method for scaling view contents of window when window size changes

Constant	Value	Meaning
OWMFIXED	0	The view isn't rescaled to fit the new window area.

		An unused margin, or clipping, may be apparent at both the top and right-hand edges. Both the horizontal and vertical scroll bars, if specified, may be activated.
OWMFITHORZ	1	The view is rescaled to give an exact fit to the new window width. The original view aspect ratio is maintained. An unused margin, or clipping, may appear at the top edge. The vertical scroll bar, if specified, may be activated.
OWMFITVERT	2	The view is rescaled to give an exact fit to the new window height. The original view aspect ratio is maintained. An unused margin, or clipping, may appear at the right-hand edge. The horizontal scroll bar, if specified, may be activated.
OWMFITBOTH	3	The view is rescaled so all of it is visible in the new window area. The original view aspect ratio is maintained. An unused margin may appear at either the top or right-hand edge. The view is never clipped and scroll bars, if specified, are never activated.
OWMFITOPT	4	The view is rescaled to the best fit between the original view aspect ratio and new aspect ratio of the window area. Clipping may occur at either the top or right-hand edges. Either the horizontal or vertical scroll bar, if specified, may be activated.
OWMFLEXIBLE	5	The view extents are adjusted to reflect the new window area.

	OWMCENTRED	256	In situations when an unused margin is apparent, the view is centered so that an equal margin appears at the top and bottom or left and right of the view. This mode is additive and may be applied to any of the other fitting modes, although the effect may not be apparent in all cases.
`szTitle`	Window name		

Return values

0 or greater	Window number of the graphing child window
-1	Failure

Description

The GSOpenChildWin function opens a new graphing window as the child of another window and returns a number by which it may be referenced.

When a window is first opened, it becomes the current window.

If you have more than one graphing window open, the GSUseView function is used to select between them.

The first parameter of this function is the Windows handle of the parent window.

The next four parameters define the position and size of the rectangular window in the client area of the parent. The units these are expressed in are called anchor units. The anchor space is a device-independent coordinate system, defined by Graphics Server to map the whole client area of the parent window.

The width and height of the parent client area, in anchor units, may be obtained by calling GSGetAXExt and GSGetAYExt. The ratio of these dimensions is called the aspect ratio and depends on the shape of the parent client area. A window also has an aspect ratio that may or may not reflect the aspect ratio of the parent. Typically, a graphing child window is positioned and sized using proportions of the dimensions of the parent anchor space.

When a graphing child window is opened, view 0 is automatically opened and is ready for drawing in. View 0, which is also called the default view, occupies the entire client area of the graphing window. The interior of view 0, and hence the client area of the window, is mapped by a logical view coordinate system. The units of this view coordinate system are the ones you use to position and size graphical objects. You choose the height of your view 0 coordinate space with the fyExt parameter. Graphics Server calculates the width of the coordinate space based on the

aspect ratio of the window.

Window fitting options

With most of the nMode fitting options the extents and aspect ratio of the view coordinate space are fixed on opening the window and don't reflect subsequent changes in the size and shape of the window. The fitting mode OWMFLEXIBLE, however, ensures that the extents and aspect ratio of your view coordinate system can be kept consistent with those of the window. The initial view extents are based on the aspect ratio of the window at the time of opening, but are subsequently adjusted whenever the size and shape of the window is changed.

In many other aspects, the OWMFLEXIBLE fitting mode behaves the same as OWMFIXED. Unless you redraw it, your image won't change size and an unused margin, or clipping, may be apparent on the new outline of the window. The scroll bar styles, OWSHSCROLL and OWSVSCROLL, aren't supported in windows opened in OWMFLEXIBLE mode.

Example

The following example illustrates opening a graphing child window.

```
double ClientWid, ClientHt;
int WinNum;
/* hWndParent is the handle of the prospective parent window
*/
ClientWid = GSGetAXExt( hWndParent );
ClientHt = GSGetAYExt( hWndParent );
WinNum = GSOpenChildWin( hWndParent,
                         0.10 * ClientWid,
                         0.10 * ClientHt,
                         0.50 * ClientWid,
                         0.50 * ClientHt,
                         1000, 0, OWMFIXED,
                         "" );
if ( WinNum < 0 ) {
   /* GSOpenChildWin failed */
}
```

This example specifies that the new window is 50% of the parent client area width wide and 50% of the client area height high. The bottom left corner of the window is located 10% of the width in and 10% of the height up from the bottom left corner of the parent client area.

J

Topic

GSOpenChildWin

Related

GSOpenWin
GSCloseWin
GSOpenView
GSUseView
GSGetAXExt
GSGetAYExt
GSWinHandle
GSWinNotify
GSWinPaint

Server:

GSCloseServer
GSOpenServer

GSOpenPrn function

Opens printer for current window

C/C++

```
int GSOpenPrn( char szDevice, char szFile, int nMode )
```

FoxPro

```
r = GSOpenPrn(szDevice, szFile, nMode)
```

Visual Basic

```
r% = GSOpenPrn(szDevice$, szFile$, nMode%)
```

Parameters

`szDevice` Printer device name. If szDevice is a null string, the default printer shown in the [devices] section of the WIN.INI file is opened.

`szFile` File name. If szFile is a null string, output is directed to the port assigned to the device in the WIN.INI file.

`nMode`

Constant	Value	Meaning
PRNSETMODE	1	Shows the print setup dialog to enable the user to configure the device
PRNPORTRAIT	2	Prints in portrait mode
PRNLANDSCAPE	4	Prints in landscape mode
PRNSETMODECANCEL	8	Aborts the print setup dialog if the Cancel button is pressed

Return values

0	Success
-1	Failure

Description

The GSOpenPrn function opens a printer device associated with the current window and optionally directs output to a named file.

PRNSETMODECANCEL operates as PRNSETMODE, but returns FAIL if the Cancel button is pressed. This enables printing to be conditionally aborted.

Note that this feature works in Windows 3.1 but not in Windows 95. In the latter case, both OK and Cancel return FAIL if nothing has been changed in the dialog. This is in line with its meaning, which is not an OK/Cancel of print but rather OK/Cancel the dialog.

Example

The following example illustrates how to open the default printer to print the current window in landscape mode:

```
if ( GSOpenPrn( "", "", PRNSETMODE ) == SUCCESS ){
```

```
   /* printer opened successfully so go ahead
   and print the current window */
   GSPrnOut( 0, 1, PRNWINDOW | PRNFF );
   GSClosePrn();
{
else {
   /* unable to open the default printer */
}
```

The following example illustrates how to open a specific printer and present the user with the print setup dialog so that the printing options can be chosen at run time:

```
if ( GSOpenPrn( "HP LaserJet Series II", "",
                PRNSETMODE ) == SUCCESS ) {
   /* printer opened successfully so go ahead
   and print the current window */
   GSPrnOut( 0, 1, PRNWINDOW | PRNFF );
   GSClosePrn();
{
else {
   /* unable to open the default printer */
}
```

J

Topic

GSOpenPrn

Related

GSClosePrn
GSGetPrnWid
GSGetPrnHt
GSPrnSetup
GSPrnOut

GSOpenServer function

Opens connection to Graphics Server

C/C++

```
int GSOpenServer( char szKey, char szHost )
```

FoxPro

```
r = GSOpenServer(szKey, szHost)
```

Visual Basic

```
r% = GSOpenServer(szKey$, szHost$)
```

Parameters

`szKey` Run-time DLL key (always a null string)

`szHost`

String	Meaning
"C"	Show Graphics Server icon on desktop
"HC"	Don't show Graphics Server icon

Return values

0	Success
-1	Failure

Description

The GSOpenServer function opens a connection with Graphics Server. You have to call it before executing any graphics functions.

J

Topic

GSOpenServer

Related

GSCloseServer

GSCloseWin

GSOpenView function

Opens view

C/C++

```
int GSOpenView( int nWin, double fxOrg, double fyOrg,
                double fWid, double fHt, double fyExt )
```

FoxPro

```
r = GSOpenView(nWin, fxOrg, fyOrg, fWid, fHt, fyExt)
```

Visual Basic

```
r% = GSOpenView(nWin%, fxOrg#, fyOrg#, fWid#, fHt#,
                fyExt#)
```

Parameters

`nWin`	Number of window in which to open the view
`fxOrg`	X bottom left
`fyOrg`	Y bottom left
`fWid`	Width of view
`fHt`	Height of view
`fyExt`	Y extent (height) of view coordinate space. Graphics Server calculates the width of the coordinate space based on the aspect ratio of the view. Different views may have quite different view coordinate spaces.

Return values

1 or greater	View number
-1	Failure

Description

The GSOpenView function opens a new view within a window, returning a view number that identifies it.

When a window is first opened by the GSOpenWin function, view 0 is automatically opened and is ready for drawing.

When a new view is opened, it immediately becomes the current view. Drawing may proceed immediately.

If other views are open, use the GSUseView function to select the view you want to be current.

The four parameters after nWin define the position and size of the rectangular view in the graphing window. These are expressed in anchor units. Anchor space is a device-independent coordinate system, defined by Graphics Server to map the whole client area of the graphing window.

The width and height of the graphing window client area, in anchor units, may be obtained by calling GSGetWXExt and GSGetWYExt. The ratio of these dimensions is called the aspect ratio and depends on the shape of the graphing window client area.

A view also has an aspect ratio, which may or may not reflect the aspect ratio of the graphing window. Typically, a view is

positioned and sized using proportions of the dimensions of the graphing window anchor space.

The interior of the new view is mapped by a logical view coordinate system. The units of this view coordinate system are the ones you use to position and size graphical objects.

Within a view, units in X and Y are always the same size (the units are isotropic).

J

Topic

GSOpenView

Related

GSClearView
GSCloseView
GSGetVXExt
GSGetVYExt
GSOffView
GSOnView
GSUseView

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

Server:

GSCloseServer
GSOpenServer

GSOpenWin function

Opens graphing window

C/C++

```
int GSOpenWin( double fxOrg, double fyOrg, double fWid,
               double fHt, double fyExt, int nStyle,
               int nMode, char szTitle )
```

FoxPro

```
r = GSOpenWin(fxOrg, fyOrg, fWid, fHt, fyExt, nStyle,
              nMode, szTitle)
```

Visual Basic

```
r% = GSOpenWin(fxOrg#, fyOrg#, fWid#, fHt#, fyExt#,
               nStyle%, nMode%, szTitle$)
```

Parameters

`fxOrg` Bottom left X origin

`fyOrg` Bottom left Y origin

`fWid` External width of window

`fHt` External height of window

`fyExt` Logical Y extent of view 0

`nStyle` Style of window

Constant	Value	Meaning
OWSTHICKFRAME	1	Includes a thick sizing frame around the window. By default, the window has no sizing frame and can't be resized by the user. Specifying OWSTHICKFRAME implies style OWSBORDER.
OWSBORDER	2	Includes a single-line black border around the window
OWSHSCROLL	4	Includes a horizontal scroll bar
OWSVSCROLL	8	Includes a vertical scroll bar
OWSMAXIMIZEBOX	16	Excludes the default maximize box in the window caption bar
OWSMINIMIZEBOX	32	Excludes the default minimize box in the window caption bar
OWSSETFOCUS	64	Sets the input focus to the graphing window after it has opened.

		By default, the input focus is left with the window that currently has it.
OWSHIDDEN	128	Opens a hidden window
OWSSYSMENU	256	Excludes the default system menu in the window caption bar
OWSCLIPCHILDREN	1024	When drawing in the client area of this graphing window, clips any output that falls in an area occupied by a child window. Applicable to a window if you subsequently intend to make it the parent of another window.
OWSTRANSPARENT	4096	Open a transparent window. Graphics Server won't draw a background in the window, so any detail underneath shows through.

`nMode` Method for scaling view contents of window when window size changes

Constant	Value	Meaning
OWMFIXED	0	The view isn't rescaled to fit the new window area. An unused margin, or clipping, may be apparent at both the top and right-hand edges. Both the horizontal and vertical scroll bars, if specified, may be activated.
OWMFITHORZ	1	The view is rescaled to give an exact fit to the new window width. The original view aspect ratio is maintained. An unused margin, or clipping, may be apparent at the top edge. The vertical scroll bar, if specified, may be activated.
OWMFITVERT	2	The view is rescaled to give an exact fit to the

			new window height. The original view aspect ratio is maintained. An unused margin, or clipping, may be apparent at the right-hand edge. The horizontal scroll bar, if specified, may be activated.
	OWMFITBOTH	3	The view is rescaled so all of it is visible in the new window area. The original view aspect ratio is maintained. An unused margin may be apparent at either the top or right-hand edge. The view is never clipped and scroll bars, if specified, are never activated.
	OWMFITOPT	4	The view is rescaled to the best fit between the original view aspect ratio and new aspect ratio of the window area. Clipping may be apparent at either the top or right-hand edges. Either the horizontal or vertical scroll bar, if specified, may be activated.
	OWMFLEXIBLE	5	The view extents are adjusted to reflect the new window area.
	OWMCENTERD	256	In situations when an unused margin is apparent, the view is centered so that an equal margin appears at the top and bottom or left and right of the view. This mode is additive and may be applied to any of the other fitting modes, although the effect may not be apparent in all cases.
`szTitle`	Window name		

Return values	0 or greater	Window number of the graphing window
	-1	Failure

Description

The GSOpenWin function opens a new graphing window and returns a number by which it may be referenced.

When a window is first opened, it becomes the current window. If you have more than one graphing window open, use the GSUseView function to select between them.

The first four parameters of this function define the position and size of the rectangular window on the display. The coordinates and size are expressed in anchor units. Anchor space is a device-independent coordinate system defined by Graphics Server to enable graphing windows to be opened without recourse to device units, such as pixels.

The width and height of the display, in anchor units, may be obtained by calling GSGetSXExt and GSGetSYExt. The ratio of these dimensions is called the aspect ratio and depends on the type of display adapter you're using. A window also has an aspect ratio, which may or may not reflect the aspect ratio of the display. Typically, a graphing window is positioned and sized using proportions of the dimensions of the anchor space.

When a graphing window is opened, view 0 is automatically opened and is ready for drawing in. View 0, which is also called the default view, occupies the entire client area of the graphing window. The interior of view 0, and hence the client area of the window, is mapped by a logical view coordinate system. The units of this view coordinate system are the ones you use to position and size graphical objects. You choose the height of your view 0 coordinate space with the fyExt parameter. Graphics Server calculates the width of the coordinate space based on the aspect ratio of the window.

Window fitting options

With most of the nMode fitting options, the extents and aspect ratio of the view coordinate space are fixed on opening the window and don't reflect subsequent changes in the size and shape of the window. The fitting mode OWMFLEXIBLE, however, ensures that the extents and aspect ratio of your view coordinate system can be kept consistent with those of the window. The initial view extents are based on the aspect ratio of the window at the time of opening, but are subsequently adjusted whenever the size and shape of the window is changed.

In many other aspects the OWMFLEXIBLE fitting mode behaves the same as OWMFIXED. Unless you redraw it, your image won't change size and an unused margin, or clipping, may be apparent on the new outline of the window. The scroll bar styles, OWSHSCROLL and OWSVSCROLL, aren't supported in windows opened in OWMFLEXIBLE mode.

Example

The following example illustrates opening a graphing window:

```
double ScreenWid, ScreenHt;
```

```
int WinNum;
ScreenWid = GSGetSXExt();
ScreenHt = GSGetSYExt();
WinNum = GSOpenWin( 0.10 * ScreenWid,
                    0.10 * ScreenHt,
                    0.50 * ScreenWid,
                    0.50 * ScreenHt,
                    1000, 0, OWMFIXED,
                    "Graphing window" );
if ( WinNum < 0 ) {
   /* GSOpenWin failed */
}
```

This example specifies that the new window is 50% of the display width wide and 50% of the display height high. The bottom left corner of the window is located 10% of the width in and 10% of the height up from the bottom left corner of the display.

J

Topic

GSOpenWin

Related

GSOpenChildWin
GSCloseWin
GSCloseView
GSUseView
GSGetSXExt
GSGetSYExt
GSOpenView
GSWinHandle
GSWinNotify
GSWinPaint

Server:

GSCloseServer
GSOpenServer

GSPicRead function

Reads image from file

C/C++

```
int GSPicRead( double fxBL, double fyBL, double fWid,
               double fHt, int nFormat, int nMode,
               char szFile )
```

FoxPro

```
r = GSPicRead(fxBL, fyBL, fWid, fHt, nFormat, nMode,
              szFile)
```

Visual Basic

```
r% = GSPicRead(fxBL#, fyBL#, fWid#, fHt#, nFormat%,
               nMode%, szFile$)
```

Parameters

`fxBL`	X bottom left		
`fyBL`	Y bottom left		
`fWid`	Image width		
`fHt`	Image height		
`nFormat`	**Constant**	**Value**	**Meaning**
	PXPMF	2	Placeable metafile format
	PXDIB	4	Windows device-independent bitmap format
`nMode`	**Constant**	**Value**	**Meaning**
		0	The image is located at the bottom left corner of the area and retains its original dimensions, with free space or clipping possible at both the top and right-hand edges.
	PXCENTER	1	For a bitmap, the image retains its original dimensions and the center of the image is located at the center of the area, with free space or clipping possible at both the horizontal and vertical edges. This mode isn't functional for a metafile.
	PXSTRETCH	2	The image is located at the bottom left corner of the area and is stretched or compressed in either direction to give an exact fit in the area.
	PXTILE	3	For a bitmap, the image retains its original

		dimensions and is tiled repetitively from left to right and bottom to top of the area. For a metafile, this is an additional stretching mode. The image is located at the bottom left corner of the area and is stretched or compressed, retaining the original aspect ratio. Free space or clipping is possible at the top or right-hand edge.
	`szFile`	Image filename

Return values

0	Success
-1	Failure

Description

The GSPicRead function reads an image from a file into the current view.

The rectangular area of the image in the view is defined by the bottom left corner and a width and height, all expressed in the current view units. If zero width and height are specified the area is presumed to extend from the point of origin to the current extents of width and height of the view.

The image file name may be any valid drive, directory and file name combination. Graphics Server doesn't use the image format to imply a file extension.

The function handles image files in a variety of different formats. The nFormat parameter enables your application to specify in what format the file exists.

The format options are mutually exclusive. If Graphics Server detects that the file isn't in the specified format, no image will be imported. Once imported, the image from the file becomes a permanent part of the view.

The image in the file may be larger or smaller than the view area defined by the fxBL, fyBL, fWid, and fHt parameters. The nMode parameter specifies how the imported image is to fit the available area. There's a difference in the way some of the modes work, depending on whether the image is a bitmap or a metafile.

Example

The following example copies a device-independent bitmap image from a file into the current view. The view is divided into quadrants and the image is copied into each of the quadrants using the different modes available.

```
void OnEditPaste()
{
static char strFile [] = "CAT.BMP";
double  fxMid, fyMid;
fxMid = GSGetVXExt() / 2;
fyMid = GSGetVYExt() / 2;
GSPicRead( 0, 0, fxMid, fyMid, PXDIB, 0,
           strFile );
GSPicRead( fxMid, 0, fxMid, fyMid, PXDIB,
           PXCENTER, strFile );
GSPicRead( 0, fyMid, fxMid, fyMid, PXDIB,
           PXSTRETCH, strFile );
GSPicRead( fxMid, fyMid, fxMid, fyMid, PXDIB,
           PXTILE, strFile );
}
```

J

Topic

GSPicRead

Related

GSPicWrite
GSClipRead
GSClipWrite
GSGetMF

GSPicWrite function

Writes image to file

C/C++

```
int GSPicWrite( double fxBL, double fyBL, double fWid,
                double fHt, int nFormat, nMode,
                char szFile )
```

FoxPro

```
r = GSPicWrite(fxBL, fyBL, fWid, fHt, nFormat, nMode,
               szFile)
```

Visual Basic

```
r% = GSPicWrite(fxBL#, fyBL#, fWid#, fHt#, nFormat%,
                nMode%, szFile$)
```

Parameters

`fxBL`	X bottom left		
`fyBL`	Y bottom left		
`fWid`	Image width		
`fHt`	Image height		
`nFormat`	**Constant**	**Value**	**Meaning**
	PXPMF	2	Placeable metafile format
	PXWMF	3	Windows metafile format
	PXDIB	4	Windows device-independent bitmap format
	PXPCX	5	PCX format
	PXJPEG	6	JPEG format
	PXGIF	7	GIF format
`nMode`	**Constant**	**Value**	**Meaning**
	CBMONO	256	Exports the image in monochrome mode
`szFile`	Image filename		

Return values

0	Success
-1	Failure

Description

The GSPicWrite function writes an image of the current graphing window to a file.

The rectangular area of the window is defined by the bottom left corner and a width and height, all expressed in the view units of view 0, the default view. If zero width and height are specified the area is presumed to extend from the point of origin to the current extents of width and height of the view. The option to specify an area of the window isn't supported in this release. An image of the whole window is always exported.

You can choose to export the image in monochrome mode by means of the nMode parameter.

The image file name may be any valid drive, directory and file name combination. Graphics Server doesn't use the image format to imply a file extension. The file is opened in create mode and any existing version will be overwritten.

The function handles image files in a variety of different formats. The nFormat parameter enables your application to specify in what format you want the file to be written.The format options are mutually exclusive.

JPEG and GIF formats require an appropriate conversion DLL to be in the home directory of GSW*16/32*.EXE. For JPEG, the conversion DLL is GSJPG*16/32*.DLL. For GIF, the conversion DLL is GSGIF*16/32*.DLL.

If the conversion DLL is not found, the function will fail. In a 32-bit environment, the function will return an error. In a 16-bit environment, a message will appear saying that the DLL is not present and then an error will be returned. In neither case will the failure affect subsequent operation of the program.

Note: The JPEG conversion DLLs are on the installation CD. Because of licensing restrictions, libraries for converting to GIF are available only from our web site. Before using them, you must obtain a license from Unisys, who hold the patent for LZW compression.

JPEG compression is fixed, with a quality of 90. JPEG always produces some loss of image quality usually seen as a noise or streaking. An image quality of 90 has been chosen as a suitable compromise between quality and compression.

Example

The following example copies an image of the current window to a file in the current directory in Windows metafile format:

```
void OnEditCopy()
{
GSPicWrite( 0, 0, 0, 0, PXWMF, 0,
            "MYWINDOW.WMF" );
}
```

J

Topic

GSPicWrite

Related

GSPicRead

GSClipRead

GSClipWrite

GSGetMF

GSWriteRegionFile

GSPie2D function

Draws 2D pie chart

C/C++

```
int GSPie2D( double fxOrg, double fyOrg, double fRad,
             int nMode )
```

FoxPro

```
r = GSPie2D(fxOrg, fyOrg, fRad, nMode)
```

Visual Basic

```
r% = GSPie2D(fxOrg#, fyOrg#, fRad#, nMode%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fRad`	Radius
`nMode`	Function mode (no modes currently implemented)

Return values

0	Success
-1	Failure

Description

The GSPie2D function draws a pie chart centered at (fxOrg,fyOrg) with radius fRad. The size of each pie "slice" is in proportion to the total of all values in the data set.

Selecting "exploded" pie slices

You can use the nAux array--through the GSDataAux function--to "explode" certain pie slices (move them slightly away from the center of the pie).

nAux setting	**Constant**	**Value**	**Meaning**
	PCNOEXPL	0	Slice isn't exploded
	PCEXPL	1	Slice is exploded

GSDataTrans parameters for 2D pie charts

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (always 1)
fA[nPts]	Pointer to amplitude array (values for pie slices)
fD[0]	Not used
nPatt[nPts]	Pointer to array containing one fill pattern for each pie slice
nSymbol[0]	Not used
nAux[nPts]	Pointer to array containing "explode" values for pie slices

nClr[nPts]	Pointer to array containing one color for each pie slice

J

Topic
GSPie2D

Related
GSPie3D

Labels:
GSLabelPie
GSLabelnPie

Legend:
GSLegend

Array initialization:
GSDataTrans

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSPie3D function

Draws 3D pie chart

C/C++

```
int GSPie3D( double fxOrg, double fyOrg, double fRad,
             double fDepth, double fAng, int nMode )
```

FoxPro

```
r = GSPie3D(fxOrg, fyOrg, fRad, fDepth, fAng, nMode)
```

Visual Basic

```
r% = GSPie3D(fxOrg#, fyOrg#, fRad#, fDepth#, fAng#,
             nMode%)
```

Parameters

`fxOrg`	X origin		
`fyOrg`	Y origin		
`fRad`	Radius		
`fDepth`	Projected depth of pie		
`fAng`	Angle at which pie is tilted to vertical		
`nMode`	**Constant**	**Value**	**Meaning**
	PCSAMECLR	2	Sides of slices in same colors as tops

Return values

0	Success
-1	Failure

Description

The GSPie3D function draws a 3D pie chart centered at (fxOrg,fyOrg) with radius fRad. The size of each pie "slice" is in proportion to the total of all values in the data set.

By default, the side of each pie slice is colored with the half-color (the base color dithered with black) of the top of the slice. You can use an nMode of PCSAMECLR to color the sides the same as the tops.

Selecting "exploded" pie slices

You can use the nAux array--through the GSDataAux function--to "explode" certain pie slices (move them slightly away from the center of the pie).

nAux setting	**Constant**	**Value**	**Meaning**
	PCNOEXPL	0	Slice isn't exploded
	PCEXPL	1	Slice is exploded

GSDataTrans parameters for 2D pie charts

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (always 1)

fA[nPts]	Pointer to amplitude array (values for pie slices)
fD[0]	Not used
nPatt[nPts]	Pointer to array containing one fill pattern for each pie slice
nSymbol[0]	Not used
nAux[nPts]	Pointer to array containing "explode" values for pie slices
nClr[nPts]	Pointer to array containing one color for each pie slice

J

Topic
GSPie3D

Related
GSPie2D

Labels:
GSLabelPie
GSLabelnPie

Array initialization:
GSDataTrans
GSDataAux

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSPolar function

Draws polar graph

C/C++

```
int GSPolar( double fxOrg, double fyOrg, double fAng,
             int nMode, int nClr )
```

FoxPro

```
r = GSPolar(fxOrg, fyOrg, fAng, nMode, nClr)
```

Visual Basic

```
r% = GSPolar(fxOrg#, fyOrg#, fAng#, nMode%, nClr%)
```

Parameters

`fxOrg` X center

`fyOrg` Y center

`fAng` Angular distance between adjacent data points. Optionally, the angular position of each point may be taken from the fD array rather than placing points at fixed intervals.

`nMode`

Constant	Value	Meaning
POGLINE	1	Connects points with lines
POGSYMBOL	2	Draws symbols at points
POGSTICK	4	Draws vertical sticks to points
POGVARANG	8	Uses fD for angular position
POGTHICK	16	Uses thick lines
POGPATT	32	Uses patterned lines
POGGROUPED	64	Multiple data-set mode

`nClr` Color of markers if you're graphing only one data set. With multiple data sets, you have to create a color array. (See Color constants.)

Return values

0 Success

-1 Failure

Description

The GSPolar function draws a polar graph centered at (fxOrg,fyOrg).

The graph may be drawn as lines, symbols, sticks radiating from the center, or a combination of these. You can assign different line styles, line thicknesses, or symbol designs to each data set.

Specifying missing-data points

You can use the nAux array--through the GSDataAux function--to flag points of a polar graph as "missing." Missing points aren't shown, whether or not you've provided values for them. If you use lines to connect points (nMode POGLINE), the connecting lines are omitted both to and from each missing point.

nAux setting	**Value**	**Meaning**
	0	Point shown normally
	256	Point is "missing" and not drawn

The size of the nAux array may be nPts or nPts nGroup. If you set the size to nPts and there's more than one group of data, the same missing points are assumed for all the groups. If the size is nPts nGroup, each point in each group has its own missing-data flag.

GSDataTrans parameters for polar graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (radial distance of plotted points from center)
fD[nPts]	Pointer to distance array (angular positions of plotted points)--used only with nMode POGVARANG
nPatt[nPts]	Pointer to array containing line style or thickness for each point--used only with nMode POGTHICK or POGPATT
nSymbol[nPts]	Pointer to array containing symbol design for each point
nAux[nPts]	Pointer to array containing missing-data flag for each point
nClr[0]	Not used

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (radial distance of plotted points from center)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (angular positions of plotted points)--used only with nMode POGVARANG
nPatt[nGroup]	Pointer to array containing line style or thickness for each data set--used only with nMode POGTHICK or POGPATT

nSymbol[nGroup]	Pointer to array containing symbol design for each data set
nAux[nPts] *or* nAux[nPts][nGroup]*	Pointer to array containing missing-data flag for each point
nClr[nGroup]	Pointer to array containing color for each data set

* GSDataTrans can't pass two-dimensional fD or nAux arrays. You have to use the GSDataDist or GSDataAux function if you want to specify individual fD or nAux values for each data set. However, you can use GSDataTrans if you want to apply the same fD or nAux values to points in all sets.

J

Topic

GSPolar

Related

GSPolarAxes

Array initialization:

GSDataTrans
GSDataDist
GSDataAux

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

GSPolarAxes function

Draws set of polar axes

C/C++

```
int GSPolarAxes( double fxOrg, double fyOrg,
                 double fRad, int nRadDivs,
                 int nAngDivs, int nMode, int nStyle,
                 int nClr )
```

FoxPro

```
r = GSPolarAxes(fxOrg, fyOrg, fRad, nRadDivs, nAngDivs,
                nMode, nStyle, nClr)
```

Visual Basic

```
r% = GSPolarAxes(fxOrg#, fyOrg#, fRad#, nRadDivs%,
                 nAngDivs%, nMode%, nStyle%, nClr%)
```

Parameters

`fxOrg` X center

`fyOrg` Y center

`fRad` Radius

`nRadDivs` Number of radial divisions

`nAngDivs` Number of angular divisions. If angular grids are specified, lines are drawn radiating from the center at degree intervals of 90 divided by nAngDivs--that is, nAngDivs defines the number of angular divisions per quadrant.

`nMode`

Constant	Value	Meaning
PARADGRID	1	Draws radial grids
PAANGGRID	2	Draws angular grids
PATHICK	4	Uses thick line style

`nStyle` Grid line style (see Line style constants)

`nClr` Color of polar axes (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSPolarAxes function draws a set of polar axes and grids.

The axes comprise four arms radiating at 0, 90, 180, and 270 degrees, of length fRad, bounded by a circle at their extremes.

If you specify radial grids, circles are drawn with radii increasing at uniform intervals of fRad over nRadDivs.

Grids may be patterned or thickened according to the parameter nStyle.

J

Topic

GSPolarAxes

Related

GSAxis

GSPolar

GSPolyFill function

Draws polygon filled with pattern

C/C++

```
int GSPolyFill( double fxr, double fya, int nMode,
                double fAng, int nPatt, int nClr )
```

FoxPro

```
r = GSPolyFill(fxr, fya, nMode, fAng, nPatt, nClr)
```

Visual Basic

```
r% = GSPolyFill(fxr#, fya#, nMode%, fAng#, nPatt%,
                nClr%)
```

Parameters

`fxr` X origin or radius

`fya` Y origin or angle

`nMode`

Constant	Value	Meaning
PFXYORG	0	Origin is (X,Y)
PFRAORG	1	Origin is radius and angle
PFXYDATA	0	Data is (X,Y)
PFRADATA	2	Data is radius and angle
PFMIRRORV	4	Mirrors around vertical axis
PFMIRRORH	8	Mirrors around horizontal axis

`fAng` Angle of rotation

`nPatt` Fill pattern (see Pattern constants)

`nClr` Color of figure (see Color constants)

Return values

0 Success

-1 Failure

Description

The GSPolyFill function draws a figure of connected points as defined in the amplitude array, fA, and fills it with a pattern. If you specify an open polygon, Graphics Server adds a closing line from the last point to the first point.

You can express the position of the origin position as (X,Y) coordinates or as an angle and radius relative to the current position.

The values in the array may be expressed as (X,Y) coordinates or as a vector length and angle relative to the origin of the figure set by (fxr,fya).

You can have the figure be mirrored around the vertical axis, the horizontal axis, or both. Also, you can rotate the entire figure around the origin in a counterclockwise rotation. Mirroring is

performed before rotation.

Note that GSPolyFill allows two levels of relativity. The data elements defined in fA are relative to the origin of the figure. The origin (fxr,fya) defined in the function is relative to the current position.

GSDataTrans parameters for polygons

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (always 2)
fA[nPts][nGroup]	Pointer to amplitude array (first set contains X positions of points or vector lengths, second set contains Y positions of points or vector angles)
fD[0]	Not used
nPatt[0]	Not used
nSymbol[0]	Not used
nAux[0]	Not used
nClr[0]	Not used

J

Topic

GSPolyFill

Related

GSArc
GSBox2D
GSCircle
GSFixPos
GSMovePos
GSGetCurX
GSGetCurY
GSPolyVec
GSLineAbs
GSMClrRgn
GSMSetRgn
GSDefPatt

GSPolyVec function

Draws polyline figure

C/C++

```
int GSPolyVec( double fxr, double fya, int nMode,
               double fAng, int nStyle, int nClr )
```

FoxPro

```
r = GSPolyVec(fxr, fya, nMode, fAng, nStyle, nClr)
```

Visual Basic

```
r% = GSPolyVec(fxr#, fya#, nMode%, fAng#, nStyle%,
               nClr%)
```

Parameters

`fxr` X origin or radius

`fya` Y origin or angle

`nMode`

Constant	Value	Meaning
PVXYORG	0	Origin is (X,Y)
PVRAORG	1	Origin is radius and angle
PVXYDATA	0	Data is (X,Y)
PVRADATA	2	Data is radius and angle
PVMIRRORV	4	Mirrors around vertical axis
PVMIRRORH	8	Mirrors around horizontal axis
PVTHICK	16	Uses thick line style

`fAng` Angle of rotation

`nStyle` Line style (see Line style constants)

`nClr` Color of figure (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSPolyVec function draws a figure of connected points as defined in the amplitude array, fA.

You can express the position of the origin position as (X,Y) coordinates or as an angle and radius relative to the current position.

The values in the array may be expressed as (X,Y) coordinates or as a vector length and angle relative to the origin of the figure set by (fxr,fya).

You can have the figure be mirrored around the vertical axis, the horizontal axis, or both. Also, you can rotate the entire figure around the origin in a counterclockwise rotation. Mirroring is performed before rotation.

Note that GSPolyVec allows two levels of relativity. The data elements defined in fA are relative to the origin of the figure. The origin (fxr,fya) defined in the function is relative to the current position.

GSDataTrans parameters for polyline figures

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (always 2)
fA[nPts][nGroup]	Pointer to amplitude array (first set contains X positions of points or vector lengths, second set contains Y positions of points or vector angles)
fD[0]	Not used
nPatt[0]	Not used
nSymbol[0]	Not used
nAux[0]	Not used
nClr[0]	Not used

J

Topic

GSPolyVec

Related

GSArc
GSFixPos
GSMovePos
GSGetCurX
GSGetCurY
GSPolyFill
GSLineAbs

GSPrnOut function

Prints view or window

C/C++

```
int GSPrnOut( int nView, nNcopies, nMode )
```

FoxPro

```
r = GSPrnOut(nView, nNcopies, nMode)
```

Visual Basic

```
r% = GSPrnOut(nView%, nNcopies%, nMode%)
```

Parameters

`nView`	View number
`nNcopies`	Number of copies
`nMode`	

Constant	Value	Meaning
PRNWINDOW	0	Prints all views within a window
PRNVIEW	1	Prints a single view only
PRNFF	2	Transmits a form feed after printing
PRNFRAME	4	Prints a border around the output region
PRNCANCEL	8	Enable Cancel dialog when printing (default is no dialog)

Return values

0	Success
-1	Failure

Description

The GSPrnOut function prints the contents of a window or a view on the device selected with the GSOpenPrn function.

The function mode determines whether the entire current window, or just one view within it are printed.

If the entire window is printed, all views within that window are printed unless they are turned off by the GSOffView function. The nView parameter is ignored.

The printed image will fit completely within the whole page unless an alternative rectangular area has previously been defined using the GSPrnSetup function.

By default the "Cancel" dialog is not shown for printing. To enable this dialog, set the mode to PRNCANCEL.

J

Topic

GSPrnOut

Related

GSOpenPrn
GSClosePrn
GSPrnSetup
GSOffView
GSGetPrnHt
GSGetPrnWid

GSPrnSetup function

Sets printing area

C/C++

```
int GSPrnSetup( double fxBL, double fyBL, double fxTR,
                double fyTR, int nUnits, int nMode )
```

FoxPro

```
r = GSPrnSetup(fxBL, fyBL, fxTR, fyTR, nUnits, nMode)
```

Visual Basic

```
r% = GSPrnSetup(fxBL#, fyBL#, fxTR#, fyTR#, nUnits%,
                nMode%)
```

Parameters

`fxBL` X coordinate of bottom left corner

`fyBL` Y coordinate of bottom left corner

`fxTR` X coordinate of top right corner

`fyTR` Y coordinate of top right corner

`nUnits`

Constant	Value	Meaning
UNMM	1	Millimeters
UNINCH	2	Inches

`nMode`

Constant	Value	Meaning
PRNFIXED	0	The view isn't re-scaled to fit the printing area. An unused margin, or clipping, may be apparent at both the top and right-hand edges.
PRNFITHORZ	1	The view is re-scaled to give an exact fit to the printing area width. The original view aspect ratio is maintained. An unused margin, or clipping, may be apparent at the top edge.
PRNFITVERT	2	The view is re-scaled to give an exact fit to the printing area height. The original view aspect ratio is maintained. An unused margin, or clipping, may be apparent at the right-hand edge.
PRNFITBOTH	3	The view is re-scaled so that the whole of it is visible in the printing area. The original view aspect ratio is maintained. An unused margin may be apparent at

		either the top or right-hand edge. The view will never be clipped.
PRNFITOPT	4	The view is re-scaled to the best fit between the original view aspect ratio and aspect ratio of the printing area. Clipping may be apparent at either the top or right-hand edges.
PRNCENTERD	256	In situations when an unused margin is apparent, the view is centered so that an equal margin appears top and bottom, or left and right of the view, as the case may be. This mode is additive and may be applied to any of the other fitting modes, although the effect may not be apparent in all cases.

Return values

0	Success
-1	Failure

Description

The GSPrnSetup function sets the printing area on the hard-copy device selected by the GSOpenPrn function.

The rectangular area is defined in terms of the bottom left and top right corners, relative to the bottom and left side of the page.

You can get the size of the page using the GSGetPrnWid and GSGetPrnHt functions. Actual printing is initiated by the GSPrnOut function.

Example

The following function illustrates how an image might be printed in an area matching the aspect ratio of the original graphing window and centered horizontally and vertically on the printed page.

```
BOOL PrintGraph( void )
{
double fWinWid, fWinHt;
double fPageWid, fPageHt;
double fFrameWid, fFrameHt;
double fXOrg, fYOrg;
if ( GSOpenPrn( "", "", 0 ) != SUCCESS ) {
return FALSE;
}
/* Get the original size of the graphing window */
fWinWid = GSGetWXExt( GWWHOLE, UNMM );
fWinHt = GSGetWYExt( GWWHOLE, UNMM );
```

```
/* Get the paper size */
fPageWid = GSGetPrnWid( UNMM );
fPageHt = GSGetPrnHt( UNMM );
/* Compare the ratios of page height to window height and
page width to window width to establish the best fit */
if ( fPageHt / fWinHt < fPageWid / fWinWid ) {
   /* The window fits the page height best. Make
   the frame height to 80% of the page height to
   leave a 10% margin top and bottom. Set the
   frame width so as to preserve the aspect ratio
   of the window */
   fFrameHt = 0.80 * fPageHt;
   fFrameWid = fFrameHt * fWinWid / fWinHt;
}
else {
/* The window fits the page width best. Make
   the frame width 80% of the page width to leave
   a 10% margin left and right. Set the frame
   height so as to preserve the aspect ratio of
   the window */
   fFrameWid = 0.80 * fPageWid;
   fFrameHt = fFrameWid * fWinHt / fWinWid;
}
/* center the frame in the page */
fXOrg = (fPageWid - fFrameWid) / 2.0;
fYOrg = (fPageHt - fFrameHt) / 2.0;
GSPrnSetup( fXOrg, fYOrg,
            fXOrg + fFrameWid,
            fYOrg + fFrameHt,
            UNMM, PRNFITOPT );
GSPrnOut( 0, 1, PRNWINDOW | PRNFRAME | PRNFF );
GSClosePrn();
return TRUE;
}
```

J

Topic
GSPrnSetup

Related
GSOpenPrn
GSClosePrn
GSPrnOut
GSGetPrnWid
GSGetPrnHt

GSRText function

Draws raster text

C/C++

```
int GSRText( double fxOrg, double fyOrg, int nCSet,
             int nTMode, int nClr, char szString )
```

FoxPro

```
r = GSRText(fxOrg, fyOrg, nCSet, nTMode, nClr,
            szString)
```

Visual Basic

```
r% = GSRText(fxOrg#, fyOrg#, nCSet%, nTMode%, nClr%,
             szString$)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`nCSet`	Character set--selects between the system font and a user font loaded with the GSLoadRFont function (see Character set constants)
`nTMode`	Text mode (see Text mode constants)
`nClr`	Text color (see Color constants)
`szString`	Text string

Return values

0	Success
-1	Failure

Description

The GSRText function draws a line of raster text.

Alignment of the text with the origin (fxOrg,fyOrg) is determined by the text mode parameter.

Text modes TXUP90 and TXDOWN90 apply only to the user-defined raster font. Text mode TXEXACT only applies to vector fonts.

You can read system font character dimensions using the functions GSGetSFWid and GSGetSFHt.

J

Topic

GSRText

Related

GSLoadRFont
GSGetRTextHt
GSGetRTextWid
GSGetSFWid
GSGetSFHt

GSSetRFontFace

GSScatter function

Draws 2D scatter graph

C/C++

```
int GSScatter( double fxOrg, double fyOrg, int nMode,
               int nClr )
```

FoxPro

```
r = GSScatter(fxOrg, fyOrg, nMode, nClr)
```

Visual Basic

```
r% = GSScatter(fxOrg#, fyOrg#, nMode%, nClr%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`nMode`	Function mode (no modes currently implemented)
`nClr`	Color of markers if you're graphing only one data set. With multiple data sets, you have to create a color array. (See Color constants.)

Return values

0	Success
-1	Failure

Description

The GSScatter function draws a 2D scatter graph. The Y position of each point is taken from the amplitude (fA) array and the X position from thedistance (fD) array. When you use the GS functions, you must provide both amplitude and distance values for scatter graphs. Through the AutoGraph functions, you can provide no distance values and have X positions set automatically (the first point at 0, the second at 1, and so on).

Note that 3D scatter graphs aren't available through Graphics Server's standard (GS) functions. All 3D scatter graphs use True3D perspective, which is available only through AutoGraph functions.

Specifying missing-data points

You can use the nAux array--through the GSDataAux function--to flag points of a 2D scatter graph as "missing." Missing points aren't shown, whether or not you've provided values for them.

nAux setting	**Value**	**Meaning**
	0	Point shown normally
	256	Point is "missing" and not drawn

The size of the nAux array may be nPts or nPts nGroup. If you set the size to nPts and there's more than one group of data, the same missing points are assumed for all the groups. If the size is nPts nGroup, each point in each group has its own missing-data flag.

GSDataTrans parameters for 2D scatter graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions of points)
fD[nPts]	Pointer to distance array (X positions of points)
nPatt[0]	Not used
nSymbol[nPts]	Pointer to array containing symbol design for each point
nAux[nPts]	Pointer to array containing missing-data flag for each point
nClr[0]	Not used

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions of points)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions of points)
nPatt[0]	Not used
nSymbol[nGroup]	Pointer to array containing symbol design for each point
nAux[nPts] *or* nAux[nPts][nGroup]*	Pointer to array containing missing-data flag for each point
nClr[nGroup]	Pointer to array containing color for each data set

* GSDataTrans can't pass two-dimensional fD or nAux arrays. You have to use the GSDataDist or GSDataAux function if you want to specify individual fD or nAux values for each data set. However, you can use GSDataTrans if you want to apply the same fD or nAux values to points in all sets.

J

Topic
GSScatter

Related

GSXYGraph

Axis/grid/legend:

GSAxis
GSGrid
GSLegend

Labels:

GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:

GSDataTrans
GSDataDist
GSDataAux

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

GSSD function

Draws standard deviation lines

C/C++

```
int GSSD( int nStyle, int nClr )
```

FoxPro

```
r = GSSD(nStyle, nClr)
```

Visual Basic

```
r% = GSSD(nStyle%, nClr%)
```

Parameters

`nStyle`	Line style (see Line style constants)
`nClr`	Line color (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSSD function draws lines representing the standard deviation of a data set.

The lines are drawn relative to the origin of the immediately preceding graphing function. The lines are clipped within a window defined by the GSStatsWin function.

The formula for standard deviation is as follows:

```
SD = SQRT(SUM(y2) - n*ymean2)))/(n - 1)
```

J

Topic

GSSD

Related

GSMean
GSGetSD
GSStatsWin
GSStatsArr
GSMinMax

GSSelectPalette function

Selects extended palette with 128 entries

C/C++

```
int GSSelectPalette( int nMode )
```

FoxPro

```
r = GSSelectPalette(nMode)
```

Visual Basic

```
r% = GSSelectPalette(nMode%)
```

Parameters

`nMode`

Constant	Value	Meaning
PALDEFAULT	0	16-entry palette with standard RGBCMY
PALGREYSCALE	1	32-127 run from black to white (grayscales)
PALPASTEL	2	32-127 are six groups of 16 pastel colors, RGBCMY, from fully saturated to white
PALRGBCMY	3	32-127 are six groups of 16 colors in ascending intensities--R, G, B, C, M, Y
PALRAINBOW	4	32-127 are two groups of 48 graded hues in two intensities
PALREDSCALE	5	32-127 run from black to red
PALGREENSCALE	6	32-127 run from black to green
PALBLUESCALE	7	32-127 run from black to blue
PALCYANSCALE	8	32-127 run from black to cyan
PALMAGENTASCALE	9	32-127 run from black to magenta
PALYELLOWSCALE	10	32-127 run from black to yellow
PALUSER	11	Realizes an extended palette with existing palette values and any changes made by GSSetPal().

Return values

0 Success

-1

Failure

Description

TheGSSelectPalette function selects a palette with 128 entries.

Before this function is called, or after it's called with a mode of 0, Graphics Server respects only the 16 basic color selection indexes, 0-15, in all its functions. This is the default palette with low and high intensities of the pure hues red, green, blue, cyan, magenta, yellow, and white.

When you call GSSelectPalette with a mode other than 0, a palette containing 128 pure (non-dithered) hues is created for and used on machines with 256-color capabilities. With a 128-entry palette in effect, functions accept a color index in the range 0-127.

The 128-entry palettes are initialized as follows:

- Entries 0-15 remain the standard colors
- Entries 16-31 are the half-intensities of 0-15, used for shading
- Entries 32-127 are given default values determined by nMode

You can reprogram any entry in the palette using the GSSetPal function. However, you should generally leave entries 0-31 unchanged.

Using the PALUSER mode, the current values in the palette remain unchanged when the extended palette is realized. Those values may either be the default values or the values set in the immediately preceding call to GSSelectPalette plus any values explicitly set using GSSetPal. This mode can be used to create a user-defined extended palette.

Example

```
'Select the child window and make the background white
r = GSUseView(ChWNum, 0)
r = GSSelectPalette(0)
r = GSSetBG(LIGHT + WHITE)
r = GSClearView(CLOPAQUE)

'Select the user defined palette and load it
r = GSSelectPalette(11)
For i = 0 To 127
    'r = GSSetPal(i, 2 * i, 2 * i, 2 * i) 'Grey
    'r = GSSetPal(i, 2 * i, 0, 0)         'Red
    'r = GSSetPal(i, 0, 2 * i, 0)         'Green
    'r = GSSetPal(i, 0, 0, 2 * i)         'Blue
    'r = GSSetPal(i, 2 * i, 2 * i, 0)     'Yellow
    r = GSSetPal(i, 2 * i, 0, 2 * i)      'Magenta
    'r = GSSetPal(i, 0, 2 * i, 2 * i)     'Cyan
Next i
r = GSClearView(CLOPAQUE)
```

J

Topic

GSSelectPalette

Related

GSSetPal

GSSetBG

GSSetBG function

Sets background color

C/C++

```
int GSSetBG( int nClr )
```

FoxPro

```
r = GSSetBG(nClr)
```

Visual Basic

```
r% = GSSetBG(nClr%)
```

Parameters

`nClr`	Color index number, referring to an entry in the current window palette (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSSetBG function sets the background color according to the current palette. To select a palette, use the GSSelectPalette function.

J

Topic

GSSetBG

Related

GSClearView
GSSetPal
GSSelectPalette

GSSetPal function

Sets palette

C/C++

```
int GSSetPal( int nClr, int nR, int nG, int nB )
```

FoxPro

```
r = GSSetPal(nClr, nR, nG, nB)
```

Visual Basic

```
r% = GSSetPal(nClr%, nR%, nG%, nB%)
```

Parameters

`nClr`	Color index (0-127)
`nR`	Red intensity value
`nG`	Green intensity value
`nB`	Blue intensity value

Return values

0	Success
-1	Failure

Description

The GSSetPal function sets an entry in the color palette. A separate palette is maintained for each window, which contains either 16 or 128 entries. The default palette contains 16 entries, but you can choose an extended palette of 128 entries using the GSSelectPalette function.

When you create a color, you specify intensities (in the range 0-255) for red, green, and blue. As an example, the following table shows the intensities for the default 16 colors:

Color name	nClr index	Red	Green	Blue
BLACK	0	0	0	0
BLUE	1	0	0	128
GREEN	2	0	128	0
CYAN	3	0	128	128
RED	4	128	0	0
MAGENTA	5	128	0	128
BROWN	6	128	128	0
WHITE	7	192	192	192
GRAY	8	128	128	128
LIGHT BLUE	9	0	0	255
LIGHT GREEN	10	0	255	0
LIGHT CYAN	11	0	255	255
LIGHT RED	12	255	0	0
LIGHT MAGENTA	13	255	0	255

YELLOW	14	255	255	0
LIGHT WHITE	15	255	255	255

J

Topic
GSSetPal

Related
GSSelectPalette
GSSetBG

GSSetRFontFace function

Sets typeface used for raster font family

C/C++

```
int GSSetRFontFace( int nFamily, char szFaceName )
```

FoxPro

```
r = GSSetRFontFace(nFamily, szFaceName)
```

Visual Basic

```
r% = GSSetRFontFace(nFamily%, szFaceName$)
```

Parameters

`nFamily`

Constant	Value	Meaning	Default font
FOROMAN	1	Roman	Times New Roman
FOSWISS	2	Swiss	Arial
FOMODERN	3	Modern	Courier New
FOSCRIPT	4	Script	(None specified)
FODECO	5	Decorative	(None specified)

`szFaceName` Face name for the family

Return values

0	Success
-1	Failure

Description

The GSSetRFontFace function sets the name of the typeface used for a given family of raster fonts. You can specify a typeface for each of the five different font families and change the typeface for a given family any number of times.

The next call to GSLoadRFont for the given family loads a font of the specified typeface (provided there's one available), ready for use in all subsequent text operations.

When Windows selects an actual fonts, the typeface specification has higher priority than generic family characteristics. You can even override the generic characteristics of a font family by specifying a typeface belonging to another family--for example, by specifying a Times typeface for the Swiss font family.

The parameter table for nFamily shows the default typeface names for the different font families.

Example

The following example sets a typeface for two of the families and then draws some text in the chosen fonts:

```
fWidth = GSGetVXExt();
fHeight = GSGetVYExt();
GSSetRFontFace(FOSWISS, "Bookman Old Style");
GSSetRFontFace(FODECO, "Wide Latin");
```

```
GSLoadRFont(FOSWISS, 0, 250, 0);
GSRText(fWidth / 2, fHeight * .6, CSUSER | CSRASTER,
        TXMID | TXBOTTOM, BLACK, "Hello");
GSLoadRFont(FODECO, FOITALIC, 250, 0);
GSRText(fWidth / 2, fHeight * .3, CSUSER | CSRASTER,
        TXMID | TXBOTTOM, BLACK, "World");
```

J

Topic

GSSetRFontFace

Related

GSRText

GSLoadRFont

GSSetVFontFace

GSSetROP function

Sets raster operation mode

C/C++

```
int GSSetROP( int nROP )
```

FoxPro

```
r = GSSetROP(nROP)
```

Visual Basic

```
r% = GSSetROP(nROP%)
```

Parameters

`nROP`

Constant	Value	Meaning
ROREPLACE	0	Replace
ROOR	1	Logical OR
ROXOR	2	Logical XOR
RONOT	3	Logical NOT (negate)

Return values

0	Success
-1	Failure

Description

The GSSetROP function sets the raster operation mode for the current view.

GSSetVFontFace function

Sets typeface used for vector font family

C/C++

```
int GSSetVFontFace( int nFamily, char szFaceName )
```

FoxPro

```
r = GSSetVFontFace(nFamily, szFaceName)
```

Visual Basic

```
r% = GSSetVFontFace(nFamily%, szFaceName$)
```

Parameters

`nFamily`

Constant	Value	Meaning
FOROMAN	1	Roman
FOSWISS	2	Swiss
FOMODERN	3	Modern
FOSCRIPT	4	Script
FODECO	5	Decorative

`szFaceName` Face name for the family

Return values

0	Success

-1 Failure

Description

The GSSetVFontFace function sets the name of the typeface that is used for a given family of vector fonts. You can specify a typeface for each of the five different font families and change the typeface for a given family any number of times.

The next call to GSLoadVFont for the given family loads a font of the specified typeface (provided there's one available), ready for use in all subsequent text operations.

The typeface specification has higher priority than generic family characteristics when Windows selects an actual font. This has the interesting side effect that you can override the generic characteristics of a font family by specifying a typeface belonging to another family--for example, by specifying a Times typeface for the Swiss font family.

A typical Windows system doesn't have many vector fonts, so the ability of this function to set a typeface is unlikely to be as useful as GSSetRFontFace, which allows selection from the wide variety of TrueType fonts available.

J

Topic

GSSetVFontFace

Related

GSVText

GSLoadVFont

GSSetRFontFace

GSShade function

Shades bounded area

C/C++

```
int GSShade( double fxOrg, double fyOrg, int nPatt,
             int nClr )
```

FoxPro

```
r = GSShade(fxOrg, fyOrg, nPatt, nClr)
```

Visual Basic

```
r% = GSShade(fxOrg#, fyOrg#, nPatt%, nClr%)
```

Parameters

`fxOrg`	X
`fyOrg`	Y
`nPatt`	Pattern (see Pattern constants)
`nClr`	Shade color (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSShade function shades the interior of a region bounded by a continuous line of the same color as the fill pattern.

J

Topic

GSShade

Related

GSDefPatt
GSClearView
GSBox2D

GSSizeSymbol function

Defines size of all symbols

C/C++

```
int GSSizeSymbol( double fDiam )
```

FoxPro

```
r = GSSizeSymbol(fDiam)
```

Visual Basic

```
r% = GSSizeSymbol(fDiam#)
```

Parameters

`fDiam`	Symbol diameter in view units

Return values

0	Success
-1	Failure

Description

The GSSizeSymbol function determines the size of all symbols used in graphs or symbol functions. Symbols are sized in terms of a characteristic dimension that roughly corresponds to its diameter.

By default, the diameter is 2.5% of the height of the view, or 25 units in a view with the default height of 1000.

When a user resizes a window, symbols are resized in a corresponding manner.

J

Topic

GSSizeSymbol

Related

GSSymbol

GSStatsArr function

Defines data for applying statistics

C/C++

```
int GSStatsArr( int nIndex )
```

FoxPro

```
r = GSStatsArr(nIndex)
```

Visual Basic

```
r% = GSStatsArr(nIndex%)
```

Parameters

`nIndex`	Set index in fA array (based on 0)

Return values

0	Success
-1	Failure

Description

The GSStatsArr function selects the data set to which statistics are applied. It's only used when more than one data set is held in the fA array.

For example, a high-low-close graph has three data sets--high (set 0), low (set 1), and close (set 2). If you call GSStats Arr with an nIndex of 2, subsequent graph statistics are based on the close data.

J

Topic

GSStatsArr

Related

GSMean
GSSD
GSLineFit
GSCurveFit
GSHLC
GSBoxWhisker
GSStatsWin

GSStatsWin function

Defines statistics clipping region

C/C++

```
int GSStatsWin( double fxBL, double fyBL, double fxTR,
                double fyTR )
```

FoxPro

```
r = GSStatsWin(fxBL, fyBL, fxTR, fyTR)
```

Visual Basic

```
r% = GSStatsWin(fxBL#, fyBL#, fxTR#, fyTR#)
```

Parameters

`fxBL`	X bottom left
`fyBL`	Y bottom left
`fxTR`	X top right
`fyTR`	Y top right

Return values

0	Success
-1	Failure

Description

The GSStatsWin function defines the window within which statistical lines are clipped. The window is defined in terms of the bottom left and top right in view coordinates. Typically, the window coincides with the axes of a graph.

J

Topic

GSStatsWin

Related

GSStatsArr
GSLineFit
GSMean
GSMinMax
GSSD

GSSymbol function

Draws symbol

C/C++

```
int GSSymbol( double fxOrg, double fyOrg, int nSymbol,
              int nClr )
```

FoxPro

```
r = GSSymbol(fxOrg, fyOrg, nSymbol, nClr)
```

Visual Basic

```
r% = GSSymbol(fxOrg#, fyOrg#, nSymbol%, nClr%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`nSymbol`	Symbol number (see Symbol constants)
`nClr`	Symbol color (see Color constants)

Return values

0	Success
-1	Failure

Description

The GSSymbol function draws a symbol from Graphics Server's internal symbol library.

The origin of every symbol is at its center.

Add the value 256 (SYHOLLOWFILL) to nSymbol to fill hollow symbols with the current window background color.

J

Topic

GSSymbol

Related

GSSizeSymbol

GSTapeGraph function

Draws tape graph

C/C++

```
int GSTapeGraph( double fxOrg, double fyOrg,
                 double fInc, double fDepth,
                 double fAng, int nMode, int nClr )
```

FoxPro

```
r = GSTapeGraph(fxOrg, fyOrg, fInc, fDepth, fAng,
                nMode, nClr)
```

Visual Basic

```
r% = GSTapeGraph(fxOrg#, fyOrg#, fInc#, fDepth#, fAng#,
                 nMode%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fInc` X increment

`fDepth` Perspective depth of graph

`fAng` Perspective angle from the horizontal

`nMode`

Constant	Value	Meaning
TAPEVARX	1	Use fD array for X position

`nClr` Colors of top surfaces of tapes (a one-dimensional array). The bottom surfaces, where visible, are colored with the half-tones of the colors in the nClr array. (See Color constants.)

Return values

0	Success
-1	Failure

Description

The GSTapeGraph function draws a tape graph showing one or more sets of data.

Visually, the tape is similar to a line graph with perspective added to suggest depth. The depth and angle of perspective are governed by the fDepth and fAng parameters. With fAng 0, the tape appears to have no depth and becomes like a line again. With fAng 90, the tape appears to be face-on to the viewer. The best results are obtained with an angle of about 30 degrees.

The data may be graphed either at fixed increments in X, as defined by fInc, or using the individual X values passed in the fD array.

If more than one data set is supplied, the graph acquires Z axis perspective, with successive data sets advancing from the back toward the front of the graph.

Specifying missing-data points

You can use the nAux array--through the GSDataAux function--to flag points of a tape graph as "missing." Missing points aren't shown, whether or not you've provided values for them.

nAux setting	**Value**	**Meaning**
	0	Point shown normally
	256	Point is "missing" and not drawn

The size of the nAux array may be nPts or nPts nGroup. If you set the size to nPts and there's more than one group of data, the same missing points are assumed for all the groups. If the size is nPts nGroup, each point in each group has its own missing-data flag.

GSDataTrans parameters for tape graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions of plotted points)
fD[nPts]	Pointer to distance array (X positions of plotted points)--used only with nMode TAPEVARX
nPatt[0]	Not used
nSymbol[0]	Not used
nAux[nPts]	Pointer to array containing missing-data flag for each point
nClr[0]	Not used

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions of plotted points)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions of plotted points)--used only with nMode TAPEVARX
nPatt[0]	Not used
nSymbol[0]	Not used
nAux[nPts] *or* nAux[nPts][nGroup]*	Pointer to array containing missing-data flag for each point

nClr[nGroup]	Pointer to array containing color for each data set

* GSDataTrans can't pass two-dimensional fD or nAux arrays. You have to use the GSDataDist or GSDataAux function if you want to specify individual fD or nAux values for each data set. However, you can use GSDataTrans if you want to apply the same fD or nAux values to points in all sets.

J

Topic

GSTapeGraph

Related

GSArea3D
GSXYGraph

Axis/cage/legend:

GSCage3D
GSLegend

Labels:

GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:

GSDataTrans
GSDataAux
GSDataDist

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

GSTimeGraph function

Draws scrolling time series graph

C/C++

```
int GSTimeGraph( double fxOrg, double fyOrg,
                 double fInc, int nPts, int nGroup,
                 int nMode )
```

FoxPro

```
r = GSTimeGraph(fxOrg, fyOrg, fInc, nPts, nGroup,
                nMode)
```

Visual Basic

```
r% = GSTimeGraph(fxOrg#, fyOrg#, fInc#, nPts%, nGroup%,
                 nMode%)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	X increment (interval between points). The total length of the display is (nPts-1) fInc
`nPts`	Number of displayed points
`nGroup`	Number of displayed groups
`nMode`	

Constant	Value	Meaning
TIGDEFAULT	0	Symbols
TIGLINES	1	Lines

Return values

0	Success
-1	Failure

Description

The GSTimeGraph function defines a time series graph of one or more sets (groups) of data. This graph differs fundamentally for all others in that GSTimeGraph must be called before any data is stored or displayed. There can only be one time series graph per window, although a single graph can show several concurrent data sets.

Initially no graph is drawn. The graph is built progressively over time by adding data points using the function GSTimeUpdate. The most recent point is drawn at the origin (fxOrg) and previous points are scrolled to the left. When the number of points on display exceeds the maximum, nPts, the oldest, now at the extreme left, is discarded.

One time series graph may display several data sets. The characteristics of each displayed data set are defined in the arrays passed using the normal GS functions. Note that these functions must be called *after* the call to GSTimeGraph and *before* the first call to GSTimeUpdate.

The symbol array defines the symbol drawn at each data point when nMode is TIGDEFAULT. The color array defines the color of the symbol and of any statistical lines. The pattern array defines the line style of the mean statistical line; standard-deviation lines

are drawn using the specified pattern index *plus 1*. The distance array (fD) defines the vertical offset of the data set from the Y origin, enabling sets to be drawn at different vertical positions.

To produce fast animation, GSTimeGraph XORs the color of a symbol with whatever color lies beneath. The color index must be one of the basic colors (0-16). This color is adjusted such that the result of XORing with the current background color produces the color as specified. If the symbols are XORed onto any area not in the background color, the result is unpredictable.

Superimposing statistical lines

You can use the nAux array--through the GSDataAux function--to superimpose the mean and standard deviation of each set. (The normal statistics and curve fitting functions don't apply to time series graphs). These statistical lines are updated automatically as the graph develops.

nAux setting	**Constant**	**Value**	**Meaning**
	TIGMEAN	1	Superimpose mean
	TIGSTDDEV	2	Superimpose standard deviation

GSDataTrans parameters for time series graphs

nPts	Number of points per data set (always 1)
nGroup	Number of data sets (no limit)
fA[0]	Not used
fD[nGroup]	Pointer to distance array (Y offsets from common graph origin)
nPatt[nGroup]	Pointer to array containing line styles or thicknesses for statistical lines
nSymbol[nGroup]	Pointer to array containing symbol design for each data set
nAux[nGroup]	Pointer to array specifying statistical lines to be drawn for each data set
nClr[nGroup]	Pointer to array containing color for each data set

J

Topic
GSTimeGraph

Related
GSTimeUpdate

Axis/grid/legend:
GSAxis
GSGrid

GSLegend

Labels:
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:
GSDataTrans
GSDataAux

Window initialization:
GSCloseWin
GSOpenWin
GSOpenChildWin

GSTimeUpdate function

Updates time series graph

C/C++

```
int GSTimeUpdate( int nMode, int nGroup, double fData )
```

FoxPro

```
r = GSTimeUpdate(nMode, nGroup, fData)
```

Visual Basic

```
r% = GSTimeUpdate(nMode%, nGroup%, fData#(0))
```

Parameters

	Constant	Value	Meaning
`nMode`			
	TIGUDPATE	0	Loads new data and redraws graph
	TIGLOAD	1	Loads new data, but doesn't redraw graph
	TIGHIDE	2	Loads no data and hides graph
	TIGSHOW	3	Loads no data, but redraws graph
`nGroup`	Number of data groups		
`fData`	Array of data for all groups		

Return values

0	Success
-1	Failure

Description

The GSTimeUpdate function updates a time series graph previously created using the GSTimeGraph function.

Under mode 0, the normal method, Graphics Server loads new data and scrolls the graph in a single operation.

Modes 1, 2, and 3 let you perform fast batch updates using a buffer. To perform a batch update, hide the graph (mode 2), add new data (mode 1), then redraw the graph (mode 3).

You can draw a time series graph in three different ways:

- Set the recording mode to BitBlit by calling GSWinPaint(4) before making any calls to draw axes, titles, and so forth. Call GSWinPaint(3) after each call to GSTimeUpdate. Animation appears smooth because the graph is updated immediately on screen, but there's a time penalty in the redraw. The window is automatically repainted after it's covered.
- Set the recording mode to none by calling GSWinPaint(2). If the window is covered, the graph isn't redrawn when it's uncovered. This method produces the fastest animation, but there's some flickering during updates, and you have to intervene to repaint the window.

Set the recording mode to metafile by calling the default mode of GSWinPaint(5) and issue commands to draw the backdrop (axes, titles, and so forth) of the graph. Then, set the recording mode to none by calling GSWinPaint(2) and start the time series graph. If the window is uncovered, the backdrop is redrawn automatically, but the application has to manually repaint the graph using GSTimeUpdate(2), then set the recording mode to none by calling GSWinPaint(2).

> You should never actually draw the graph using the default mode, GSWinPaint(5), because all of the drawing commands are saved in the metafile over the entire time the graph is active.

J

Topic

GSTimeUpdate

Related

GSTimeGraph

GSWinPaint

Array initialization:

GSDataTrans

GSUseView function

Uses view

C/C++

```
int GSUseView( int nWin, int nView )
```

FoxPro

```
r = GSUseView(nWin, nView)
```

Visual Basic

```
r% = GSUseView(nWin%, nView%)
```

Parameters

`nWin`	Window number
`nView`	View number

Return values

0	Success
-1	Failure

Description

The GSUseView function selects a window and a view to draw in. The view remains current until the next GSUseView call or until the view is closed.

You can use GSUseView at any time to switch between views.

J

Topic

GSUseView

Related

GSHotGraph
GSMStatus
GSMGetX
GSMGetY
GSMNotify
GSWinNotify
GSWinPaint

View functions:

GSClearView
GSCloseView
GSGetVXExt
GSGetVYExt
GSOffView
GSOnView
GSOpenView

GSViewClip function

Applies a clipping window within the current view

C/C++

```
int GSViewClip(double fxOrg, double fyOrg,
               double fWidth, double fHt)
```

FoxPro

```
r = GSViewClip(fxOrg, fyOrg, fWidth, fHt)
```

Visual Basic

```
r% = GSViewClip(fxOrg#, fyOrg#, fWidth#, fHt#)
```

Parameters

`fxOrg`	X bottom left
`fyOrg`	Y bottom left
`fWidth`	Width of clipping window
`fHt`	Height of clipping window

Return values

`0`	Success
`-1`	Failure

Description

GSViewClip applies a clipping window within the current view, defined by the lower left corner and the width and height.

Topic

GSViewClip

Related

GSOpenView

GSUseView

GSVText function

Draws vector text

C/C++

```
int GSVText( double fxOrg, double fyOrg, double fWid,
             double fHt, double fAng, int nCSet,
             int nTMode, int nClr, char szString )
```

FoxPro

```
r = GSVText(fxOrg, fyOrg, fWid, fHt, fAng, nCSet,
            nTMode, nClr, szString)
```

Visual Basic

```
r% = GSVText(fxOrg#, fyOrg#, fWid#, fHt#, fAng#,
             nCSet%, nTMode%, nClr%, szString$)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fWid`	Width of text box
`fHt`	Height of text box
`fAng`	Angle of text in degrees (0 is horizontal; angles increase counterclockwise)
`nCSet`	Character set--selects between the system font and a user font loaded with the GSLoadVFont function (see Character set constants)
`nTMode`	Text mode (see Text mode constants)
`nClr`	Text color (see Color constants)
`szString`	Text string

Return values

0	Success
-1	Failure

Description

The GSVText function draws a line of vector text. Characters are scaled so that the line fits in a rectangle of width fWid and height fHt.

By default, Graphics Server achieves a rough fit using the average width of characters. You can get an exact fit by including TXEXACT in the nTMode parameter, at a slight cost of drawing speed.

The alignment of the text with the origin (fxOrg,fyOrg) is determined by the nTMode parameter.

J

Topic

GSVText

Related

GSLoadVFont

GSSetVFontFace

GSWinHandle function

Returns Windows handle of graphing window

C/C++

```
HWND GSWinHandle( int nWindow )
```

FoxPro

```
r = GSWinHandle(nWindow)
```

Visual Basic

```
r% = GSWinHandle(nWindow%)
```

Parameters

`nWindow`	Graphing window number

Return values

1 or greater	Windows handle of the graphing window
-1	Failure

Description

The GSWinHandle function returns the Windows handle of a graphing window that your application has opened using the GSOpenWin function.

When you open a new graphing window with GSOpenWin, Graphics Server returns a logical window number that identifies the window in the scope of your application. The logical window number has no significance in the Windows environment as a whole. Your application may find it useful to have the Windows handle identifying a graphing window so the window can be operated on globally, using functions in the Windows API.

J

Topic

GSWinHandle

Related

GSOpenWin
GSOpenChildWin
GSWinNotify

GSWinNotify function

Enables and disables notification of graphing window events

C/C++

```
int GSWinNotify( HWND hWnd, int nWM, int nEvents )
```

FoxPro

```
r = GSWinNotify(hWnd, nWM, nEvents
```

Visual Basic

```
r% = GSWinNotify(hWnd%, nWM%, nEvents%)
```

Parameters

`hWnd` Client notification window handle

`nWM` Client window message number

`nEvents`

Constant	Value	Meaning
WNPAINT	1	Window client area needs repainting
WNSIZE	2	Window has changed size

Return values

0	Success
-1	Failure

Description

The GSWinNotify function enables or disables asynchronous notification of graphing window events. Each graphing window can have different event notifications in force.

GSWinNotify always acts on the current window. Use the GSUseView function to change the current window.

The hWnd parameter specifies the Windows handle of the window to which event messages should be posted. The nWM parameter specifies the window message number to be used. This number should be uniquely identified within the window procedure of the notified window.

A graphing window event message is posted in the following form:

Parameter	Type	Meaning
hWnd	HWND	Windows handle of the notified window as originally specified to GSWinNotify
uMsg	UINT	Windows message number as originally specified to GSWinNotify
wParam	UINT	The type of window event, which is *one* of the nEvents settings originally specified to GSWinNotify. Although events may be enabled together, they are always notified individually.
lParam	LONG	Event qualifying information (currently not used)

Example

The following example illustrates how to open a graphing window and enable it for window event notification.

```
#define WM_WINDOWEVENT  (WM_USER + 1)
double ScreenWid, ScreenHt;
int WinNum;
ScreenWid = GSGetSXExt();
ScreenHt = GSGetSYExt();
WinNum = GSOpenWin( 0.10 * ScreenWid,
                    0.10 * ScreenHt,
                    0.50 * ScreenWid,
                    0.50 * ScreenHt,
                    1000, 0, OWMFIXED,
                    "Graphing window" );
if ( WinNum < 0 ) {
   /* GSOpenWin failed */
}
/* enable window event notification */
GSWinNotify( hWndHdlr, WM_WINDOWEVENT,
             WNPAINT | WNSIZE );
```

The window procedure for the window hWndHdlr might contain this code:

```
LONG WndHdlr_WndProc( HWND hWnd, UINT uMsg,
                      UINT wParam, LONG lParam )
{
switch ( uMsg ) {
case WM_WINDOWEVENT:
   /* the user caused a window event so see
   what type it is */
   switch ( wParam ) {
case WNPAINT:
   /* a repainting event etc. */
case WNSIZE:
   /* a resizing event etc. */
}
}
}
```

J

Topic

GSWinNotify

Related

GSOpenWin
GSOpenChildWin
GSWinHandle
GSUseView

GSWinPaint function

Sets graphing window painting mode

C/C++

```
int GSWinPaint( int nMode )
```

FoxPro

```
r = GSWinPaint(nMode)
```

Visual Basic

```
r% = GSWinPaint(nMode%)
```

Parameter

nMode	**Constant**	**Value**	**Meaning**
Recording modes	WPMETAFILE	5	Graphics Server records all the drawing functions in metafile format. The graphing window is automatically updated by individual drawing functions. This is the default recording mode.
	WPBITMAP	4	Graphics Server records all the drawing functions in a bitmap. The graphing window isn't automatically updated by individual drawing functions.
	WPNONE	2	Graphics Server sends output directly to the graphing window and doesn't record the drawing functions. If the window needs repainting, your application must arrange to do this by recalling the original drawing functions.
Repainting modes	WPAUTO	0	Graphics Server automatically repaints the graphing window from the bitmap or metafile. This is the default repainting mode.
	WPMANUAL	1	Graphics Server doesn't repaint the graphing window automatically. Your application can repaint the window by calling GSWinPaint with mode WPPAINT for the metafile or bitmap or, if the recording mode is WPNONE, by recalling the original drawing functions.
Additional	WPPAINT	3	Graphics Server

mode	immediately repaints the graphing window from the metafile or bitmap. If the recording mode is WPNONE, this mode has no effect.

Return values

0	Success
-1	Failure

Description

The GSWinPaint function sets the painting mode for a graphing window. All of the nMode options are exclusive.

Graphics Server normally keeps a record of all the drawing functions performed in a graphing window so that when the window is uncovered or resized by the user, Graphics Server can repaint the contents of the window without requiring your application to repeat the original drawing functions.

Graphics Server uses two different recording modes to keep track of drawing functions: bitmap mode and metafile mode.

In bitmap mode, a physical copy of the display area of the graphing window is kept in off-screen memory and is used to record the drawing functions in a device-dependent form. Calls to the drawing functions update the bitmap without automatically updating the window.

At an appropriate time after your application has caused a complete image to be drawn off-screen, it can request Graphics Server to copy the whole bitmap into the window in one operation called a BitBLt.

An advantage of bitmap mode is that it can be used to hide the stages of drawing from the user and give the impression of a much more immediate presentation of a new image. A disadvantage is that the bitmap isn't re-scalable nor readily adaptable to other types of output device such as the printer.

Graphics Server currently won't print a window when bitmap recording mode is set. Graphics Server uses bitmap stretching techniques to resize the bitmap to suit a changing window size, but this doesn't always produce satisfactory results and considerably slows down the repainting speed.

In metafile mode, Graphics Server keeps a logical record of the drawing functions in a Windows metafile. Calls to the drawing functions simultaneously update the metafile and the window. Graphics Server replays the whole metafile into the window whenever the window needs repainting.

The metafile is a cumulative recording mode, which means that new drawing functions don't overwrite existing ones, but are simply added to the end. You need to call GSClearView at regular intervals to clear the metafile and prepare it for drawing a new image.

An advantage of the metafile mode is that the recorded image is completely rescalable and adaptable to any printer or other output device. A slight disadvantage is that the user can

perceive the stages of drawing, making the metafile mode appear slightly less "immediate" than bitmap mode. In fact, drawing takes about the same length of time in either mode.

Metafile recording mode is the default for a newly opened graphing window.

Each graphing window has its own painting mode. GSWinPaint always acts on the current window. Use GSUseView to change the current window.

J

Topic

GSWinPaint

Related

GSOpenWin
GSOpenChildWin
GSGetMF
GSClearView
GSUseView
GSTimeUpdate

GSWriteRegionFile function

Creates an image map for use in an HTML page

C/C++

```
int GSWriteRegionFile(int nMode,
        char* lpstrFile, char* lpstrTemplate,
        char* lpstrPolySpec, char* lpstrRectSpec,
        int nRefStrs, char *lpstrRefStrs[])
```

FoxPro

```
r = GSWriteRegionFile(nMode,
        @szFile, @szTemplate,
        @szPolySpec, @szRectSpec,
        nRefStrs, @szRefStrs(1))
```

Visual Basic

```
r% = GSWriteRegionFile(nMode%,
        szFile$, szTemplate$,
        szPolySpec$, szRectSpec$,
        nRefStrs%, szRefStrs$(0))
```

Parameters

`nMode`

Constant	Value	Meaning
	0	Create szFile
REGIONFILEAPPEND	1	Append szFile

`szFile` Path (optional), file name and file extension. If you don't specify a path, the file is written to the current directory.

`szTemplate` String specifying the form in which definitions for hot regions are written.

`szPolySpec` The term used in the map to specify a polygon. The default is POLYGON.

`szRectSpec` The term used in the map to specify a rectangle. The default is RECT.

`nRefStrs` The number of elements in the szRefStrs array. If the array is omitted, pass 0. Otherwise, pass nGroups * nPoints.

`szRefStrs` (Optional) Array of reference strings, in URL form, to be substituted in hot spot definitions. The map will have one hot region for each data point in every data set. Include one link for each hot region.

Return values

0	Success
-1	Failure

Description

The GSWriteRegionFile function creates a file with hot region definitions that can be used as an image map referenced in an HTML document.

GSWriteRegionFile() uses Graphics Server's hot-graphing feature to map hot regions for each point on the displayed graph. Before calling the function, you must display a graph and turn hot-graphing on. To save the image to which the map applies, call

GSPicWrite().

Standard image maps

Server maps. A server-side image map is stored in a special text file external to the HTML document that references it. The convention is to use the same base file name for both the image and the map. Often the map file's extension is .MAP, though other extensions are usually acceptable.

All web browsers that support HTML 2.0 or higher support server maps. The format for the map will depend on your web server.

The two most common formats are NCSA (National Center for Supercomputing Applications) and CERN (European Laboratory for Particle Physics). Each of these uses a slightly different way to define a hot region:

```
# NCSA image map
shape url x1,y1 x2,y2. . .
. . .

# CERN image map
shape (x1,y1) (x2,y2). . . url
. . .
```

Server maps often require that URLs be fully qualified. Consult your server documentation.

Client maps. A client-side image map is stored within the HTML document that references it. The setting for MapFile should be the file name and extension of the HTML document.

All web browsers that support HTML 3.0 or higher also support client maps. The format for the map is defined by the HTML specification:

```
<! Client image map >
<MAP NAME=mapname
<AREA SHAPE=shape COORDS=x1,y1 x2,y2. . . HREF=url>
. . .
</MAP>
```

Client maps can use either partial URLs (bar.htm) or fully qualified URLs (http://www.foo.com/bar.htm).

Note that GSWriteRegionFile() creates only a list of hot spot definitions (<AREA . . . >). Your program will need to frame them with MAP tags.

Constructing a template

The template string (szTemplate) controls the appearance of each entry GSWriteRegionFile() writes into the image map. It consists of a framework for the entry plus symbols denoting the places where the function substitutes variable information for individual entries.

The items of variable information that can be substituted are

listed below.

Symbol	Substitutes
%1	One-based data set number of the region.
%2	One-based point number of the region.
%3	Term for the shape of the region. Taken from szPolySpec or szRectSpec, whichever is applicable.
%4	List of vertices outlining the region. Each vertex is an x,y coordinate pair.
%5	Reference string (URL) for this region. Taken from the element of the szRefStrs array corresponding to the set and point.
%%	Deferred parameter substitution. One percent sign is written to the file so that your program can post-process it, substituting whatever text you choose.

For example, the following template, in C string-literal form, might be used to format the entries for a client-side image map:

```
<AREA SHAPE=\"%3\" COORDS=\"%4\" HREF=\"P%1-%2.htm\">\r\n
```

Note that "\" escape sequences are necessary to include quotation marks within the string and the "\r\n" escape sequence adds a carriage-return, line-feed at the end of the entry.

If the graph is a bar chart with one data set of five point, this template could result in the following output:

```
<AREA SHAPE="RECT" COORDS="209,151 243,220"
HREF="P1-5.htm">
<AREA SHAPE="RECT" COORDS="171,82 205,220"
HREF="P1-4.htm">
<AREA SHAPE="RECT" COORDS="134,123 167,220"
HREF="P1-3.htm">
<AREA SHAPE="RECT" COORDS="96,152 130,220"
HREF="P1-2.htm">
<AREA SHAPE="RECT" COORDS="59,96 92,220"
HREF="P1-1.htm">
```

In this case reference strings are automatically generated using parameter substitution:

```
. . .HREF=\"P%1-%2.htm\". . .
```

If the szRefStrs array is passed to GSWriteRegionFile(), the strings could be generated using parameter substitution:

```
. . .HREF=\"%5\". . .
```

Or the reference strings can be added by your program by using deferred substitution. The template would read:

```
. . .HREF=\"%%5.htm\". . .
```

And the output would be:

```
. . .HREF="%5.htm". . .
```

Your program could then read the output line and make the substitution for %5 using the Windows API function FormatMessage.

You can specify the format of a substitution by immediately following it with a printf-style format specification in angle brackets "<>". For example, the following template specification might be used to output the shape-specifier in a 12-character, fixed-width field and the vertex list in space-separated, rather than comma-separated, x, y form:

```
%3<%12s> P%1-%2.htm %4<%d %d>\r\n
```

This is an advanced feature you should only use for very specific formatting requirements. To use it requires that you understand the C print formatting system. Also note that you can control the format but not the underlying type of the substitution variables. The types and default formats are listed below.

Parameter	Substitutes	Type	Default format
%1	Set	integer	%d
%2	Point	integer	%d
%3	Shape	string	%s
%4	Coordinates	integer	%d,%d
%5	URL	string	%s

Examples

This example, in Visual Basic, generates an HTML document with an image of the current graph and a client-side image map.

```
Private Sub Command1_Click()
'Write an image of the graph
r& = GSPicWrite(0, 0, 0, 0, 7, 0, "clientmap.gif")

'Prepare the HTML document
Open "clientmap.htm" For Output Access Write As #1
Print #1, "<HTML>"
Print #1, "<HEAD>"
Print #1, "<TITLE>Client map</TITLE>"
Print #1, "</HEAD>"
Print #1, "<BODY>"
Print #1, "<IMG SRC=" + _
    Chr$(34) + "clientmap.gif" + Chr$(34) + _
    " USEMAP=" + Chr$(34) + "#graph" + Chr$(34) + ">"
Print #1, "<MAP NAME=" + Chr$(34) + "graph" + _
    Chr$(34) + ">"
Close #1

'Write the hot spot definitions
'Note that nMode = 1 (Append)
```

```
r& = GSWriteRegionFile(1, "clientmap.htm", _
    "<AREA SHAPE=" + Chr$(34) + "%3" + Chr$(34) + _
    " COORDS=" + Chr$(34) + "%4" + Chr$(34) + _
    " HREF=" + Chr$(34) + "RGN%1-%2.HTM" + Chr$(34) + _
    ">" + Chr$(13) + Chr$(10), "POLYGON", "RECT")

'Finish the HTML document
Open "clientmap.htm" For Append Access Write As #1
Print #1, "</MAP>"
Print #1, "</BODY>"
Print #1, "</HTML>"
Close #1
End Sub
```

Depending on the graph, the resulting HTML document could look like this:

```
<HTML>
<HEAD>
<TITLE>Client map</TITLE>
</HEAD>
<BODY>
<IMG SRC="clientmap.gif" USEMAP="#graph">
<MAP NAME="graph">
<AREA SHAPE="POLYGON" COORDS="309,46 389,46 389,306
  309,306" HREF="RGN1-3.HTM">
<AREA SHAPE="POLYGON" COORDS="208,306 208,163 288,163
  297,166 297,279 288,306" HREF="RGN1-2.HTM">
<AREA SHAPE="POLYGON" COORDS="107,306 107,195 153,191
  217,191 217,279 188,306" HREF="RGN1-1.HTM">
<AREA SHAPE="POLYGON" COORDS="393,279 393,136 409,126
  490,126 490,306 409,306" HREF="RGN1-4.HTM">
<AREA SHAPE="POLYGON" COORDS="473,279 473,147 510,139
  591,139 591,306 510,306" HREF="RGN1-5.HTM">
</MAP>
</BODY>
</HTML>
```

The next example shows part of a C++ application preparing a server-side image map for a JPEG image and creating a small HTML document that references them.

```
GSPicWrite( 0, 0, 0, 0, PXJPEG, 0, "graph.jpg" );
GSWriteRegionFile( 0, "graph.map", "%3 RGN%1-%2.HTM %4\r\n",
"POLY", "RECT" );

FILE* pFile = fopen( "graph.htm", "wt" );

if ( pFile != NULL ) {
     fputs( "<HTML>", pFile );
     fputs( "<HEAD>", pFile );
     fputs( "<TITLE>Server map</TITLE>", pFile );
```

```
        fputs( "</HEAD>", pFile );
        fputs( "<BODY>", pFile );
        fputs( "<A HREF=\"graph.map\">", pFile );
        fputs( "<IMG SRC=\"graph.jpg\" ISMAP>", pFile );
        fputs( "</A>", pFile );
        fputs( "</BODY>", pFile );
        fputs( "</HTML>", pFile );

        fclose( pFile );
}
```

Topic
GSWriteRegionFile

Related
GSHotGraph
GSPicWrite

GSXDataScale function

Applies scale factor to distance data

C/C++ `int GSXDataScale( double fScale )`

FoxPro `r = GSXDataScale(fScale)`

Visual Basic `r% = GSXDataScale(fScale#)`

Parameter `fScale` Data scale factor

Return values

0	Success
-1	Failure

Description

The GSXDataScale function scales distance data used in any of the graph or chart functions. Data in the distance (fD) array is multiplied by fScale before graphing.

The default factor of unity is reset whenever new data is transferred.

J

Topic

GSXDataScale

Related

GSDataDist
GSDataGetDist
GSDataGetDistErr
GSDataStoreDist
GSDataScale

Array initialization:

GSDataAmp
GSDataAux
GSDataClr
GSDataPatt
GSDataSym
GSDataTrans
GSDataZ

GSXYGraph function

Draws line graph

C/C++

```
int GSXYGraph( double fxOrg, double fyOrg, double fInc,
               int nMode, int nClr )
```

FoxPro

```
r = GSXYGraph(fxOrg, fyOrg, fInc, nMode, nClr)
```

Visual Basic

```
r% = GSXYGraph(fxOrg#, fyOrg#, fInc#, nMode%, nClr%)
```

Parameters

`fxOrg` X origin

`fyOrg` Y origin

`fInc` X increment

`nMode`

Constant	Value	Meaning
XYGLINE	1	Connects points with lines
XYGSYMBOL	2	Draws symbols at points
XYGSTICK	4	Draws vertical sticks to points
XYGVARX	8	Uses fD array for X positions
XYGTHICK	16	Uses thick lines
XYGPATT	32	Uses patterned lines
XYGGROUPED	64	Multiple data-set mode

`nClr` Color of markers if you're graphing only one data set. With multiple data sets, you have to create a color array. (See Color constants.)

Return values

0 Success

-1 Failure

Description

The GSXYGraph function draws a line graph. The graph can include lines, symbols, vertical sticks, or any combination of these, according to the nMode parameter.

If you use symbols (nMode XYGSYMBOL), you have to load an array with values for the desired symbols and pass a pointer to this array in a call to the GSDataTrans function. A similar procedure is needed if you use patterned (nMode XYGPATT) or thick (nMode XYGTHICK) lines--load an array with values for line patterns or thicknesses, then pass a pointer to the array using GSDataTrans.

In symbol (XYGSYMBOL) mode, the array is presumed to contain a series of symbol values in the range 0-11. In patterned

(XYGPATT) mode, the array is presumed to contain a series of line style values specified from the set LSSOLID, LSDOT, and so forth. In thick-lines (XYGTHICK) mode, the values are presumed to specify the approximate thicknesses of the lines in pixel units.

The number of elements in an array of symbols, patterns, or thicknesses will depend on the number of data sets in your graph. With a single data set, the values in the appropriate arrays are applied on a per-point basis--you can use a symbol or line style to differentiate each point. In that case, the number of elements in the array must equal the number of points in the graph. With multiple data sets, the arrays are applied on a per-set basis so you can differentiate data sets on the graph. In that case, the number of elements in the array must equal the number of sets.

You can graph data either at fixed increments in X as defined by fInc (the default mode) or using individual X values for points (nMode XYGVARX). In the latter case, you use the fD (distance) array to specify the X positions.

GSDataTrans parameters for XY graphs

One data set

nPts	Number of points in data set (no limit)
nGroup	Number of data sets (1)
fA[nPts]	Pointer to amplitude array (Y positions of plotted points)
fD[nPts]	Pointer to distance array (X positions of plotted points)--used only with nMode XYGVARX
nPatt[nPts]	Pointer to array containing line style or thickness for each line element--used only with nMode XYGTHICK or XYGPATT
nSymbol[nPts]	Pointer to array containing symbol design for each point
nAux[nPts]	Pointer to array containing missing-data flag for each point
nClr[0]	Not used

Multiple data sets

nPts	Number of points per data set (no limit)
nGroup	Number of data sets (no limit)
fA[nPts][nGroup]	Pointer to amplitude array (Y positions of plotted points)
fD[nPts] *or* fD[nPts][nGroup]*	Pointer to distance array (X positions of plotted points)--used only with nMode XYGVARX

nPatt[nGroup]	Pointer to array containing line style or thickness for each data set--used only with nMode XYGTHICK or XYGPATT
nSymbol[nGroup]	Pointer to array containing symbol design for each data set
nAux[nPts] *or* nAux[nPts][nGroup]*	Pointer to array containing missing-data flag for each point
nClr[nGroup]	Pointer to array containing color for each data set

* GSDataTrans can't pass two-dimensional fD or nAux arrays. You have to use the GSDataDist or GSDataAux function if you want to specify individual fD or nAux values for each data set. However, you can use GSDataTrans if you want to apply the same fD or nAux values to points in all sets.

J

Topic

GSXYGraph

Related

GSLinLog
GSLogLin
GSLogLog
GSScatter
GSTapeGraph

Axis/grid/legend:

GSAxis
GSGrid
GSLegend

Labels:

GSDataLabels
GSLabelnX
GSLabelX
GSLabelnY
GSLabelY

Array initialization:

GSDataTrans
GSDataDist
GSDataAux

Window initialization:

GSCloseWin
GSOpenWin
GSOpenChildWin

VBAGDataLabels function

Enables and sets text for data labels in Visual Basic

C/C++ Use AGDataLabels function

FoxPro Use AGDataLabels function

Visual Basic

```
r% = VBAGDataLabels(nMode%, nLabs%, sLabelString$)
```

Parameters

`nMode`

Constant	Value	Meaning
AGDLTEXT	0	Labels supplied in sLabelString$
AGDLDATA	1	Labels derived from data
AGDLGROUPCLR	4	Color as data group

`nLabs`

Value	Meaning
0	Use if deriving labels from data (nMode AGDLDATA)
1 or greater	Use for number of labels if supplying text labels (nMode AGDLTEXT) The label array must be of size nPts nGroup to provide text labels for each data item on display. The exceptions are high-low-close, open-high-low-close, candlestick, and box-whisker graphs, which require a text array of size nPts (only one label is provided for each compound symbol, of which there are nPts).

`sLabelString` String of data labels delimited by tabs, Chr$(9)

Note: If nMode is AGDLDATA, pass " " as the label string.

Return values

0	Success
-1	Failure

Description

The VBAGDataLabels function, a Visual Basic-specific version of the AGDataLabels function, enables data labels, which are labels--either numeric or text--attached to each point of a graph. Data labels are available for all 2D graph types except pie charts (which have their own labeling scheme) and time series graphs.

They aren't available for 3D graphs.

Visual Basic doesn't let you pass arrays of text to API functions, so VBAGDataLabels, unlike AGDataLabels, requires you to create a single string containing the text for all data labels. You use the tab character, Chr$(9), to separate label entries within this string.

You can use Visual Basic's Format$ functions to show labels in currency, percent, date, and scientific forms.

In high-low-close, open-high-low-close, box-whisker, and candlestick graphs, if you choose to have data labels derived from data (nMode AGDLDATA), they're derived from the close or median.

J

Topic

VBAGDataLabels

Related

AGDataLabels
AGFontStyle
VBAGLabelY
VBAGLabelZ
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

VBAGLabels function

Defines labels for axis or pie chart in Visual Basic

C/C++ Use AGLabels function

FoxPro Use AGLabels function

Visual Basic

```
r% = VBAGLabels(nLabs%, szLabs$)
```

Parameters

`nLabs`	Number of labels
`szLabs`	String of labels delimited by tabs, Chr$(9)

Return values

0	Success
-1	Failure

Description

The VBAGLabels function, a Visual Basic-specific version of the AGLabels function, transfers an array of labels to label the axis of the independent variable of a graph or the slices of a pie chart.

Visual Basic doesn't let you pass arrays of text to API functions, so VBAGLabels, unlike AGLabels, requires you to create a single string containing the text for all labels. You use the tab character, Chr$(9), to separate label entries within this string.

J

Topic

VBAGLabels

Related

AGFontStyle
AGLabels
VBAGDataLabels
VBAGLabelY
VBAGLabelZ
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

VBAGLabelY function

Defines labels for left or right Y axis in Visual Basic

C/C++ Use AGLabelY function

FoxPro Use AGLabelY function

Visual Basic

```
r% = VBAGLabelY(nSelect%, nNLabs%, szString$)
```

Parameters

	Constant	Value	Meaning
`nSelect`	AGLABYLEFT	0	Sets left-hand Y axis labels
	AGLABYRIGHT	1	Sets right-hand Y axis labels
`nNLabs`	Number of labels		
`szString`	String of labels delimited by tabs, Chr$(9)		

Return values

0	Success
-1	Failure

Description

The VBAGLabelY function, a Visual Basic-specific version of the AGLabelY function, transfers an array of labels for the Y axis. By default, the Y axis is labeled with numeric values according to the axis scale, which is either calculated automatically or set by the AGYAxisStyle function. This function allows arbitrary text labels to replace the numeric values.

The AGYAxisStyle function must be called to set the number of ticks on the axis and hence the number of labels to be supplied in the array.

The nSelect parameter selects between the left- and right-hand Y axes. The latter is only of relevance to combination graphs with a second Y axis drawn to a different scale.

Note that it's possible in graphs with a *single* Y axis to position that axis on the right, using the AGYAxisStyle function. However, this function still treats the axis as a left axis, and you should use nSelect = 0.

Visual Basic doesn't let you pass arrays of text to API functions, so VBAGLabelY, unlike AGLabelY, requires you to create a single string containing the text for all labels. You use the tab character, Chr$(9), to separate label entries within this string.

J

Topic

VBAGLabelY

Related

AGFontStyle
AGLabelY
VBAGDataLabels
VBAGLabels
VBAGLabelZ
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

VBAGLabelZ function

Defines labels for Z axis in Visual Basic

C/C++ Use AGLabelZ function

FoxPro Use AGLabelZ function

Visual Basic

```
r% = VBAGLabelZ(nMode%, nNLabs%, szString$)
```

Parameters

`nMode`	**Value**	**Meaning**
	0	Currently no modes implemented
`nNLabs`	Number of labels	
`szString`	String of labels delimited by tabs, Chr$(9)	

Return values

0	Success
-1	Failure

Description

The VBAGLabelZ function, a Visual Basic-specific version of the AGLabelZ function, transfers an array of labels for the Z axis in True3D graphs. By default, this axis either carries no labels (for all True3D graph types except scatter) or is labeled with numeric values (for True3D scatter graphs). AGLabelZ lets you specify text labels to override these defaults.

The number of labels you need depends on the graph type:

- For True3D area (stacked style) and bar (simple, stacked, or clustered style) graphs, you need only one label.
- *For True3D area (absolute style), bar (z-clustered style), surface, and tape graphs,* you need one label for each data group. The groups are always drawn from back to front, and label array follows that order.
- *For True3D scatter graphs,* Z data values are provided in the Z data array, with the origin at the front. In this case, the Z axis is either drawn to a scale calculated automatically from the data or as specified in the AGZAxisStyle function. If you want to supply text labels, be sure to use AGZAxisStyle to set the number of ticks (and hence the number of labels) for the axis.

Visual Basic doesn't let you pass arrays of text to API functions, so VBAGLabelZ, unlike AGLabelZ, requires you to create a single string containing the text for all labels. You use the tab character, Chr$(9), to separate label entries within this string.

Topic

VBAGLabelZ

Related

AGFontStyle
AGZAxisStyle
VBAGDataLabels
VBAGLabels
VBAGLabelY
VBAGLegend

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

VBAGLegend function

Defines legend labels for grouped data in Visual Basic

C/C++ Use AGLegend function

FoxPro Use AGLegend function

Visual Basic

```
r% = VBAGLegend(nLegs%, szString$)
```

Parameters

`nLegs`	Number of legend labels
`szString`	String of legend labels delimited by tabs, Chr$(9)

Return values

0	Success
-1	Failure

Description

The VBAGLegend function, a Visual Basic-specific version of the AGLegend function, transfers an array of labels for the graph legend.

Visual Basic doesn't let you pass arrays of text to API functions, so VBAGLegend, unlike AGLegend, requires you to create a single string containing the text for all labels. You use the tab character, Chr$(9), to separate label entries within this string.

J

Topic

VBAGLegend

Related

AGFontStyle
AGLegend
AGLegendStyle
VBAGDataLabels
VBAGLabels

Window initialization:

GSOpenWin
GSOpenChildWin

Graph display:

AGOpen
AGShow
AGClose

VBGSDataLabels function

Enables and sets text for data labels in Visual Basic

C/C++ Use GSDataLabels function

FoxPro Use GSDataLabels function

Visual Basic

```
r% = VBGSDataLabels(nMode%, nPrec%, nCSet%, nTMode%,
                    nClr%, fDataOffset#, nLabs%,
                    szLabels$)
```

Parameters

`nMode`

Constant	Value	Meaning
DLTEXT	0	Labels supplied in array szLabels
DLDATA	1	Labels derived from data
DLGROUPCLR	4	Colored as data group

DLGROUPCLR overrides nClr to set the color of labels to the color of the associated data group, except in graph types where the label would overprint block color and be invisible (such as bubble graphs and Gantt charts). In those cases, the DLGROUPCLR flag is ignored and the label is always rendered in nClr.

`nPrec`

Value	Meaning
0 or greater	Specific decimal precision (use 0 if you supply text labels)
-1	Precision is automatically calculated:

- If all values in array are integers from 0 to 999,999, numbers are represented in full
- *If all values in array are fractional,* each number is represented with three-digit precision
- *For arrays containing mixed values,* all numbers are scaled to the closest power of 1000 and represented with three-digit precision (for example, 3,456,000 is shown as 3.45)

`nCSet` Character set (see Character set constants)

`nTMode` Text mode (see Text mode constants)

`nClr` Color of data labels (see Color constants)

`fDataOffset` Number to be subtracted from the data values to compensate for a nonzero origin (numeric labels only; ignored for text labels)

`nLabs`

Value	Meaning
0	Use if deriving labels from data (nMode DLDATA)
1 or greater	Use for number of labels if supplying text labels (nMode DLTEXT) You need a number of labels equal to nPts nGroup to provide text labels for each data item on display. The exceptions are high-low-close, open-high-low-close, candlestick, and box-whisker graphs, which require a text array of size nPts (only one label is provided for each compound symbol, of which there are nPts).

`szLabels` String of data labels delimited by tabs, Chr$(9)

Note: If nMode is DLDATA, pass " " as the label string.

Return values

0	Success
-1	Failure

Description

The VBGSDataLabels function, a Visual Basic-specific version of the GSDataLabels function, enables data labels, which are labels--either numeric or text--attached to each point of a graph. Data labels are available for all 2D graph types except pie charts (which have their own labeling scheme) and time series graphs. They aren't available for 3D graphs.

Visual Basic doesn't let you pass arrays of text to API functions, so VBGSDataLabels, unlike GSDataLabels, requires you to create a single string containing the text for all data labels. You use the tab character, Chr$(9), to separate label entries within this string.

You can use Visual Basic's Format$ functions to show labels in currency, percent, date, and scientific forms.

In high-low-close, open-high-low-close, box-whisker, and candlestick graphs, if you choose to have data labels derived from data (nMode DLDATA), they're derived from the close or median.

You have to call VBGSDataLabels before you call the graphing function, such as GSBar2D, because labels are drawn at the same time as the graph itself.

J

Topic
VBGSDataLabels

Related
GSDataLabels

VBGSLabelPie function

Draws pie chart text labels in Visual Basic

C/C++ Use GSLabelPie function

FoxPro Use GSLabelPie function

Visual Basic

```
r% = VBGSLabelPie(fxOff#, fRad#, fWid#, fHt#, nLabs%,
                  nMode%, nCSet%, nTMode%, nClr%,
                  szLabs$)
```

Parameters

`fxOff`	Horizontal offset
`fRad`	Radius of the arc on which the labels are drawn. This radius must be at least 1.1 times greater than the pie radius (or 1.35 times greater if any segments are exploded). The pie radius is taken from the preceding GSPieChart function call.
`fWid`	Width of label
`fHt`	Height of label
`nLabs`	Number of labels in array
`nMode`	

Constant	Value	Meaning
LPSEGCLR	1	Colors labels the same as segments
LPNOLINES	2	Omits pointing lines from pie to labels

`nCSet`	Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.
`nTMode`	Text mode (see Text mode constants)
`nClr`	Text color (see Color constants)
`szLabs`	String of labels delimited by tabs, Chr$(9)

Return values

0	Success
-1	Failure

Description

The VBGSLabelPie function, a Visual Basic-specific version of the GSLabelPie function, draws a sequence of text labels to complement a pie chart. The pie chart must be drawn first because this function adopts certain parameters from the preceding pie chart function call.

Visual Basic doesn't let you pass arrays of text to API functions, so VBGSLabelPie, unlike GSLabelPie, requires you to create a single

string containing the text for all labels. You use the tab character, Chr$(9), to separate label entries within this string.

There must be the same number of labels as pie slices. The angular position of each label is calculated from the data for the pie chart.

The labels are drawn in an arc on each side of the pie, connected to their respective segments by pointing lines. These lines are drawn from the label horizontally a distance fXOff, then radially toward the center of the pie in a direction bisecting the segment.

By default, this function chooses a vector character set and each label is drawn to fit a rectangle fWid wide and fHt high. Since vector text is shaped to fit this area, the string segments for individual labels should all be of the same length to give a uniform character appearance. This may require padding of some string segments with spaces.

The color of the labels may either correspond to those of their respective pie slices or be uniformly the same.

J

Topic

VBGSLabelPie

Related

GSLabelPie
GSLabelnPie
GSLoadRFont
GSLoadVFont
GSPie2D
GSPie3D

VBGSLabelX function

Draws text labels along X axis in Visual Basic

C/C++ Use GSLabelX function

FoxPro Use GSLabelX function

Visual Basic

```
r% = VBGSLabelX(fxOrg#, fyOrg#, fInc#, fWid#, fHt#,
                nLabs%, nCSet%, nTMode%, nClr%,
                szLabs$)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	X increment
`fWid`	Width of label
`fHt`	Height of label
`nLabs`	Number of labels
`nCSet`	Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.
`nTMode`	Text mode (see Text mode constants)
`nClr`	Text color (see Color constants)
`szLabs`	String of labels delimited by tabs, Chr$(9)

Return values

0	Success
-1	Failure

Description

The VBGSLabelX function, a Visual Basic-specific version of the GSLabelX function, draws a horizontal sequence of text labels starting on screen at (fxOrg,fyOrg) and at intervals of fInc to the right of this point.

Visual Basic doesn't let you pass arrays of text to API functions, so VBGSLabelX, unlike GSLabelX, requires you to create a single string containing the text for all labels. You use the tab character, Chr$(9), to separate label entries within this string.

By default, this function chooses a vector character set and each label is drawn to fit a rectangle fWid wide and fHt high. Since vector text is shaped to fit this area, the string segments for individual labels should all be of the same length to give a uniform character appearance. This may require padding of some string segments with spaces.

J

Topic

VBGSLabelX

Related

GSLabelX
GSLabelnX
GSLoadRFont
GSLoadVFont
GSXYGraph

VBGSLabelY function

Draws text labels along Y axis in Visual Basic

C/C++ Use GSLabelY function

FoxPro Use GSLabelY function

Visual Basic

```
r% = VBGSLabelY(fxOrg#, fyOrg#, fInc#, fWid#, fHt#,
                nLabs%, nCSet%, nMode%, nClr%,
                szLabs$)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fInc`	Y increment
`fWid`	Width of label
`fHt`	Height of label
`nLabs`	Number of labels
`nCSet`	Character set (see Character set constants). Use CSRASTER (2) here to select a raster character set; when you do this, labels aren't sized to fit within the width and height you specify.
`nTMode`	Text mode (see Text mode constants)
`nClr`	Text color (see Color constants)
`szLabs`	String of labels delimited by tabs, Chr$(9)

Return values

0	Success
-1	Failure

Description

The VBGSLabelY function, a Visual Basic-specific version of the GSLabelY function, draws a vertical sequence of text labels starting on screen at (fxOrg,fyOrg) and at intervals of fInc above this point.

Visual Basic doesn't let you pass arrays of text to API functions, so VBGSLabelY, unlike GSLabelY, requires you to create a single string containing the text for all labels. You use the tab character, Chr$(9), to separate label entries within this string.

By default, this function chooses a vector character set and each label is drawn to fit a rectangle fWid wide and fHt high. Since vector text is shaped to fit this area, the string segments for individual labels should all be of the same length to give a uniform character appearance. This may require padding of some string segments with spaces.

J

Topic

VBGSLabelY

Related

GSLabelY
GSLabelnY
GSLoadRFont
GSLoadVFont
GSXYGraph

VBGSLegend function

Draws legend in Visual Basic

C/C++ Use GSLegend function

FoxPro Use GSLegend function

Visual Basic

```
r% = VBGSLegend(fxOrg#, fyOrg#, fWid#, fHt#, nNLeg%,
                nRows%, nMode%, nCSet%, nTMode%, nClr%,
                nBClr%(0), nBPatt%(0), szLegs$)
```

Parameters

`fxOrg`	X origin
`fyOrg`	Y origin
`fWid`	Width of bounding area
`fHt`	Height of bounding area
`nNLeg`	Number of legend entries
`nRows`	Number of rows in legend
`nMode`	

Constant	Value	Meaning
LGBOX	1	Draws black box around legend area
LGTXCLR	2	Text takes its color from the associated legend box
LGBG	4	Fills the bounding area with the current background color
LGLINE	8	Shows line patterns
LGSYMBOL	16	Shows symbols

`nCSet`	Character set (see Character set constants)
`nTMode`	Text mode (see Text mode constants)
`nClr`	Text color (see Color constants)
`nBClr`	Pointer to array of legend box colors (see Color constants)
`nBPatt`	Pointer to array of legend box patterns (see Pattern constants)
`szLegs`	String of legend entries delimited by tabs, Chr$(9)

Return values

0	Success
-1	Failure

Description The VBGSLegend function, a Visual Basic-specific version of the

GSLegend function, draws a legend to accompany a graph or chart.

Visual Basic doesn't let you pass arrays of text to API functions, so VBGSLegend, unlike GSLegend, requires you to create a single string containing the text for all legend items. You use the tab character, Chr$(9), to separate legend entries within this string.

The legend, consisting of a stack or row of patterned and colored boxes associated with text strings, is drawn within a bounding rectangle defined by its width and height and located by the origin at its bottom left. If you choose, you can have the legend show line patterns rather than fill patterns (use nMode LGLINE).

Each legend entry is defined by elements in two arrays--color and pattern--and by a tab-delimited string of text.

J

Topic

VBGSLegend

Related

GSLegend

GSLoadRFont

GSLoadVFont

Parameter constants

Using parameter constants
Character set constants
Color constants
Line style constants
Pattern constants
Symbol constants
Text mode constants

Using parameter constants

When you pass a numeric parameter to Graphics Server, you can generally use either the integer value or a symbolic constant. Symbolic constants are mnemonically named strings that substitute for a numeric value. For example, when you call the AGShow function, you can replace the number 1 in the first parameter with the symbolic constant AGPIE2D. Although there's no performance advantage to using a constant over a numeric value, the constants make your code easier to read and document.

Additive values

Values listed with a prefix of "(+)" are additive. For example, a color parameter of LIGHT + GREEN selects light green.

Character set constants

Constant	Value	Meaning
CSSYSTEM	0	Default vector font
CSUSER	1	Vector font loaded with GSLoadVFont function
(+) CSRASTER	2	Flag for raster font
CSRASTER + CSSYSTEM	2	Default raster font
CSRASTER + CSUSER	3	Raster font loaded with GSLoadRFont function

When you choose a raster font, you can't specify the exact height and width of a block of text.

Color constants

Graphics Server draws its graphing windows using a default palette of 16 colors. These colors are achieved by combining several color constants (BLACK, BLUE, and so on) with an additive LIGHT flag.

Constant	Value
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
WHITE	7
(+) LIGHT	8
LIGHT + BLACK (or GRAY)	8
LIGHT + BLUE	9
LIGHT + GREEN	10
LIGHT + CYAN	11
LIGHT + RED	12
LIGHT + MAGENTA	13
LIGHT + BROWN (or YELLOW)	14
LIGHT + WHITE	15

You may notice that the names for these constants differ in some cases from how equivalent colors are described in the Graph control's documentation. The color called WHITE in the DLL's list of constants is called "light gray" in documentation for the Graph control. The color called LIGHT + BLACK here is described as "dark gray" in the manual for the control. The equivalent of LIGHT + WHITE is described as simply "white." The differences are only in terminology. Despite the different names, the actual numeric color values are the same.

Half-tone colors

Graphics Server reserves sixteen additional color values (16-31) for the half-tone colors of settings 0 through 15. These colors are automatically used for shaded items such as the sides of 3D bars and pie slices and the undersides of 3D "tapes." No symbolic constants are defined for the half-tone colors.

128-color palettes

In this edition of Graphics Server, you aren't limited to the default color palette. The GSSelectPalette function lets you choose from 10 128-color palettes. In that case, color settings 0-15 follow the standard palette, settings 16-31 are the standard half-tones, and settings 32-127 are determined by the palette you select. No symbolic constants are defined for settings 32-127.

Line style constants

Windows lets you apply a pattern or thickness--but not both--to lines drawn on the screen. Several Graphics Server functions ask you to specify either a line pattern or thickness.

Patterned lines

Six patterns are available, including a "null" (invisible) line.

Constant	Value	Pattern
LSSOLID	0	
LSDASH	1	
LSDOT	2	
LSDASHDOT	3	
LSDASHDD	4	
LSNULL	5	Null (invisible) line

Thick lines

The value you specify is the line width in pixels (1 to 5).

Pattern constants

Graphics Server gives you a choice of 24 fill patterns (including solid, null, six hatch patterns, and 16 bitmap patterns) for drawing such graph markers as pie slices, bars, and area plots.

Constant	Value	Pattern
BRSOLID	0	
BRNULL	1	
BRHATCH	2	
BRHATCH + 1	3	
BRHATCH + 2	4	
BRHATCH + 3	5	
BRHATCH + 4	6	
BRHATCH + 5	7	
(+) BRHATCHMAX	6	Flag for maximum number of hatch patterns
BRBITMAP	16	
BRBITMAP + 1	17	
BRBITMAP + 2	18	
BRBITMAP + 3	19	
BRBITMAP + 4	20	
BRBITMAP + 5	21	
BRBITMAP + 6	22	
BRBITMAP + 7	23	
BRBITMAP + 8	24	
BRBITMAP + 9	25	
BRBITMAP + 10	26	
BRBITMAP + 11	27	
BRBITMAP + 12	28	
BRBITMAP + 13	29	
BRBITMAP + 14	30	
BRBITMAP + 15	31	
(+) BRBITMAPMAX	16	Flag for maximum number of bitmap patterns
(+) BRTRANS	64	Flag for transparent mode

The BRHATCHMAX and BRBITMAPMAX flags are included for ease of numbering. Hatch patterns start at BRHATCH and increase up to BRHATCH + BRHATCHMAX - 1; bitmap patterns start at BRBITMAP and increase up to BRBITMAP + BRBITMAPMAX - 1.

When you include the BRTRANS flag (or add 64 to any pattern number), the pattern is transparent and any underlying image shows through. By default, patterns are opaque and underlying images are hidden.

Symbol constants

This edition of Graphics Server offers 14 symbol designs for the graph types that use symbols, such as scatter and line graphs.

Value	Symbol
0	+
1	×
2	△
3	▲
4	▽
5	▼
6	□
7	■
8	◇
9	◆
10	⊠
11	⊕
12	○
13	●

Text mode constants

Graphics Server text-handling functions often include a text mode parameter, which determines how lines of text are drawn. You may pass one of the values--or the sum of several values--in the table below.

Note: The values shown in the table are additive. However, a few are also mutually exclusive. For example, text cannot simultaneously be *left-aligned*, *right-aligned* and *centered*, so do not combine TXLEFT + TXMID + TXRIGHT.

Constant	Value	Meaning
(+) TXEXACT	1	Text conforms to specified height and width (vector fonts only)
(+) TXLEFT	0	Text aligns at left
(+) TXMID	2	Text aligns at midpoint
(+) TXRIGHT	4	Text aligns at right
(+) TXBOTTOM	0	Text aligns along extreme bottom of characters
(+) TXBASELINE	8	Text aligns along baseline of characters
(+) TXTOP	16	Text aligns along top of characters
(+) TXUP90	32	Rotates text 90 degrees counterclockwise
(+) TXDOWN90	64	Rotates text 90 degrees clockwise
(+) TXTRANS	256	Text background is transparent

When you include the TXTRANS flag, the text background (a normally invisible rectangle around the text) is transparent and any underlying image shows through. By default, text backgrounds are opaque and underlying images are hidden.

Label Formats

Numeric format strings
Date/time format strings

Numeric formats

Axes and data points can be labeled either with text or with numbers. When the labels are numeric (set, point, or data values) the numbers can be formatted for display by calling AGLabelFormat. This function accepts a format string similar to those for user-defined numeric formats in Visual Basic. A sequence of placeholder and control characters defines how numbers are formatted for display.

Numeric format characters

Character	Meaning
0	*Digit placeholder* Display a digit or a zero. If the number being formatted has a digit in the position where the 0 appears, display it. Otherwise display a zero. If the number has fewer digits than there are zeros in the format expression, display leading or trailing zeros.
#	*Digit placeholder* Display a digit or nothing. If the number being formatted has a digit in the position where the # appears, display it. Otherwise display nothing. This symbol works like the 0 except that leading and trailing zeros aren't displayed.
.	*Decimal placeholder* The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression has nothing to the left of this symbol, numbers smaller than 1 begin with a decimal separator. If you want a leading zero always to be displayed with fractional numbers, use 0 as the first digit placeholder to the left of the decimal separator. The actual character used as a decimal separator in the formatted output depends on your system settings.
,	*Thousand separator* The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a comma surrounded by digit placeholders (0 or #). Two adjacent commas or a comma immediately to the left of the decimal separator (whether or not a decimal is specified) means "scale the number by dividing by a thousand, rounding as needed". You can scale large numbers using this technique. For example, you can use the format string "##0,," to represent 100 million as 100. The actual character used as the thousand separator in the formatted output depends on your system settings.
%	*Percentage placeholder* The number is multiplied by 100. The percent character (%) is inserted in the position where it appears in the format string.
E-E+e-e+	*Scientific format* If the format expression contains at least one digit placeholder (0 or #) to the right of E-, E+, e- or e+ the number is displayed in scientific format. The character "E" or "e" is inserted between the number and its exponent. The number of digits in the exponent is determined by the number of digit placeholders to the right.

Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a minus sign next to negative exponents and a plus sign next to positive exponents.

\ *Literal character*

Display the character immediately following "\" in the format string.

Literal characters can be inserted before or after the formatted number. For example "\$##0.0" will insert "$" before the formatted number. "##0.0\D\M" will insert "DM" after the formatted number.

Examples

Number	Format string	Displayed
1234	0	1234
1234	00000	01234
1234	#,##0	1,234
1234	#,##0.00	1,234.00
1234	\$#,##0.00	$1,234.00
123456	\$#,##0,\k	$123k
12345678	\$#,##0,,\m	$12m
12345678	0.0E+00	1.2E+07
0.1234	0.00	0.12
0.1234	0.00000	0.12340
0.1234	0.0E-00	1.2E-01
0.1234	0.00%	12.34%
0.1234	0%	12%

J

Topic

Numeric formats

Related

AGLabelFormat

Date/time formats

Date/time formats

The X axis can be labeled with text, numbers or an automatically generated series of date and time labels. To label the axis with a series of dates or times, call the AGLabelDateTime function. To apply a format to the dates or times, call AGLabelFormat.This function accepts a label format string similar to those for user-defined date and time formats in Visual Basic. A sequence of placeholder and control characters defines how dates and times are formatted for display.

Date format characters

Character	Meaning
/ - : . ,	Delimiters. Any of these characters, as well as a space character, may be used to separate elements of the displayed date.
c	Display the date as ddddd and the time as ttttt in that order.
d	Display the day as a number without a leading zero (1-31).
dd	Display the day as a number with a leading zero (01-31).
ddd	Display the day as an abbreviation (Sun/Sat).
dddd	Display the day as a full name (Sunday/Saturday).
ddddd	Display the date as a short date (including day, month, and year), formatted according to your system's short date format setting. The default short date format is m/d/yy.
dddddd	Display the date as a long date (including day, month, and year) formatted according to the long date setting recognized by your system. The default long date format is mmmm dd, yyyy.
w	Display the day of the week as a number (1 for Sunday through 7 for Saturday).
ww	Display the week of the year as a number (1-53).
m	Display the month as a number without a leading zero (1-12).
mm	Display the month as a number with a leading zero (01-12).
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full month name (January-December).
q	Display the quarter of the year as a number (1-4).
y	Display the day of the year as a number (1-366).
yy	Display the year as a 2-digit number (00-99).
yyyy	Display the year as a 4-digit number (1900-2037).

Examples

yyyy-mm-dd	Format string	Displayed
1997-03-01	d mmm yy	1 Mar 97
1997-03-01	dddd	Saturday
1997-03-01	mmmm	March
1997-03-01	dd mmmm yyyy	01 March 1997

1997-03-01	dddd, mmmm d, yyyy	Saturday, March 1, 1997
1997-03-01	w	7
1997-03-01	q	1
1997-03-01	y	60

Time format characters

Character	Meaning
/ - : . ,	Delimiters. Any of these characters, as well as a space character, may be used to separate elements of the displayed time.
h	Display the hour as a number without leading zeros (1-23).
hh	Display the hour as a number with leading zeros (01-23).
n	Display the minute as a number without leading zeros (0-59).
nn	Display the minute as a number with leading zeros (00-59).
s	Display the second as a number without leading zeros (0-59).
ss	Display the second as a number with leading zeros (00-59).
t t t t t	Display a time as a complete time (including hour, minute, and second), formatted using the time separator defined by the time format recognized by your system. The default time format is h:mm:ss.
AM/PM	Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 P.M.
am/pm	Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 P.M.
A/P	Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 P.M.
a/p	Use the 12-hour clock and display a lowercase A with any hour before noon; display a lowercase P with any hour between noon and 11:59 P.M.

Examples

hh-mm-ss	Format string	Displayed
13-30-25	ttttt	13:30:25
13-30-25	h:nn	13:30
13-30-25	nn:ss	30:25
13-30-25	h:nn AM/PM	1:30 PM
13-30-25	hh:nn:ss a/p	01:30:25 p

J

Topic

Date/time formats

Related

AGLabelFormat

Numeric formats