GAUSSplot™

Professional Graphics

Part Number: 005712
Version 8.0
Documentation Revision: 860 July 20, 2007

Contents

1 Overview

2 Command Overview

1 Overview

GAUSSplot is simple enough to allow users to create a graph with only a few lines of code, yet flexible enough to allow for the creation of elaborate graphs incorporating different types of plots. Users can add text, legends, arrows and symbols to a graph, as well as configure a wide variety of output and view options.

To illustrate **GAUSSplot**'s ease of use, let us begin by demonstrating how to create a simple XY Line plot. The graph in Figure 1.1 is generated by these lines of code:

```
// Set up library and include file with structure definitions
library gaussplot;
#include gp.sdf

// Create gpPlotControl structure and set members to default
// values
struct gpPlotControl gp;
gp = gpXYLinePlotCreate;

// Generate data
x = seqa(0.2,.2,15);
```

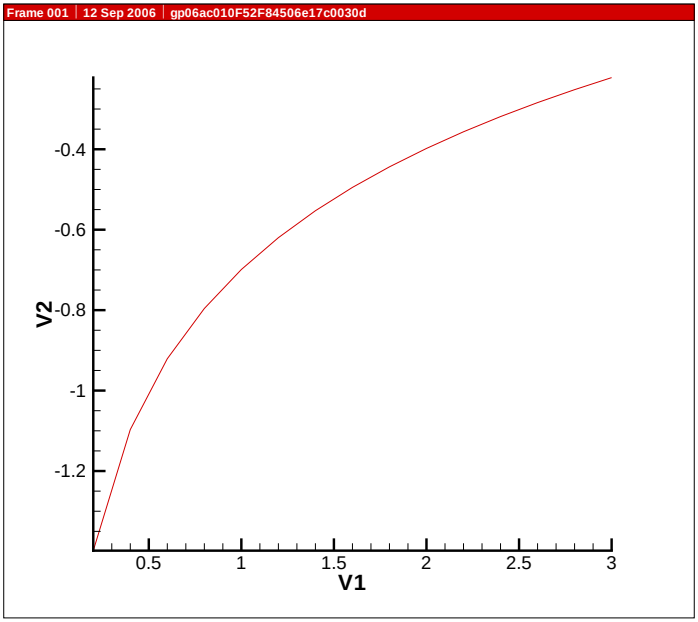


Figure 1.1: Simple XY Plot

```
y = log(x/5);

// Write data to data file and create graph
ret = gpMakeXYLinePlot(&gp,x,y,"");
```

This example is a slightly simplified version of `xylineplot1.e`, which can be found in the `gaussplot/examples` subdirectory of your **GAUSS** installation directory. For more examples, see Section 1.10 or your `gaussplot/examples` subdirectory.

1.1 Getting Started

There are four steps to creating a graph with **GAUSSplot**:

1. Setup and initialization
2. Creating data and writing the data file
3. Setting various plot options
4. Plotting the graph

These must all be present in any program that creates a graph using **GAUSSplot**, with the exception of the second step if the necessary data file already exists.

1.1.1 Setup and Initialization

The setup and initialization process must include the following calls:

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
```

as well as a call to the appropriate **gp***PlotType***PlotCreate** command for the type of graph that is to be created. For instance, if you wanted to create a 3D Cartesian graph, you would add the following line to the above:

```
gp = gp3DCartesianPlotCreate;
```

See Section 1.5.4 for a list of all of the plot types that are supported in **GAUSSplot** and Section 2.1.1 for the available **gp***PlotType***PlotCreate** commands.

1.1.2 Creating Data

If the necessary data file does not already exist, then the next step is to create data. If you are going to use **gpPlot** to plot the graph (see Section 1.1.4), then you must also write the data file at this point in your program. However, if you use one of the **gpMake***PlotType***Plot** commands to plot the graph, then it will write the data to a data file for you, and you need only to generate that data.

The following code creates data for a data file with one zone, containing three variables, and writes the data file:

```
struct gpData gdat;

// Generate data for 3D Cartesian frame
a = seqa(0,pi/12,25)';
r = seqa(6,-0.25,25);
x = r.*cos(a);
y = r.*sin(a);
z = rev(1.5*r).*ones(25,25);

// Put data into an array
a = aconcat(x,y,3);
a = aconcat(a,z,3);

// List variable names
string vnames = { "X", "Y", "Z" };
```

```
// Set data in gpData structure
gdat = gpInitPlotData(vnames);
ret = gpAddZone(&gdat,"Zone One",a);

// Write data file
ret = gpWritePlotData(&gdat,"mydata.plt");
```

For an explanation of zones and data file structure, see Section 1.3.

1.1.3 Setting Plot Options

GAUSSplot contains many plot option commands, which allow you to specify the number of line maps or field zones that are displayed in a plot and how they are to be displayed. For an explanation of line maps and field zones, see Section 1.5.4. Most of these commands are optional. **GAUSSplot** will use defaults for all options that have not been explicitly set. However, if you use **gpPlot** to plot your graph, then it is necessary to call **gpSetDataFile** before calling **gpPlot** in order to specify which data file(s) are to be used for the graph.

It is at this stage in a **GAUSSplot** program that you need to add frames for a multiple frame graph (see Section 1.4 for an explanation of frames). If you use **gpPlotCreate** to initialize your plot control structure, then your graph does not contain any frames by default. Therefore, you need to use the various **gpAdd***PlotType***Frame** commands to add frames before plotting the graph.

Alternatively, if you use one of the **gp***PlotType***PlotCreate** commands to initialize your plot control structure, then your graph contains one frame by default. In this case, it is not necessary (though allowable) to use the appropriate **gpAdd***PlotType***Frame** command to add additional frames because the plot type for all frames in the graph has already been determined by the call to one of the **gp***PlotType***PlotCreate** commands. Therefore, you may add additional frames implicitly by calling **gpSetLineMaps** or **gpSetLineMapZone** for line plots or **gpSetAxisMap** for field plots with a frame number argument that is higher than one. Frames added in this manner should be added in order. For example, you should not call **gpSetLineMaps** with a frame number of 3 before calling **gpSetLineMaps** or **gpSetLineMapZone** with a frame number of 2.

Following the above code, we might call the following commands to configure our 3D Cartesian plot with one frame, containing one field zone:

```
framenum = 1;
fieldzonenum = 1;
activate = 1;
transparency = 70;
xpos = 41;
ypos = 91;
positionunit = 1;

// Specify the data file to use
ret = gpSetDataFile(&gp,framenum,"mydata.plt");

// Activate the shade layer and set the shade color
ret = gpShowShadeLayer(&gp,framenum,activate);
ret = gpSetShadeColor(&gp,framenum,fieldzonenum,"red");

// Activate the transparency effect and set the percentage of
// transparency
ret = gpUseTransparencyEffect(&gp,framenum,activate);
ret = gpSetTransparency(&gp,framenum,fieldzonenum,transparency);

// Make the axes sized independently of one another
ret = gpMakeAxesIndependent(&gp,framenum);

// Add text for a title
ret = gpAddTextObject(&gp,framenum,"text1",xpos,ypos,
                          positionunit,"3D Surface Plot");
```

1.1.4 Plotting the Graph

Once all of the desired plot options have been set, you may plot the graph:

```
ret = gpPlot(&gp);
```

For simple graphs with only one frame, containing one or more line maps or one field zone, you may use the appropriate **gpMake***PlotType***Plot** command to write the data to a data file and plot

the graph. See Chapter 3 for details on using these commands.

The graph plotting commands take the information in the **gpPlotControl** structure and write it out to a **GAUSSplot** macro file, which is then run in **GAUSSplot** to create the graph.

After the graph has been plotted, the information for the graph may be saved as a layout file by selecting 'Save Layout' from the **GAUSSplot** 'File' menu. The layout file may then be re-opened with the same or different data by a call to **gpOpenLayoutFile**.

1.2 What Next

1.2.1 Customizing Your Graph

Once you have learned how to create basic graphs using **GAUSSplot** and are ready to try a complex, highly customized graph, see the COMMAND OVERVIEW (Chapter 2) to point you to the commands that you need to customize various options. If you cannot find what you are looking for in the COMMAND OVERVIEW chapter, then complete the following steps to finish customizing your graph:

1. Run your **GAUSSplot** program in **GAUSS**.
2. In the **GAUSSplot** graphical user interface (GUI), select "Macro"→"Record" from the 'File' menu.
3. In the 'Write Macro File' dialog box, choose a file name and location for the macro file that you are creating. Be sure that you are not using the same name as the macro file that is created by your **GAUSSplot** program (use **gpSetMacroFile** in your **GAUSSplot** program to ensure this).
4. Make any desired changes to your graph using the **GAUSSplot** GUI buttons and menu options.
5. Select the "Stop Recording" button on the 'Macro Record' dialog box, and then select "OK" when you are asked if you are sure that you want to stop recording.

6. Now go back to your **GAUSSplot** program and call **gpAddMacroFile** with the name of your newly created macro file right before the call to plot the graph. Then when you run your program again, the changes that you made with the GUI will be be done for you automatically.

See the TECPLOT® USER'S MANUAL in the file `tpum.pdf` for more information on using the graphical user interface.

1.2.2 GP Tools

GP TOOLS is a free package containing additional procedures that may be used with **GAUSSplot**. The package includes quick and dirty procedures for producing simple graphs as well as procedures that make use of **GAUSSplot**'s graphing capabilities in specific applications.

GP TOOLS may be downloaded from the `gptools` directory on Aptech's ftp site. The directory contains a `.zip` file for use on Windows, a `.tar.gz` file for use on UNIX and Linux and a `.pdf` with installation instructions and documentation for each of the procedures in the package.

1.3 Data Files

1.3.1 Structure of a Data File

A **GAUSSplot** graph is created from data in a **GAUSSplot** format binary data file, which is indicated by a `.plt` extension. A `.plt` data file is organized into zones (blocks of data). The number of variables and the variable names are set once at the top of the data file and apply to each zone in the file. Thus, if a `.plt` data file contains three variables (**X**, **Y**, and **Z**), each zone in the file must contain three variables named **X**, **Y**, and **Z**.

Within a zone, each variable must be the same size (i.e., each variable must contain the same number of rows and the same number of columns). However, the number of elements contained in a variable may vary from zone to zone. For example, the following describes a valid data file format:

Zone 1		
Variable	Rows	Cols
X	100	50
Y	100	50
Z	100	50

Zone 2		
Variable	Rows	Cols
X	50	30
Y	50	30
Z	50	30

Zone 3		
Variable	Rows	Cols
X	15	15
Y	15	15
Z	15	15

1.3.2 Data File Creation Commands

GAUSSplot contains a set of routines for creating `.plt` data files. These routines allow you to create and initialize a **gpData** structure, add data to it, and write the data contained in the structure to a `.plt` file:

gpInitPlotData	Creates and initializes a **gpData** structure using user-specified variable names.
gpAddZone	Adds all of the variables for one zone of the data file to the **gpData** structure.
gpSetPlotData	Creates, initializes, and adds data to a **gpData** structure for a data file that will contain only one zone with K variables that are each N×1.
gpWritePlotData	Writes a `.plt` data file using the data contained in a **gpData** structure.

The following example illustrates how you may use the data file creation commands to create a data file with two zones, containing three 2-dimensional variables;

```
library gaussplot;
#include gp.sdf

// Generate data
l = 20;
pil = pi/(l-1);
x = seqa(-10,2.5,l)'*pil;
y = seqa(-10,2.5,l)*pil;

// Create 3 variables that are 20x20 for zone 1
xz1 = x.*ones(l,l);
yz1 = y.*ones(l,l);
zz1 = sin(x).*cos(x-y);

// Create 3 variables that are 11x6 for zone 2
xz2 = xz1[5:15,5:10];
yz2 = yz1[5:15,5:10];
zz2 = cos(xz1)-sin(yz1)+1;

// Copy the data for zone 1 into a 3x20x20 array
az1 = aconcat(xz1,yz1,3);
az1 = aconcat(az1,zz1,3);

// Copy the data for zone 2 into a 3x11x6 array
az2 = aconcat(xz2,yz2,3);
az2 = aconcat(az2,zz2,3);

// Initialize gpData structure, add data, and write data file
struct gpData gdat;
string vnames = { "X", "Y", "Z" };
gdat = gpInitPlotData(vnames);
ret = gpAddZone(&gdat,"Zone 1",az1);
ret = gpAddZone(&gdat,"Zone 2",az2);
ret = gpWritePlotData(&gdat,"mydata.plt");
```

1.3.3 Inputting Data

You may use **gpSetPlotData** to add data stored in vectors or matrices to a data structure for Bar, XY Line, and Polar Line plots. For all other plot types, **gpAddZone** must be used to add the data to a data structure. **gpAddZone** inputs the data as a matrix or an array.

Data for Bar, XY Line and Polar Line Plots

For Bar, XY Line and Polar Line plots, the data input into **gpAddZone** may be either an N×K matrix or a 3-dimensional (K×N×1) array containing K variables that are N×1. For example, suppose you created *x* and *y* variables for a line plot as follows:

```
// Create gpData structure
struct gpData gdat;

// Create data - 2 variables that are 8x1
x = ones(8,1);
y = seqa(8,-1,8);
string vnames = { "X", "Y" };

// Set data in gpData structure
gdat = gpInitPlotData(vnames);
ret = gpAddZone(&gdat,"Zone 1",x~y);

// Write data file
ret = gpWritePlotData(&gdat,"lineplot.plt");
```

The resulting set of data points (x, y) would be laid out as in Figure 1.2 and could be represented by a series of line segments connecting each point $(x[i, 1], y[i, 1])$ with the points immediately preceding and following (i.e., $(x[i - 1, 1], y[i - 1, 1])$ and $(x[i + 1, 1], y[i + 1, 1])$).

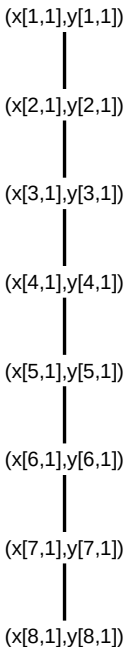


Figure 1.2: Line Plot Point Arrangement

Data for 2D Cartesian and 2D Contour Plots

For 2D Cartesian and 2D Contour plots, the data array that is input into **gpAddZone** should be a 3-dimensional (K×N×P) array containing K variables that are N×P. To illustrate this, suppose you used the following commands to create the 4×4 variables *x* and *y* for a 2D Cartesian plot:

```
// Create gpData structure
struct gpData gdat;

// Create data - 2 variables that are 4x4
x = seqa(1,1,4)'.*ones(4,1);
y = seqa(4,-1,4).*ones(1,4);

// Copy the data into an array that is 2x4x4
a = aconcat(x,y,3);
string vnames = { "X", "Y" };

// Set data in gpData structure
gdat = gpInitPlotData(vnames);
ret = gpAddZone(&gdat,"Zone 1",a);

// Write data file
ret = gpWritePlotData(&gdat,"surfaceplot.plt");
```

The resulting set of data points (x, y) would be laid out as in Figure 1.3, where *a* = **aconcat(***x*,*y*,*3***)**. The set of data points can be represented, as is shown in Figure 1.3, by a series of line segments connecting each point $(x[i, j], y[i, j])$ with its neighboring points (i.e., $(x[i - 1, j], y[i - 1, j])$, $(x[i, j - 1], y[i, j - 1])$, $(x[i + 1, j], y[i + 1, j])$, and $(x[i, j + 1], y[i, j + 1])$).

Data for 3D Cartesian and 3D Contour Plots

For 3D Cartesian and 3D Contour plots, the data array input into **gpAddZone** may be either a 3-dimensional (K×N×P) array containing K 2-dimensional variables (to plot a surface) or a 4-dimensional (K×L×N×P) array containing K 3-dimensional variables (to plot a volume).

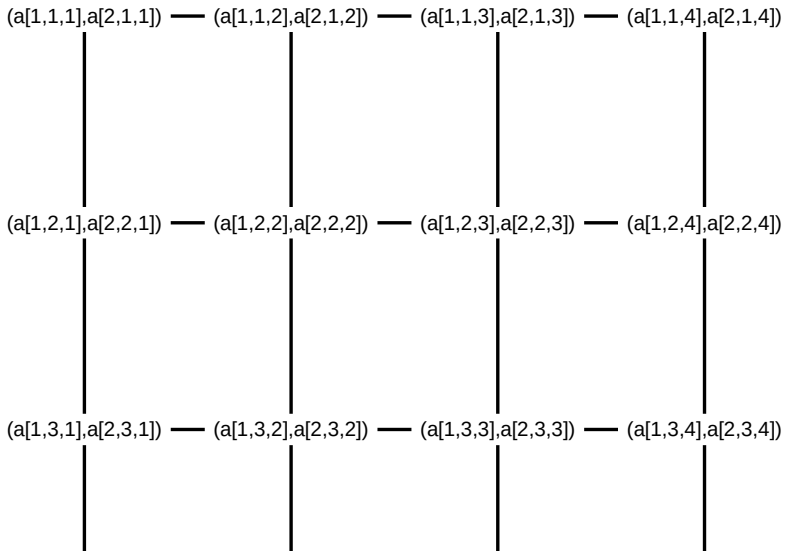


Figure 1.3: 2D Cartesian Plot Point Arrangement

A 3D Cartesian plot created with 3 variables (corresponding to the X, Y and Z axes) that are N×P would be laid out just like the 2D Cartesian plot in Figure 1.3 except that each point would have X, Y and Z coordinates (i.e., each point would be defined as $(x[i, j], y[i, j], z[i, j])$).

For an example of a 3D Cartesian plot with 3 variables that are L×N×P, suppose you used the following commands to create the 2×3×4 variables *x*, *y* and *z* for a 3D Cartesian plot:

```
// Create gpData structure
struct gpData gdat;

// Create data - 3 variables that are 2x3x4
x = seqa(1,1,4)'.*ones(3,1);
x = aconcat(x,x,3);
y = seqa(3,-1,3).*ones(1,4);
y = aconcat(y,y,3);
z = ones(3,4);
z = aconcat(z,2*z,3);
```

```
// Copy the data into an array that is 3x2x3x4
a = aconcat(x,y,4);
a = aconcat(a,z,4);
string vnames = { "X", "Y", "Z" };

// Set data in gpData structure
gdat = gpInitPlotData(vnames);
ret = gpAddZone(&gdat,"Zone 1",a);

// Write data file
ret = gpWritePlotData(&gdat,"volumeplot.plt");
```

The resulting set of data points (x, y, z) would be laid out as in Figure 1.4 and could be represented by a series of line segments connecting each point $(x[k, i, j], y[k, i, j], z[k, i, j])$ with its neighboring points in the same plane (i.e., $(x[k, i-1, j], y[k, i-1, j], z[k, i-1, j])$, $(x[k, i, j-1], y[k, i, j-1], z[k, i, j-1])$, $(x[k, i+1, j], y[k, i+1, j], z[k, i+1, j])$, and $(x[k, i, j+1], y[k, i, j+1], z[k, i, j+1])$) as well as with the corresponding points in its neighboring plane(s) (i.e., $(x[k-1, i, j], y[k-1, i, j], z[k-1, i, j])$ and $(x[k+1, i, j], y[k+1, i, j], z[k+1, i, j])$).

The points in Figure 1.4 are defined in Table 1.1, where a contains x, y and z concatenated in the fourth dimension in that order.

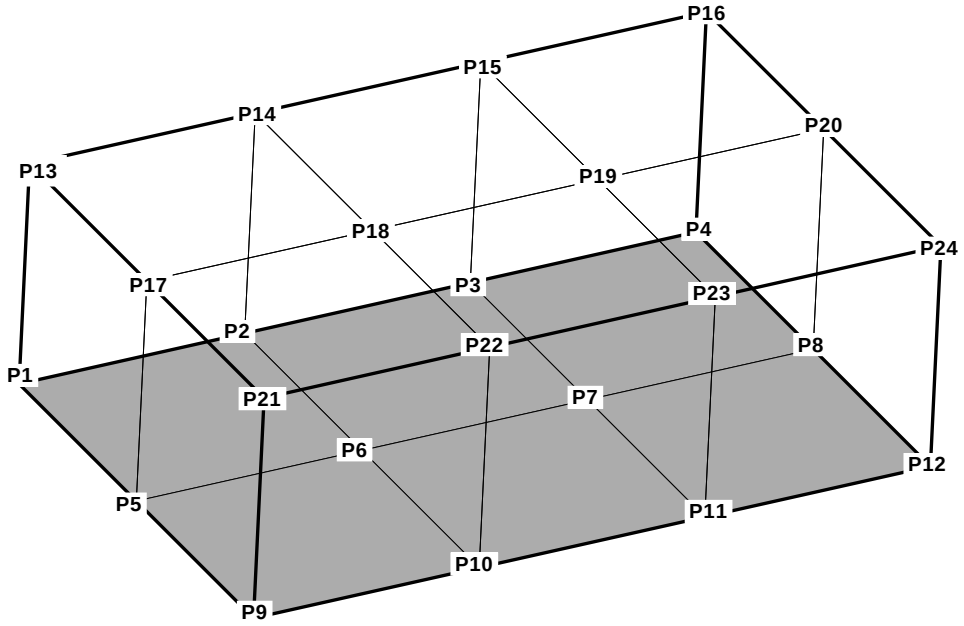


Figure 1.4: 3D Cartesian Plot Point Arrangement

Point	Definition
P1	$(a[1,1,1,1], a[2,1,1,1], a[3,1,1,1])$
P2	$(a[1,1,1,2], a[2,1,1,2], a[3,1,1,2])$
P3	$(a[1,1,1,3], a[2,1,1,3], a[3,1,1,3])$
P4	$(a[1,1,1,4], a[2,1,1,4], a[3,1,1,4])$
P5	$(a[1,1,2,1], a[2,1,2,1], a[3,1,2,1])$
P6	$(a[1,1,2,2], a[2,1,2,2], a[3,1,2,2])$
P7	$(a[1,1,2,3], a[2,1,2,3], a[3,1,2,3])$
P8	$(a[1,1,2,4], a[2,1,2,4], a[3,1,2,4])$
P9	$(a[1,1,3,1], a[2,1,3,1], a[3,1,3,1])$
P10	$(a[1,1,3,2], a[2,1,3,2], a[3,1,3,2])$
P11	$(a[1,1,3,3], a[2,1,3,3], a[3,1,3,3])$
P12	$(a[1,1,3,4], a[2,1,3,4], a[3,1,3,4])$
P13	$(a[1,2,1,1], a[2,2,1,1], a[3,2,1,1])$
P14	$(a[1,2,1,2], a[2,2,1,2], a[3,2,1,2])$
P15	$(a[1,2,1,3], a[2,2,1,3], a[3,2,1,3])$
P16	$(a[1,2,1,4], a[2,2,1,4], a[3,2,1,4])$
P17	$(a[1,2,2,1], a[2,2,2,1], a[3,2,2,1])$
P18	$(a[1,2,2,2], a[2,2,2,2], a[3,2,2,2])$
P19	$(a[1,2,2,3], a[2,2,2,3], a[3,2,2,3])$
P20	$(a[1,2,2,4], a[2,2,2,4], a[3,2,2,4])$
P21	$(a[1,2,3,1], a[2,2,3,1], a[3,2,3,1])$
P22	$(a[1,2,3,2], a[2,2,3,2], a[3,2,3,2])$
P23	$(a[1,2,3,3], a[2,2,3,3], a[3,2,3,3])$
P24	$(a[1,2,3,4], a[2,2,3,4], a[3,2,3,4])$

Table 1.1: Graph Point/Array Correspondence for Figure 1.4

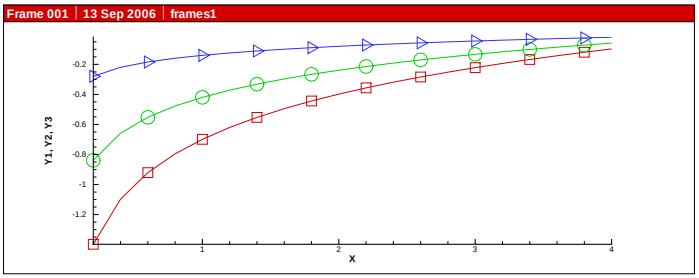


Figure 1.5: Frames

1.4 Basic Structure of a Graph

1.4.1 Workspace

In **GAUSSplot**, a graph is created in a workspace. The workspace is the page that the graph is drawn on. All elements of the graph are contained in the workspace.

1.4.2 Frame

The most basic unit within a workspace is a frame. A graph can contain multiple frames, and each frame contains a set of axes (although the user may choose not to show them) and the plotting of one or more sets of data points. See Figure 1.5.

1.5 Plot Types

1.5.1 Basic Plot Types

There are four basic plot types in the underlying graphics engine used by **GAUSSplot**. Each frame must be associated with one and only one of these basic plot types. The basic plot types are:

- XY Line
- Polar Line
- 2D Cartesian
- 3D Cartesian

1.5.2 Line Plots

XY Line and Polar Line plots both graph linear sets of points (and are thus termed line plots), that is, for both XY Line and Polar Line plots, one N×1 variable is associated with the X or Theta-axis and one N×1 variable is associated with the Y or R-axis. Together these two variables specify a linear set of points. This linear set of points is referred to as a line map. See Figure 1.6.

An XY or Polar Line plot may contain multiple line maps, which are each controlled separately. For each line map, you must specify one variable to be associated with the X or Theta-axis and one variable to be associated with the Y or R-axis. Each line map may be represented by one or more of the following layers: lines, symbols, and (for XY Line plots), bars and error bars.

1.5.3 Field Plots

Both 2D Cartesian and 3D Cartesian plots are field plots, that is, for both types of plots, there is one N×P (or L×N×P in the case of 3D Cartesian volume plots) variable associated with the X-axis, one N×P (or L×N×P) variable associated with the Y-axis, and, for 3D Cartesian plots, one N×P (or L×N×P) variable associated with the Z-axis. Together these variables specify a locus of

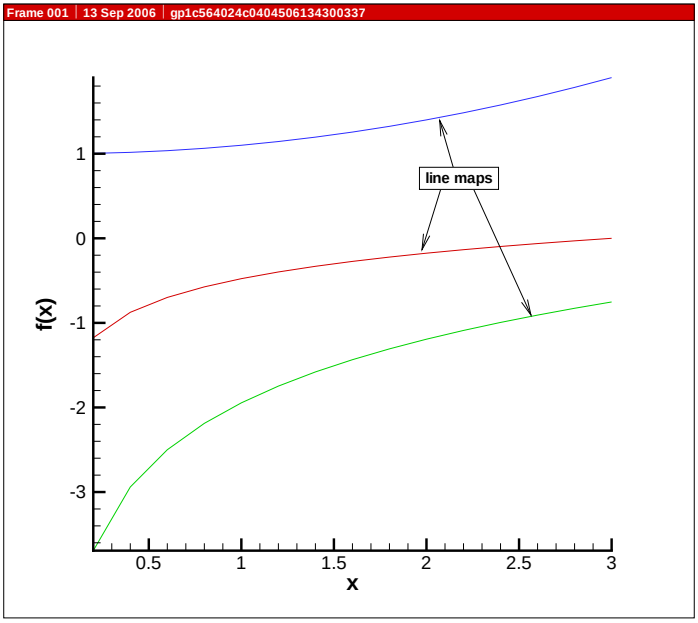


Figure 1.6: Line Maps

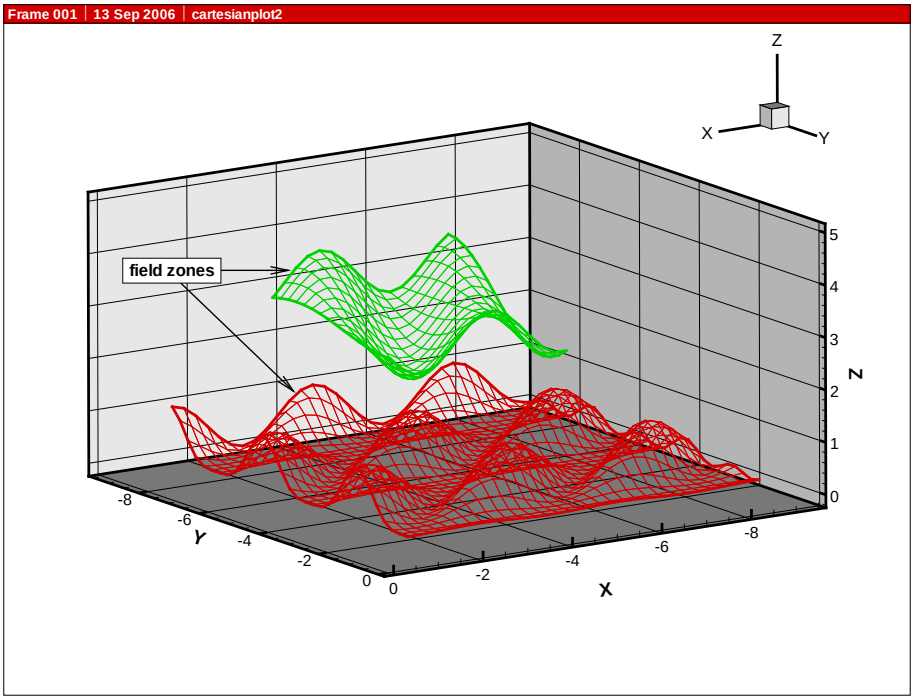


Figure 1.7: Field Zones

points in 2D or 3D space that constitute a field (i.e., a surface or a volume), which may be represented by one or more of the following layers: mesh connecting neighboring points, contour, vector, scatter, shade, and boundary.

A field plot may contain multiple fields or field zones if the associated `.plt` data file contains more than one zone. Section 1.3.1 contains an explanation of zones. See Figure 1.7.

For each 2D and 3D Cartesian plot, you must specify one variable to be associated with the X-axis, one variable to be associated with the Y-axis, and, for 3D plots, one variable to be associated with the Z-axis. This specification of the variables to be mapped to each axis is called an axis map. Axis maps are set for the entire plot, not for a particular field zone. All of the field zones in a frame use the same axis map, but different zones in the data file. Thus a field plot will contain as many field zones as there are zones in the associated data file. For example, if a user

wanted to create a 3D Cartesian plot with the following data file:

Zone 1		
Variable	Rows	Cols
V1	50	10
V2	50	10
V3	50	10

Zone 2		
Variable	Rows	Cols
V1	20	5
V2	20	5
V3	20	5

then he would have two field zones in his plot. In other words, supposing he associated $V1$ with the X-axis, $V2$ with the Y-axis and $V3$ with the Z-axis, then he would have two completely separate sets of data points defined by ($V1$,$V2$,$V3$), the first containing 500 data points, and the second containing 100 data points. Display options, such as which layers are activated (e.g. mesh, contour, etc.), are controlled separately for each field zone.

1.5.4 GAUSSplot Plot Types

In addition to the four basic plot types, **GAUSSplot** supports other plot types that are built from the basic plot types using various view options. Since each of these additional plot types are derived from one of the basic plot types, it is that basic plot type which must be associated with the frame containing the plot. For example, one of the additional plot types is a Bar plot, which is an XY Line plot in which the line maps are displayed as bars. A frame that contains a Bar plot will be associated with the XY Line plot type.

Plot types that are derived from the same basic plot type may be displayed together in the same frame because they differ only in the view options used to display the data or the method by which the data is presented.. For example, you may create a Bar plot with three line maps (i.e. three sets of data points) and then change one of the line maps to be displayed as a line instead of bars, which would result in a frame that contained both Bar and XY Line plots. However, you may not

show plot types that are derived from different basic plot types together in a frame. For example, you may not have a Bar plot and a 2D Cartesian plot in the same frame.

Following is a complete list of the **GAUSSplot** plot types, including the basic plot types from which they are derived and the view or data presentation options that are associated with each:

Plot Type	Basic Plot Type	View Options
2D Cartesian plot	2D Cartesian plot	Field zones displayed as mesh
2D Contour plot	2D Cartesian plot	Field zones displayed with contour flood
3D Cartesian plot	3D Cartesian plot	Field zones displayed as mesh
3D Contour plot	3D Cartesian plot	Field zones displayed with contour flood
Bar plot	XY Line plot	Line maps displayed as bars
Box plot	XY Line plot	Displays data using the box graph percentile method
Hist plot	XY Line plot	Displays a frequency histogram for the data
HistF plot	XY Line plot	Displays a frequency histogram for the data, given frequency counts
HistP plot	XY Line plot	Displays a percentage histogram for the data
LogX plot	XY Line plot	Uses log scaling for the X-axis
LogY plot	XY Line plot	Uses log scaling for the Y-axis
LogLog plot	XY Line plot	Uses log scaling for the X and Y axes
Polar Line plot	Polar Line plot	Line maps displayed as lines
XY Line plot	XY Line plot	Line maps displayed as lines

The following plot types are considered **line plots**: Bar, Box, Hist, HistF, HistP, LogX, LogY, LogLog, Polar Line, XY Line.

2D Cartesian, 2D Contour, 3D Cartesian, and 3D contour plots are all **field plots** .

Because Box, Hist, HistF, and HistP plots all require a manipulation of the data, they are handled slightly differently than other plot types.

Box Plots

To create a graph with one frame containing a box plot, you may use **gpBoxPlotCreate** to initialize your plot control structure, and **gpMakeBoxPlot** to generate the graph. For example, the

following code:

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpBoxPlotCreate;

rndseed 12345;
nboxes = 5;
npoints = 7;
x=seqa(1,1,nboxes)';
y=rndn(npoints,nboxes);

ret = gpMakeBoxPlot(&gp,x,y);
```

produces the graph in Figure 1.8

To create a graph with multiple frames, each containing a box plot, you may use **gpBoxPlotCreate** to initialize your plot control structure, **gpAddBoxFrame** to add frames to your graph, and **gpPlot** to generate the graph. Unlike with other plot types, when you initialize a plot control structure with **gpBoxPlotCreate**, you must use **gpAddBoxFrame** to add EACH frame, including the first. The reason for this is that **gpAddBoxFrame** takes your data as an argument and generates the boxes for the frame.

Note that you will still need to use **gpWritePlotData** to write your data file, and **gpSetDataFile** to set the data file in your plot control structure.

For example, you could use the following code to generate the graph in Figure 1.9 with two frames, each containing box plots:

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpBoxPlotCreate;
```

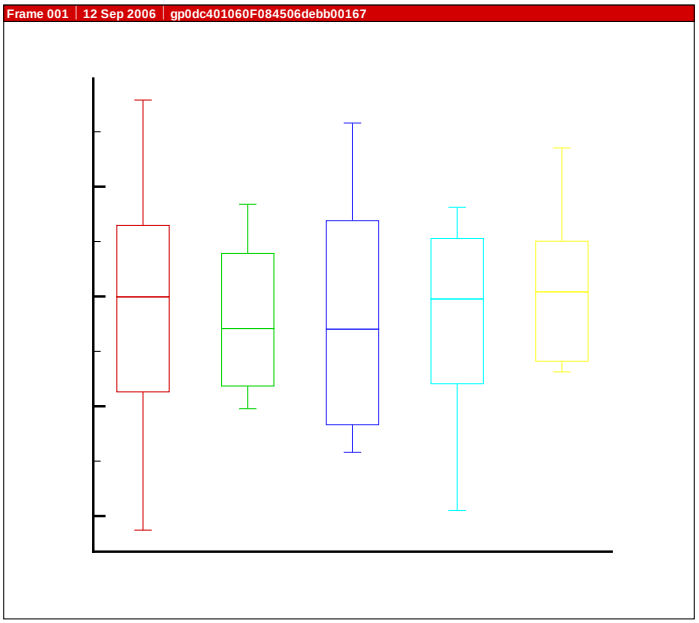


Figure 1.8: Simple Box Plot

```
// Create random data
rndseed 14738;
nboxes = 10;
npoints = 20;
x=reshape(seqa(1,1,nboxes)',npoints,nboxes);
y=rndn(npoints,nboxes);

// Write data file
struct gpData gdat;
gdat = gpSetPlotData(x~y,"");
call gpWritePlotData(&gdat,"gpbox.plt");

// Set data and macro files
gpSetMacroFile(&gp,"gpbox.mcr");
call gpSetDataFile(&gp,0,"gpbox.plt");

// Set up frame 1
call gpAddBoxFrame(&gp,x,y);
call gpMoveFrame(&gp,1,1,0.25,9,3.925);

// Set up frame 2
call gpSetBoxMinMaxMethod(&gp,2,2);
call gpAddBoxFrame(&gp,x,y);
call gpShowSymbolLayer(&gp,2,1);
call gpSetSymbolSize(&gp,2,seqa(1,1,nboxes),1);
call gpSetBoxColor(&gp,2,seqa(1,1,nboxes),"red");
call gpSetSymbolColor(&gp,2,seqa(1,1,nboxes),"cyan");
call gpMoveFrame(&gp,2,1,4.325,9,3.925);

// Plot the graph
call gpPlot(&gp);
```

To create a graph with multiple frames, containing different types of plots, use **gpPlotCreate** to initialize the plot control structure, **gpAdd***PlotType***Frame** to plot the graph.

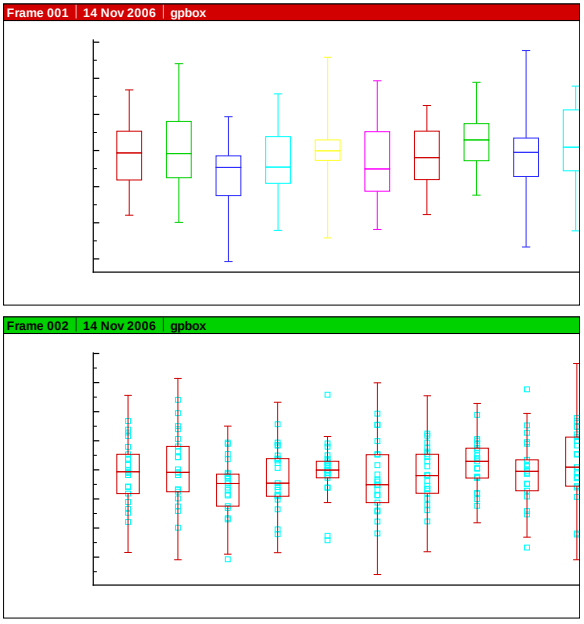


Figure 1.9: Multiple-Frame Box Plot

Histograms

To create a graph with one frame containing a Hist plot, you may use **gpBoxPlotCreate** to initialize your plot control structure, and **gpMakeHistPlot** to generate the graph. For example, the following code:

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpHistPlotCreate;

rndseed 454356;
x = round(rndn(4000,1)*100);

{ b, m, freq, ret } = gpmakehistplot(&gp,x,50);
```

produces the graph in Figure 1.10

To create a graph with multiple frames, containing one or more hist plots, you may use **gpHistPlotCreate** to initialize your plot control structure, **gpAddHistFrame** as well as the other **gpAdd***PlotType***Frame** commands to add frames to your graph. and **gpPlot** to generate the graph. However, you must also call **gpCreateHistData** before writing your data file to manipulate your data for the hist plot.

The commands available for generating various types of histograms are as follows:

Hist plots	HistF plots	HistP plots
gpHistPlotCreate	gpHistFPlotCreate	gpHistPPlotCreate
gpCreateHistData		gpCreateHistPData
gpAddHistFrame	gpAddHistFFrame	gpAddHistPFrame
gpMakeHistPlot	gpMakeHistFPlot	gpMakeHistPPlot

For example, you could use the following code to generate the graph in Figure 1.11 with two frames, the first containing a hist plot, and the second a histp plot:

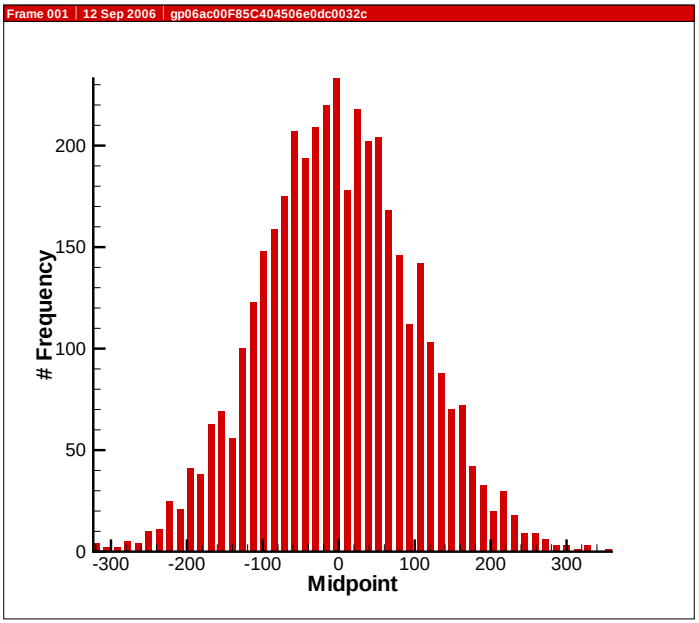


Figure 1.10: Simple Hist Plot

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

// Create random data
rndseed 454356;
x = round(rndn(4000,1)*100);
{ v1, m1, freq1 } = gpCreateHistData(x,50);
{ v2, m2, freq2, freq2p } = gpCreateHistPData(x,50);

// Set data in gpdata structure
struct gpData gdat;
gdat = gpInitPlotData("X"$|"Y");
call gpAddZone(&gdat,"",aconcat(m1,freq1,3));
call gpAddZone(&gdat,"",aconcat(m2,freq2p,3));

// Write data file
ret = gpWritePlotData(&gdat,"xylineplot.plt");
if (ret);
    print gpGetErrorMessage(ret);
endif;

// Set data file
call gpSetDataFile(&gp,0,"xylineplot.plt");

// Set up frame 1
call gpAddHistFrame(&gp);
call gpSetLineMapZone(&gp,1,1,1);
call gpMoveFrame(&gp,1,0.25,0.25,5.175,7.85);

// Set up frame 2
call gpAddHistPFrame(&gp);
call gpSetLineMapZone(&gp,2,1,2);
call gpSetBarColor(&gp,2,1,"blue","blue");
call gpMoveFrame(&gp,2,5.525,0.25,5.175,7.85);
```

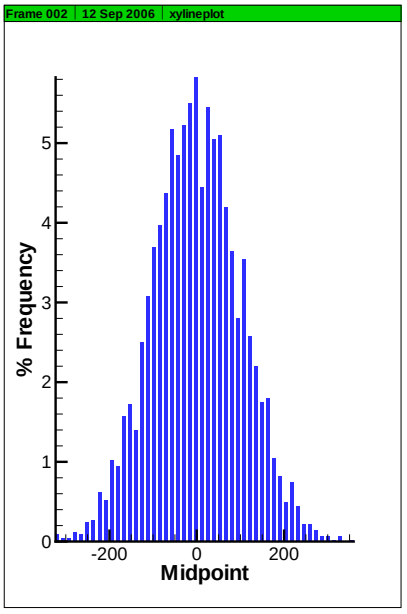


Figure 1.11: Multiple-Frame Hist/HistP Plot

```
// Plot the graph
call gpPlot(&gp);
```

1.6 Categorization of Commands

A **GAUSSplot** graph is created from the information contained in a **gpPlotControl** structure. **GAUSSplot** contains commands for creating a **gpPlotControl** structure and initializing its members to default values for a particular plot type, specifying line maps for line plots and axis maps for field plots, setting various output and view options, and plotting the graph.

Following the basic structure of a graph that was outlined in Section 1.4, commands may be

categorized into five different groups, depending on what unit of the graph they operate on. The five categories are as follows:

Commands that operate on an entire graph:

Example Command	Purpose
gpBarPlotCreate	Creates a **gpPlotControl** structure and initializes its members to default values for a bar plot.
gpPlot	Plots a graph using information contained in a **gpPlotControl** structure.

Commands that operate on a frame:

Example Command	Purpose
gpAddTextObject	Adds a text object to a specified location in a frame.
gpMoveFrame	Moves the frame to a specified location in the workspace.
gpSetXAxisTitle	Sets the title of the X-axis in a frame.

Commands that operate on a line map (for line plots):

Example Command	Purpose
gpSetBarColor	Sets the line color and fill color for the bars in a line map.
gpSetLineMaps	Sets the variables that are to be associated with each axis for one or more line maps.

Commands that operate on a field zone (for field plots):

Example Command	Purpose
gpSetMeshColor	Sets the color of the mesh in a field zone.
gpSetScatterSize	Sets the size of the symbols for the scatter layer in a field zone.

For 2D and 3D Contour plots, all contour options are set within a contour group. Each field zone for which the contour layer is activated must be tied to a particular contour group. Up to four contour groups may be used in any one frame, and one contour group may be used to set options for multiple field zones in a frame. Thus the final category consists of:

Commands that operate on a contour group (for 2D and 3D Contour plots):

Example Command	Purpose
gpAddContourLegend	Adds a legend to display the values associated with each color used in the contour plot.
gpSetContourVariable	Sets the variable to be used for contour plotting.

1.7 Working With Structure Pointers

All of the information for a **GAUSSplot** graph is stored in a **gpPlotControl** structure, which you must create and initialize at the beginning of your program. You can create a **gpPlotControl** structure with the following line:

```
struct gpPlotControl gp;
```

To initialize a plot control structure, you may call **gpPlotCreate** or one of the plot type specific **gp***PlotType***PlotCreate** commands. For example:

```
gp = gpXYLinePlotCreate;
```

initializes *gp* using default values for an XY Line plot.

GAUSSplot commands are implemented using structure pointers, which hold the address of a structure in the **GAUSS** symbol table. You may input a pointer to a structure into a command by placing an '&' in front of the structure name, indicating that you are passing in the address of that structure. For example:

```
call gpSetLineColor(&gp,1,1,"blue");
```

Or you may create a structure pointer and set it to point at the plot control structure created above with the following lines of code:

```
struct gpPlotControl *pgp;
pgp = &gp;
```

In the first line, placing an asterisk ('*') in front of *pgp* tells **GAUSS** to make *pgp* a pointer to a plot control structure. The second line sets *pgp* to the address of *gp*, making it a pointer to *gp*. If you create a structure pointer this way, then you may input it into a **GAUSSplot** command like this:

```
call gpSetLineColor(pgp,1,1,"blue");
```

The **GAUSSplot** data file creation commands are also implemented using structure pointers; however, they operate on **gpData** structures, rather than plot control structures. See Section 1.3.2 for an example of using structure pointers in these commands.

Structure pointers are not yet fully implemented in **GAUSS**. Enough functionality exists to support the **GAUSSplot** API. However, you may run into problems if you try to use them in ways that are not documented here.

1.8 Structure Files

When you write a data file with **gpWritePlotData**, or a data file is created automatically with one of the **gpMake***PlotType***Plot** commands, then the information in the **gpData** structure containing the data is saved to a file. This file has the same name as the associated data file, except with an `.fsr` extension instead of a `.plt` extension.

This structure file is accessed by **gpAddFieldZoneLegendFrame** as well as by **GAUSSplot** commands that allow you to specify variables using variable names. These commands open the `.fsr` file to obtain needed information or to convert the input variable names to indices of the variables into the data file. If you want to add a field zone legend or specify variables using variable names in **GAUSSplot** commands, then you need to keep the `.fsr` structure file together with your `.plt` data file at all times.

Call **gpCleanUp** after running your program to delete the structure and data files.

1.9 Error Handling

GAUSSplot supports three levels of error handling:

1 Return error codes if errors are encountered (default).
2 Log error messages and return error codes if errors are encountered.
3 Log error message and stop program run if an error is encountered.

Call **gpSetErrorLevel** to set the level of error handling in your program.

If you use the error handling default and your program encounters errors, you may call **gpGetErrorMessage** to retrieve the associated error messages.

If all of the commands in your program return with a success code, but your graph either does not appear when you run your program or is displayed with errors, then set the global variable **_gp_debug** to 1 in your program to display debugging information.

1.10 Examples

Here are some complete example programs that create graphs in **GAUSSplot**. See the **GAUSSplot** Cookbook for line-by-line explanation of some of the examples.

1.10.1 Line Plots

Example 1

This example creates a graph with one frame containing an XY Line plot with only one line map.

This example can be found in the file `xylineplot1.e` in the `gaussplot/examples` subdirectory of your **GAUSS** installation directory.

```
// Set up library and include file with structure definitions
```

```
library gaussplot;
#include gp.sdf

// Create gpPlotControl structure and set members to default
// values
struct gpPlotControl gp;
gp = gpXYLinePlotCreate;

// Generate data
x = seqa(0.2,.2,15);
y = log(x/5);

// Set the names of macro and data files to create
gpSetMacroFile(&gp,"xylineplot1.mcr");
ret = gpSetDataFile(&gp,0,"xylineplot1.plt");

// Create graph
ret = gpMakeXYLinePlot(&gp,x,y,"");
```

Example 2

This example creates a graph with one frame containing an XY Line plot. The vector in *x* is plotted against each of the three columns in *y*, making three separate line maps. Each line map is displayed in a different color. The axes titles are set and a line legend added before the graph is plotted.

This example can be found in the file `xylineplot2.e` in the `gaussplot/examples` subdirectory of your **GAUSS** installation directory.

```
// Set up library and include file with structure definitions
library gaussplot;
#include gp.sdf

// Create gpPlotControl structure and set members to default
// values
```

```
struct gpPlotControl gp;
gp = gpXYLinePlotCreate;

// Generate data
x = seqa(0.2,.2,15);
y = log(x/5)~ln(x/5)~exp(x/5);

// Set axes titles
ret = gpSetXAxisTitle(&gp,1,1,"x");
ret = gpSetYAxisTitle(&gp,1,1,"f(x)");

// Add line legend
ret = gpAddLineLegend(&gp,1);
ret = gpMoveLineLegend(&gp,1,85,40);

// Set the names of macro and data files to create
gpSetMacroFile(&gp,"xylineplot2.mcr");
ret = gpSetDataFile(&gp,0,"xylineplot2.plt");

// Create graph
ret = gpMakeXYLinePlot(&gp,x,y,"");
```

Example 3

This example creates a graph with three frames, each containing a different XY Line plot. Since the plot type for the entire graph is set to XY Line plot at the beginning of the program with the call to **gpXYLinePlotCreate**, there is no need to add frames using **gpAddXYLineFrame**. Instead the frames are created implicitly in the calls to **gpSetLineMaps**.

A call to **gpSetLineMaps** will create the specified frame if it does not already exist and then specify which variables are to be associated with the X and Y axes for one or more line maps in the frame. In this example, the first call to **gpSetLineMaps** creates five line maps (which will be displayed as five distinct lines) in the first frame, the second call creates three line maps in the second frame, and the third call creates four line maps in the third frame.

Since the graph contains multiple frames, **gpPlot** must be used to plot the graph. For a graph with

only one frame containing an XY Line plot, you may use **gpMakeXYLinePlot** to write the data file and plot the graph.

This example can be found in the file xylineplot3.e in the gaussplot/examples subdirectory of your **GAUSS** installation directory.

```
// Set up library and include file with structure definitions
library gaussplot;
#include gp.sdf

// Create gpPlotControl structure and set members to default
// values
struct gpPlotControl gp;
gp = gpXYLinePlotCreate;

// Create gpData structure
struct gpData gdat;

// Generate data
// X-axis variable
x = seqa(.2,.2,20);

// Y-axis variables
y = log(x[.,1]/5);
y2 = ln(x[.,1]/5);
y3 = exp(x[.,1]/5);
x = x~y;
x = x~y*0.8;
x = x~y*0.6;
x = x~y*0.4;
x = x~y*0.2;
x = x~y2;
x = x~y2*0.6;
x = x~y2*0.2;
x = x~y3;
x = x~y3*0.8;
x = x~y3*0.6;
```

```
x = x~y3*0.4;

// List variable names
string vnames = { "X1", "Y1", "Y2", "Y3", "Y4", "Y5", "Y6", "Y7",
                  "Y8", "Y9", "Y10", "Y11", "Y12" };

// Set data in gpData structure
gdat = gpSetPlotData(x,vnames);

// Write data file
ret = gpWritePlotData(&gdat,"xylineplot3.plt");
if (ret);
     print gpGetErrorMessage(ret);
endif;

// Set error level to 2 - print error messages if errors are
// encountered
call gpSetErrorLevel(&gp,2);

// Specify names for macro file to be created and data file to be
// opened
gpSetMacroFile(&gp,"xylineplot3.mcr");
call gpSetDataFile(&gp,0,"xylineplot3.plt");

// Specify line maps for all three frames
string linemaps1 = { "X1" "Y1", "X1" "Y2", "X1" "Y3", "X1" "Y4",
                     "X1" "Y5" };
call gpSetLineMaps(&gp,1,linemaps1);
string linemaps2 = { "X1" "Y6", "X1" "Y7", "X1" "Y8" };
call gpSetLineMaps(&gp,2,linemaps2);
string linemaps3 = { "X1" "Y9", "X1" "Y10", "X1" "Y11", "X1"
                     "Y12" };
call gpSetLineMaps(&gp,3,linemaps3);

// Show symbol layer and change symbol options
call gpShowSymbolLayer(&gp,1|2|3,1);
call gpSetSymbolSkip(&gp,1,1|3|5,2,2);
```

```
call gpSetSymbolSkip(&gp,1,2|4,3,2);
call gpSetSymbolSkip(&gp,2,seqa(1,1,3),15,3);
call gpSetSymbolSkip(&gp,3,seqa(1,1,4),2,2);
call gpSetSymbolSize(&gp,1,seqa(1,1,5),2);
call gpSetSymbolSize(&gp,2,seqa(1,1,3),3);
call gpSetSymbolSize(&gp,3,seqa(1,1,4),2);
call gpSetSymbolShape(&gp,1,2,"circle");
call gpSetSymbolShape(&gp,1,3,"rtri");
call gpSetSymbolShape(&gp,1,4,"diamond");
call gpSetSymbolShape(&gp,1,5,"ltri");
call gpSetSymbolShape(&gp,2,2,"circle");
call gpSetSymbolShape(&gp,2,3,"rtri");
call gpSetSymbolShapeASCII(&gp,3,1,1,"$");
call gpSetSymbolShapeASCII(&gp,3,2,3,"r");
call gpSetSymbolShapeASCII(&gp,3,3,2,"p");
call gpSetSymbolShapeASCII(&gp,3,4,2,"x");
call gpSetSymbolColor(&gp,1,5,"purple");
call gpSetSymbolLineThickness(&gp,2,1,0.8);
call gpSetSymbolFillMode(&gp,2,2,2,"");
call gpSetSymbolFillMode(&gp,2,3,4,"CUSTOM34");

// Change line options
call gpSetLineColor(&gp,1,5,"purple");
call gpSetLineThickness(&gp,3,1|4,0.8);
call gpSetLinePattern(&gp,3,1|2,"dashdot");
call gpSetLinePatternLength(&gp,3,1,6);

// Resize and move frames to show all three in window
call gpMoveFrame(&gp,1,1,0.25,4.425,3.925);
call gpMoveFrame(&gp,2,5.575,0.25,4.425,3.925);
call gpMoveFrame(&gp,3,1,4.325,9,3.925);

// Specify titles of axes
call gpSetXAxisTitle(&gp,1,1,"X AXIS");
call gpSetYAxisTitle(&gp,1,1,"Y AXIS");
call gpSetXAxisTitle(&gp,2,1,"X AXIS");
call gpSetYAxisTitle(&gp,2,1,"Y AXIS");
```

```
call gpSetXAxisTitle(&gp,3,1,"X");
call gpSetYAxisTitle(&gp,3,1,"F(X)");

// Add titles for each frame
call gpAddTextObject(&gp,1,"title1",44,89,1,"log(x/5)");
call gpAddTextObject(&gp,2,"title2",44,89,1,"ln(x/5)");
call gpAddTextObject(&gp,3,"title3",46,89,1,"exp(x/5)");

// Add text box in frame 3
call gpAddTextObject(&gp,3,"text1",32,56,1,
                        "using math/Greek characters as symbols");
call gpSetTextObjectFont(&gp,3,"text1","times",13,2);
call gpSetTextObjectColor(&gp,3,"text1","custom41");
call gpSetTextObjectBoxType(&gp,3,"text1",2);
call gpSetTextObjectBoxColor(&gp,3,"text1","custom41","");
call gpSetTextObjectMargin(&gp,3,"text1",40);
call gpRotateTextObject(&gp,3,"text1",15);

// Show each frame one at a time and pause between each frame
gpShowDrawing(&gp,2);
call gpFramePause(&gp,1|2,0.1);

// Create graph
call gpPlot(&gp);
```

1.10.2 Field Plots

Example 1

This example creates a graph with one frame, containing a 3D Contour plot. Since the graph has only one frame, containing one field zone, **gpMake3DContourPlot** is used to write the data to a data file and plot the graph. For a more complicated graph, you must first write the data file using **gpWritePlotData** and then call **gpPlot** to plot the graph.

This example can be found in the file `contourplot1.e` in the `gaussplot/examples`

subdirectory of your **GAUSS** installation directory.

```
// Set up library and include file with structure definitions
library gaussplot;
#include gp.sdf

// Create gpPlotControl structure and set members to default
// values
struct gpPlotControl gp;
gp = gp3DContourPlotCreate;

// Generate data
xl = 51;
yl = 51;
pix = pi/(xl-1);
piy = pi/(yl-1);
x = seqa(-88,2.5,xl)'*pix;
y = seqa(-88,2.5,yl)*piy;
z = sin(x).*cos(x-y);
cv = cos(x).*sin(x-y);
x = x.*ones(51,51);
y = y.*ones(51,51);

// List variable names
string vnames = { "X", "Y", "Z", "CV" };

// Specify names for macro file to be created and data file to be
// opened
gpSetMacroFile(&gp,"contourplot1.mcr");
ret = gpSetDataFile(&gp,0,"contourplot1.plt");

// Add title for frame
ret = gpAddTextObject(&gp,1,"",43,93,1,"3D Contour Plot");

// Create contour plot
ret = gpMake3DContourPlot(&gp,x,y,z,cv,vnames);
```

Example 2

This example creates a graph with one frame, containing a 3D Cartesian plot. The plot is a cylindrical solid, which is created by using 3-dimensional variables.

This example can be found in the file `volumeplot1.e` in the `gaussplot/examples` subdirectory of your **GAUSS** installation directory.

```
// Set up library and include file with structure definitions
library gaussplot;
#include gp.sdf

// Create gpPlotControl structure and set members to default
// values
struct gpPlotControl gp;
gp = gp3DCartesianPlotCreate;

// Generate data
a = seqa(0,pi/10,21)';
r = seqa(.25,0.75,4);
x = r.*cos(a);
x = areshape(x,5|4|21);
y = r.*sin(a);
y = areshape(y,5|4|21);
r = r|(r[rows(r)]+0.75);

z = areshape(2*r[1].*ones(4,21),4|21);
for i(2,5,1);
     z = aconcat(z,2*r[i].*ones(4,21),3);
endfor;

// Show only shade layer and use translucency
ret = gpShowMeshLayer(&gp,1,0);
ret = gpShowBoundaryLayer(&gp,1,0);
ret = gpShowShadeLayer(&gp,1,1);
ret = gpSetShadeColor(&gp,1,1,"blue");
```

```
ret = gpUseTranslucencyEffect(&gp,1,1);
ret = gpSetTranslucency(&gp,1,1,30);

// Specify names for macro and data files to be created
gpSetMacroFile(&gp,"volumeplot1.mcr");
ret = gpSetDataFile(&gp,0,"volumeplot1.plt");

// Plot the graph
ret = gpMake3DCartesianPlot(&gp,x,y,z,"");
```

1.10.3 Graphs Containing Different Plot Types

This example creates a graph with three frames: a large frame containing a 2D Contour plot with a contour legend, an inset frame containing a 3D Cartesian plot and an inset frame containing an XY Line plot. Since the generic **gpPlotCreate** command is used to initialize the plot control structure, there is no plot type set for the entire graph. This allows for the creation of multiple frames, each containing different plot types. In this example, the frames are created with **gpAdd2DContourFrame**, **gpAdd3DCartesianFrame**, and **gpAddXYLineFrame**.

This example can be found in the file `plot1.e` in the `gaussplot/examples` subdirectory of your **GAUSS** installation directory.

```
// Set up library and include file with structure definitions
library gaussplot;
#include gp.sdf

// Create gpPlotControl structure and set members to default values
struct gpPlotControl gp;
gp = gpPlotCreate;

// Create gpData structure
struct gpData gdat;

// Generate data
xl = 51;
```

```
yl = 51;
pix = pi/(xl-1);
piy = pi/(yl-1);
x = seqa(-88,2.5,xl)'*pix;
y = seqa(-88,2.5,yl)*piy;

// Z-axis variable
z = sin(x).*cos(x-y); // For contour and cartesian plots
z2 = zeros(20,1); // For xy line plot
z3 = zeros(60,1); // For xy line plot

// X and Y-axis variables
x = x.*ones(51,51);
x2 = seqa(0.2,0.2,20);
x3 = seqa(0.05,0.05,60);
y = y.*ones(51,51);
y2 = log(x2[.,1]);
y3 = ln(x3[.,1]);

// Put data into an array
a = aconcat(x,y,3);
a = aconcat(a,z,3);
a2 = aconcat(x2,y2,3);
a2 = aconcat(a2,z2,3);
a3 = aconcat(x3,y3,3);
a3 = aconcat(a3,z3,3);

// List variable names
string vnames = { "X", "Y", "Z" };

// Set data in gpData structure
gdat = gpInitPlotData(vnames);
ret = gpAddZone(&gdat,"Zone One",a);
ret = gpAddZone(&gdat,"Zone Two",a2);
ret = gpAddZone(&gdat,"Zone Three",a3);

// Write data file
```

```
ret = gpWritePlotData(&gdat,"plot1.plt");

// Specify names for macro file to be created and data file to be
// opened
gpSetMacroFile(&gp,"plot1.mcr");
ret = gpSetDataFile(&gp,0,"plot1.plt");

// Add a 2D Contour frame, a 3D Cartesian frame, and an XY Line
// frame
gpAdd2DContourFrame(&gp);
gpAdd3DCartesianFrame(&gp);
gpAddXYLineFrame(&gp);

// Use only field zone one for contour and cartesian plots
// And use field zones two and three for XY line plot
ret = gpSetActiveFieldZones(&gp,1,1);
ret = gpSetActiveFieldZones(&gp,2,1);
ret = gpSetLineMapZone(&gp,3,1,2);
ret = gpSetLineMapZone(&gp,3,2,3);

// Change color distribution to continuous for contour frame
ret = gpSetContourColorDistribution(&gp,1,1,2);

// Set line map options
ret = gpSetLineColor(&gp,3,2,"custom33");
ret = gpShowSymbolLayer(&gp,3,1);
ret = gpSetSymbolSkip(&gp,3,1,2,2);
ret = gpSetSymbolSkip(&gp,3,2,5,2);
ret = gpSetSymbolSize(&gp,3,1|2,2);
ret = gpSetSymbolShape(&gp,3,1,"square");
ret = gpSetSymbolShape(&gp,3,2,"diamond");
ret = gpSetSymbolColor(&gp,3,2,"custom33");
ret = gpSetSymbolFillMode(&gp,3,1|2,2,"");

// Resize and move frames to show all three in window
ret = gpMoveFrame(&gp,1,0.25,0.25,10.5,8);
ret = gpMoveFrame(&gp,2,2,4.6,3.5,2.5);
```

```
ret = gpMoveFrame(&gp,3,5.75,1.25,3.5,2.5);

// Make X and Y axes independent of one another in frame 1
// And make axes XYZ dependent in frame 2
ret = gpMakeAxesIndependent(&gp,1);
ret = gpMakeAxesXYZDependent(&gp,2,1,0.5);

// Add title for frame 1
ret = gpAddTextObject(&gp,1,"",32,91,1,
                      (chrs(ones(1,21)*32)$+"2D Contour Plot")$|
                      "with 3D Cartesian and XY Line Plot Insets");

// Add contour legend for contour group 1
ret = gpAddContourLegend(&gp,1,1);
ret = gpMoveContourLegend(&gp,1,1,85,49);
ret = gpSetContourLegendBoxStyle(&gp,1,1,2,0.2,1,2);
ret = gpSetContourLegendSkip(&gp,1,1,2);
ret = gpSetContourLegendMargin(&gp,1,1,5);

// Do not show border or axes in frame 2
// And do not show border or background in frame 3
ret = gpShowXAxis(&gp,2,0,0);
ret = gpShowYAxis(&gp,2,0,0);
ret = gpShowZAxis(&gp,2,0);
ret = gpSetFrameStyle(&gp,2,0,0,1);
ret = gpSetFrameStyle(&gp,3,0,0,0);

// Create contour plot
ret = gpPlot(&gp);
```

See the `gaussplot/examples` subdirectory of your **GAUSS** installation directory for more examples.

2 Command Overview

Following is a complete list of the commands and global variables contained in **GAUSSplot**.

2.1 Commands

2.1.1 Setup and Initialization

gp2DCartesianPlotCreate	Creates a plot control structure and initializes its members to default values for a 2D Cartesian plot.
gp2DContourPlotCreate	Creates a plot control structure and initializes its members to default values for a 2D Contour plot.
gp3DCartesianPlotCreate	Creates a plot control structure and initializes its members to default values for a 3D Cartesian plot.
gp3DContourPlotCreate	Creates a plot control structure and initializes its members to default values for a 3D Contour plot.

gpBarPlotCreate	Creates a plot control structure and initializes its members to default values for a Bar plot.
gpBoxPlotCreate	Creates a plot control structure and initializes its members to default values for a Box plot.
gpHistPlotCreate	Creates a plot control structure and initializes its members to default values for a Hist plot.
gpHistFPlotCreate	Creates a plot control structure and initializes its members to default values for a HistF plot.
gpHistPPlotCreate	Creates a plot control structure and initializes its members to default values for a HistP plot.
gpLogLogPlotCreate	Creates a plot control structure and initializes its members to default values for a LogLog plot.
gpLogXPlotCreate	Creates a plot control structure and initializes its members to default values for a LogX plot.
gpLogYPlotCreate	Creates a plot control structure and initializes its members to default values for a LogY plot.
gpPlotCreate	Creates a plot control structure and initializes its members to generic default values for a graph with frames containing different plot types.
gpPolarLinePlotCreate	Creates a plot control structure and initializes its members to default values for a Polar Line plot.
gpXYLinePlotCreate	Creates a plot control structure and initializes its members to default values for an XY Line plot.

2.1.2 Creating Data Files

gpAddZone	Adds all of the variables for one zone of the data file to the data structure.

gpCreateHistData	Generates vectors of category breakpoints, category midpoints, and frequency counts for a Hist plot (frequency histogram).
gpCreateHistPData	Generates vectors of category breakpoints, category midpoints, frequency counts, and frequency percentages for a HistP plot (percentage histogram).
gpInitPlotData	Creates and initializes a data structure using user-specified variable names.
gpSetDataSetTitle	Sets the title of a data set.
gpSetPlotData	Creates, initializes and adds data to a data structure for a data file that will contain only one zone with *K* variables that are N×1.
gpWritePlotData	Writes a `.plt` data file using the data contained in a data structure.

2.1.3 Specifying Data and Macro Files

gpSetDataFile	Specifies the data file(s) to use.
gpSetMacroFile	Specifies the name of the macro file to write.

2.1.4 Adding Frames to a Graph

gpAdd2DCartesianFrame	Add a 2D Cartesian plot frame.
gpAdd2DContourFrame	Add a 2D Contour plot frame.
gpAdd3DCartesianFrame	Add a 3D Cartesian plot frame.
gpAdd3DContourFrame	Add a 3D Contour plot frame.
gpAddBarFrame	Add a Bar plot frame.
gpAddBoxFrame	Add a Box plot frame.

gpAddHistFrame	Add a Hist plot frame.
gpAddHistFFrame	Add a HistF plot frame.
gpAddHistPFrame	Add a HistP plot frame.
gpAddLogLogFrame	Add a LogLog plot frame.
gpAddLogXFrame	Add a LogX plot frame.
gpAddLogYFrame	Add a LogY plot frame.
gpAddPolarLineFrame	Add a Polar Line plot frame.
gpAddXYLineFrame	Add an XY Line plot frame.

2.1.5 Setting Line Map Options

Line Map Specification

gpSetLineMapAxes	Sets the X and Y axes to be used for the specified line map(s).
gpSetLineMapNames	Sets names for all of the line maps in a frame.
gpSetLineMaps	Sets all of the line maps for a frame.
gpSetLineMapZone	Sets the zone in the data file to be used for the specified line map(s).

Line Layer Options

gpSetLineColor	Sets the color of the line(s) in the specified line map(s).
gpSetLinePattern	Sets the pattern of the line(s) in the specified line map(s).

gpSetLinePatternLength	Sets the length of the pattern of the line(s) in the specified line map(s).
gpSetLineThickness	Sets the thickness of the line(s) in the specified line map(s).
gpShowLineLayer	Activates or deactivates the line layer in the specified frame(s).
gpShowLineMapLineLayer	Activates or deactivates the line layer for the specified line map(s).

Curve Fit Options

gpSetCurveFitDataFile	Sets the file(s) to which curve fit details and data are written.
gpSetCurveFitDependentVariable	Sets the dependent variable for one or more curve fits.
gpSetCurveFitPoints	Sets the number of points on the curve(s) in the specified line map(s).
gpSetCurveFitPolynomialOrder	Sets the order of the polynomial function to be used for one or more curve fits.
gpSetCurveFitRange	Sets the range of the independent variable for the curve(s) in the specified line map(s).
gpSetCurveFitSplineClamp	Sets the derivative at the beginning and ending points of one or more spline fits.
gpSetCurveFitType	Sets the curve fit type of the line(s) in the specified line map(s).
gpSetCurveFitWeightVariable	Specifies a weight variable for one or more curve fits.
gpSetGeneralCurveFitOptions	Sets options for one or more general curve fits.

Symbol Layer Options

gpSetSymbolColor	Sets the color of the symbols in the specified line map(s).
gpSetSymbolFillMode	Sets the fill mode of the symbols in the specified line map(s).
gpSetSymbolLineThickness	Sets the line thickness of the symbols in the specified line map(s).
gpSetSymbolShape	Sets the shape of the symbols in the specified line map(s).
gpSetSymbolShapeASCII	Sets the shape of the symbols in the specified line map(s) to an ASCII character.
gpSetSymbolSize	Sets the size of the symbols in the specified line map(s).
gpSetSymbolSkip	Sets the an option to show symbols at only a subset of the data points in the specified line map(s).
gpShowLineMapSymbolLayer	Activates or deactivates the symbol layer for the specified line map(s).
gpShowSymbolLayer	Activates or deactivates the symbol layer in the specified frame(s).

Bar Layer Options

gpSetBarColor	Sets the color of the bars in the specified line map(s).
gpSetBarLineThickness	Sets the thickness of the bar outlines in the specified line map(s).
gpSetBarOrientation	Sets the orientation of the bars in the specified line map(s).
gpSetBarWidth	Sets the width of the bars in the specified line map(s).

gpShowBarLayer	Activates or deactivates the bar layer in the specified frame(s).
gpShowLineMapBarLayer	Activates or deactivates the bar layer for the specified line map(s).

Error Bar Layer Options

gpSetErrorBarColor	Sets the color of the error bars in the specified line map(s).
gpSetErrorBarLineThickness	Sets the line thickness of the error bars in the specified line map(s).
gpSetErrorBarSize	Sets the length of the crossbar for the error bars in the specified line map(s).
gpSetErrorBarSkip	Sets an option to show error bars at only a subset of the data points in the specified line map(s).
gpSetErrorBarType	Sets the type of the error bars in the specified line map(s).
gpSetErrorBarVariable	Sets the variable to be used for the error bars in the specified line map(s).
gpShowErrorBarLayer	Activates or deactivates the error bar layer in the specified frame(s).
gpShowLineMapErrorBarLayer	Activates or deactivates the error bar layer for the specified line map(s).

Percentile Box Symbol Options for Box plots

gpSetBoxColor	Sets the color of the percentile box symbol for the specified line map(s).

gpSetBoxMinMaxMethod	Sets the method by which the maximum and minimum values for the percentile box symbols in the specified frame are calculated.
gpSetBoxWidth	Sets the width of the percentile box symbols for the line maps in the specified frame.

Adding Line Legends

gpAddLineLegend	Adds a line legend to a frame.
gpMoveLineLegend	Moves a line legend to a specified location in the frame.
gpSetLineLegendBoxColor	Sets the colors of the outline and background of the box surrounding a line legend.
gpSetLineLegendBoxStyle	Sets the style of the box surrounding a line legend.
gpSetLineLegendBoxThickness	Sets the thickness of the outline of the box surrounding a line legend.
gpSetLineLegendBoxType	Sets the type of the box surrounding a line legend.
gpSetLineLegendFont	Sets the font used in a line legend.
gpSetLineLegendLineSpacing	Sets the spacing between the lines of text in a line legend.
gpSetLineLegendMargin	Sets the size of the margin between the line legend text and the surrounding box.

2.1.6 Setting Field Zone Options

Field Zone Specification

`gpSetActiveFieldZones`	Specifies which field zones are to be displayed.
`gpSetAxisMap`	Sets the variable to be associated with each axis in a frame.
`gpSetFieldZoneNames`	Sets names for all of the field zones in a frame.

Mesh Layer Options

`gpSetMeshColor`	Sets the color of the mesh in the specified field zone(s).
`gpSetMeshLinePattern`	Sets the mesh line pattern in the specified field zone(s).
`gpSetMeshLinePatternLength`	Sets the length of the mesh line pattern in the specified field zone(s).
`gpSetMeshLineThickness`	Sets the line thickness of the mesh in the specified field zone(s).
`gpSetMeshType`	Sets the type of the mesh in the specified field zone(s).
`gpShowFieldZoneMeshLayer`	Activates or deactivates the mesh layer for the specified field zones(s).
`gpShowMeshLayer`	Activates or deactivates the mesh layer in the specified frame(s).

Contour Layer Options

gpAddContourLegend	Adds a contour legend for the specified contour group(s).
gpMoveContourLegend	Moves a contour legend to a specified location in the frame.
gpSetContourColorDistribution	Sets the color map distribution of contour flooding for the specified contour group(s).
gpSetContourFloodGroup	Sets the contour group to be used for contour flooding in the specified field zone(s).
gpSetContourLegendBoxColor	Sets the colors of the outline and background of the box surrounding a contour legend.
gpSetContourLegendBoxStyle	Sets the style of the box surrounding a contour legend.
gpSetContourLegendBoxThickness	Sets the thickness of the outline of the box surrounding a contour legend.
gpSetContourLegendBoxType	Sets the type of the box surrounding a contour legend.
gpSetContourLegendHeaderFont	Sets the font used for the header text in a contour legend.
gpSetContourLegendLabelFont	Sets the font used for the labels in a contour legend.
gpSetContourLegendLineSpacing	Sets the spacing between the lines of text in a contour legend.
gpSetContourLegendMargin	Sets the size of the margin between the contour legend text and the surrounding box.

gpSetContourLegendOrientation	Sets the orientation of a contour legend.
gpSetContourLegendSkip	Sets an option to show only a subset of the contour levels in a contour legend.
gpSetContourLegendView	Sets view options for a contour legend.
gpSetContourLineColor	Sets the color of the contour lines in the specified field zone(s).
gpSetContourLineGroup	Sets the contour group to be used for contour lines in the specified field zone(s).
gpSetContourVariable	Sets the contour variable for the specified contour group(s).
gpShowContourLayer	Activates or deactivates the contour layer in the specified frame(s).
gpShowFieldZoneContourLayer	Activates or deactivates contour lines and flooding in the specified field zones(s).

Vector Layer Options

gpMakeVectorsTangent	Projects the vectors in the vector layer onto the 3D field zone surface.
gpSetVectorArrowHeadType	Sets the arrow head type for the vectors in the specified field zone(s).
gpSetVectorColor	Sets the color of the vectors in the specified field zone(s).
gpSetVectorLinePattern	Sets the line pattern of the vectors in the specified field zone(s).
gpSetVectorLinePatternLength	Sets the length of the vector line pattern in the specified field zone(s).

gpSetVectorLineThickness	Sets the line thickness of the vectors in the specified field zone(s).
gpSetVectorType	Sets the type of the vectors in the specified field zone(s).
gpSetVectorVariables	Sets the variables to use as the vector components for the vectors in the specified field zone(s).
gpShowFieldZoneVectorLayer	Activates or deactivates the vector layer for the specified field zones(s).
gpShowVectorLayer	Activates or deactivates the vector layer in the specified frame(s).

Scatter Layer Options

gpSetScatterColor	Sets the color of the scatter symbols in the specified field zone(s).
gpSetScatterFillMode	Sets the fill mode of the scatter symbols in the specified field zone(s).
gpSetScatterLineThickness	Sets the line thickness of the scatter symbols in the specified field zone(s).
gpSetScatterShape	Sets the shape of the scatter symbols in the specified field zone(s).
gpSetScatterShapeASCII	Sets the shape of the scatter symbols in the specified field zone(s) to an ASCII character.
gpSetScatterSize	Sets the size of the scatter symbols in the specified field zone(s).
gpShowFieldZoneScatterLayer	Activates or deactivates the scatter layer for the specified field zones(s).

gpShowScatterLayer	Activates or deactivates the scatter layer in the specified frame(s).

Shade Layer Options

gpSetShadeColor	Sets the color of the shading in the specified field zone(s).
gpShowFieldZoneShadeLayer	Activates or deactivates the shade layer for the specified field zones(s).
gpShowShadeLayer	Activates or deactivates the shade layer in the specified frame.

Boundary Layer Options

gpSetBoundaryColor	Sets the color of the boundary line in the specified field zone(s).
gpSetBoundaryThickness	Sets the thickness of the boundary line in the specified field zone(s).
gpShowBoundaryLayer	Activates or deactivates the boundary layer in the specified frame(s).
gpShowFieldZoneBoundaryLayer	Activates or deactivates the boundary layer for the specified field zones(s).

Lighting Effect Options

gpSetLightingEffect	Sets the lighting effect to use in the specified field zone(s).
gpUseFieldZoneContourLighting	Activates or deactivates contour lighting for the specified field zone(s).

gpUseFieldZoneShadeLighting	Activates or deactivates shade lighting for the specified field zone(s).
gpUseLightingEffect	Activates or deactivates the lighting effect in the specified frame(s).

Translucency Effect Options

gpSetTranslucency	Sets the percentage of translucency in the specified field zone(s).
gpUseFieldZoneTranslucency	Activates or deactivates the translucency effect for the specified field zone(s).
gpUseTranslucencyEffect	Activates or deactivates the translucency effect in the specified frame(s).

Adding Field Zone Legends

gpAddFieldZoneLegendFrame	Adds a field zone legend frame as an inset in the specified frame.
gpMoveFieldZoneLegend	Moves a field zone legend frame to a specified location in the parent frame.
gpSetFieldZoneLegendBoxColor	Sets the colors of the outline and background of the box surrounding a field zone legend.
gpSetFieldZoneLegendBoxStyle	Sets the style of the box surrounding a field zone legend.
gpSetFieldZoneLegendBoxThickness	Sets the thickness of the outline of the box surrounding a field zone legend.
gpSetFieldZoneLegendBoxType	Sets the type of the box surrounding a field zone legend.

gpSetFieldZoneLegendFont	Sets the font used in a field zone legend.
gpSetFieldZoneLegendLineSpacing	Sets the spacing between the lines of text in a field zone legend.
gpSetFieldZoneLegendLowerMargin	Sets size of margin between the bottom of the field zone legend text and the bottom of the surrounding box.
gpSetFieldZoneLegendTitle	Sets a title for a field zone legend.
gpSetFieldZoneLegendTitleFont	Sets the font of the title in a field zone legend.
gpSetFieldZoneLegendUpperMargin	Sets size of margin between the top of the field zone legend text and the top of the surrounding box.

Surface Plotting Options For Volume Plots

gpSetIPlaneRange	Sets the range of I-planes to display in I, IJ, IK or IJK plane surface plotting.
gpSetJPlaneRange	Sets the range of J-planes to display in J, JK, IJ or IJK plane surface plotting.
gpSetKPlaneRange	Sets the range of K-planes to display in K, IK, JK or IJK plane surface plotting.
gpSetSurfacesToPlot	Specifies which surfaces are to be displayed for a 3D Cartesian (or 3D Contour) volume plot.

2.1.7 Setting Axes Options

X,Y,Z Axes Options

gpFillGridArea	Specifies whether or not the grid area is filled.

gpMakeAxesIndependent	Allows the axes to be sized independently.
gpMakeAxesXYDependent	Makes the X and Y axes dependent on each other, so they may not be sized independently.
gpMakeAxesXYZDependent	Makes the X, Y and Z axes dependent on one another, so they may not be sized independently.
gpPositionGrid	Positions the grid in the frame.
gpPositionXAxis	Positions an X-axis.
gpPositionYAxis	Positions a Y-axis.
gpSetGridAreaFillColor	Sets the fill color for the grid area in a frame.
gpSetLineMapAxes	Sets the X and Y axes to be used for the specified line map(s).
gpSetXAxisColor	Sets the color of an X-axis.
gpSetXAxisGridLineColor	Sets the color of the X-axis major gridlines.
gpSetXAxisGridLineThickness	Sets the thickness of the X-axis major gridlines.
gpSetXAxisGridPattern	Sets the pattern of the X-axis major gridlines.
gpSetXAxisGridPatternLength	Sets the length of the X-axis major gridline pattern.
gpSetXAxisLabelColor	Sets the color of X-axis tick labels.
gpSetXAxisLabelFont	Sets the type and size of the font used for X-axis tick labels.
gpSetXAxisLabelOffset	Sets the offset of X-axis tick labels from the axis line.

gpSetXAxisLabelSkip	Sets an option to show only a subset of X-axis tick labels.
gpSetXAxisMinorGridLineColor	Sets the color of the X-axis minor gridlines.
gpSetXAxisMinorGridLineThickness	Sets the thickness of the X-axis minor gridlines.
gpSetXAxisMinorGridPattern	Sets the pattern of the X-axis minor gridlines.
gpSetXAxisMinorGridPatternLength	Sets the length of the X-axis minor gridline pattern.
gpSetXAxisRange	Sets the range of an X-axis.
gpSetXAxisThickness	Sets the thickness of an X-axis line.
gpSetXAxisTicks	Sets the distance between major tick marks and the number of minor tick marks on an X-axis.
gpSetXAxisTitle	Sets the title for an X-axis.
gpSetXAxisTitleColor	Sets the color of an X-axis title.
gpSetXAxisTitleFont	Sets the type and size of the font used for an X-axis title.
gpSetXAxisTitleOffset	Sets the offset of an X-axis title from the axis line.
gpSetYAxisColor	Sets the color of a Y-axis.
gpSetYAxisGridLineColor	Sets the color of the Y-axis major gridlines.
gpSetYAxisGridLineThickness	Sets the thickness of the Y-axis major gridlines.
gpSetYAxisGridPattern	Sets the pattern of the Y-axis major gridlines.

gpSetYAxisGridPatternLength	Sets the length of the Y-axis major gridline pattern.
gpSetYAxisLabelColor	Sets the color of Y-axis tick labels.
gpSetYAxisLabelFont	Sets the type and size of the font used for Y-axis tick labels.
gpSetYAxisLabelOffset	Sets the offset of Y-axis tick labels from the axis line.
gpSetYAxisLabelSkip	Sets an option to show only a subset of Y-axis tick labels.
gpSetYAxisMinorGridLineColor	Sets the color of the Y-axis minor gridlines.
gpSetYAxisMinorGridLineThickness	Sets the thickness of the Y-axis minor gridlines.
gpSetYAxisMinorGridPattern	Sets the pattern of the Y-axis minor gridlines.
gpSetYAxisMinorGridPatternLength	Sets the length of the Y-axis minor gridline pattern.
gpSetYAxisRange	Sets the range of a Y-axis.
gpSetYAxisThickness	Sets the thickness of a Y-axis line.
gpSetYAxisTicks	Sets the distance between major tick marks and the number of minor tick marks on a Y-axis.
gpSetYAxisTitle	Sets the title for a Y-axis.
gpSetYAxisTitleColor	Sets the color of a Y-axis title.
gpSetYAxisTitleFont	Sets the type and size of the font used for a Y-axis title.
gpSetYAxisTitleOffset	Sets the offset of a Y-axis title from the axis line.

gpSetZAxisColor	Sets the color of the Z-axis.
gpSetZAxisGridLineColor	Sets the color of the Z-axis major gridlines.
gpSetZAxisGridLineThickness	Sets the thickness of the Z-axis major gridlines.
gpSetZAxisGridPattern	Sets the pattern of the Z-axis major gridlines.
gpSetZAxisGridPatternLength	Sets the length of the Z-axis major gridline pattern.
gpSetZAxisLabelColor	Sets the color of the Z-axis tick labels.
gpSetZAxisLabelFont	Sets the type and size of the font used for the Z-axis tick labels.
gpSetZAxisLabelOffset	Sets the offset of the Z-axis tick labels from the axis line.
gpSetZAxisLabelSkip	Sets an option to show only a subset of the Z-axis tick labels.
gpSetZAxisMinorGridLineColor	Sets the color of the Z-axis minor gridlines.
gpSetZAxisMinorGridLineThickness	Sets the thickness of the Z-axis minor gridlines.
gpSetZAxisMinorGridPattern	Sets the pattern of the Z-axis minor gridlines.
gpSetZAxisMinorGridPatternLength	Sets the length of the Z-axis minor gridline pattern.
gpSetZAxisRange	Sets the range of the Z-axis.
gpSetZAxisThickness	Sets the thickness of the Z-axis line.
gpSetZAxisTicks	Sets the distance between major tick marks and the number of minor tick marks on the Z-axis.

gpSetZAxisTitle	Sets the title for the Z-axis.
gpSetZAxisTitleColor	Sets the color of the Z-axis title.
gpSetZAxisTitleFont	Sets the type and size of the font used for the Z-axis title.
gpSetZAxisTitleOffset	Sets the offset of the Z-axis title from the axis line.
gpShow3DOrientationAxis	Activates or deactivates display of the 3-D orientation axis in a frame.
gpShowXAxis	Activates or deactivates display of an X-axis in a frame.
gpShowXAxisGridLines	Activates or deactivates display of X-axis major gridlines in a frame.
gpShowXAxisLabels	Activates or deactivates display of X-axis tick labels in a frame.
gpShowXAxisMinorGridLines	Activates or deactivates display of X-axis minor gridlines in a frame.
gpShowXAxisTicks	Activates or deactivates display of X-axis tick marks in a frame.
gpShowYAxis	Activates or deactivates display of a Y-axis in a frame.
gpShowYAxisGridLines	Activates or deactivates display of Y-axis major gridlines in a frame.
gpShowYAxisLabels	Activates or deactivates display of Y-axis tick labels in a frame.
gpShowYAxisMinorGridLines	Activates or deactivates display of Y-axis minor gridlines in a frame.
gpShowYAxisTicks	Activates or deactivates display of Y-axis tick marks in a frame.

gpShowZAxis	Activates or deactivates display of the Z-axis in a frame.
gpShowZAxisGridLines	Activates or deactivates display of Z-axis major gridlines in a frame.
gpShowZAxisLabels	Activates or deactivates display of Z-axis tick labels in a frame.
gpShowZAxisMinorGridLines	Activates or deactivates display of Z-axis minor gridlines in a frame.
gpShowZAxisTicks	Activates or deactivates display of Z-axis tick marks in a frame.
gpUseGridAreaLighting	Activates or deactivates light source shading in the grid area.
gpUseXAxisLogScaling	Activates or deactivates log scaling on an X-axis.
gpUseYAxisLogScaling	Activates or deactivates log scaling on a Y-axis.

Polar Axes Options

gpFillGridArea	Specifies whether or not the grid area is filled.
gpSetGridAreaFillColor	Sets the fill color for the grid area in a frame.
gpSetPolarAxesTitles	Sets the titles for the polar axes.
gpSetRAxisColor	Sets the color of the R-axis.
gpSetRAxisLabelColor	Sets the color of the R-axis tick labels.
gpSetRAxisLabelFont	Sets the type and size of the font used for the R-axis tick labels.
gpSetRAxisLabelOffset	Sets the offset of the R-axis tick labels from the axis line.

gpSetRAxisLabelSkip	Sets an option to show only a subset of the R-axis tick labels.
gpSetRAxisRange	Sets the range of the R-axis.
gpSetRAxisThickness	Sets the thickness of the R-axis line.
gpSetRAxisTicks	Sets the distance between major tick marks and the number of minor tick marks on the R-axis.
gpSetRAxisTitle	Sets the title for the R-axis.
gpSetRAxisTitleColor	Sets the color of the R-axis title.
gpSetRAxisTitleFont	Sets the type and size of the font used for the R-axis title.
gpSetRAxisTitleOffset	Sets the offset of the R-axis title from the axis line.
gpSetThetaAxisColor	Sets the color of the Theta-axis.
gpSetThetaAxisLabelColor	Sets the color of the Theta-axis tick labels.
gpSetThetaAxisLabelFont	Sets the type and size of the font used for the Theta-axis tick labels.
gpSetThetaAxisLabelOffset	Sets the offset of the Theta-axis tick labels from the axis line.
gpSetThetaAxisLabelSkip	Sets an option to show only a subset of the Theta-axis tick labels.
gpSetThetaAxisRange	Sets the range of the Theta-axis.
gpSetThetaAxisThickness	Sets the thickness of the Theta-axis line.
gpSetThetaAxisTicks	Sets the distance between major tick marks and the number of minor tick marks on the Theta-axis.
gpSetThetaAxisTitle	Sets the title for the Theta-axis.
gpSetThetaAxisTitleColor	Sets the color of the Theta-axis title.

gpSetThetaAxisTitleFont	Sets the type and size of the font used for the Theta-axis title.
gpSetThetaAxisTitleOffset	Sets the offset of the Theta-axis title from the axis line.
gpSetThetaMode	Specifies whether theta should be measured in degrees or radians.
gpSetThetaPeriod	Sets the theta range of a complete circle.
gpShowPolarAxes	Specifies which polar axes are to be displayed in a frame.

3D Perspective and Rotation

gpRotateAxes	Rotates a plot about the X, Y and Z axes.
gpRotateXAxis	Rotates a plot about the X-axis.
gpRotateYAxis	Rotates a plot about the Y-axis.
gpRotateZAxis	Rotates a plot about the Z-axis.
gpSet3DViewDistance	Sets the distance between the viewer and the center point of a plot.
gpSet3DViewerPosition	Sets the position of the viewer in 3D grid coordinates.
gpUse3DPerspective	Activates 3D perspective in a frame so that the shape of the plot is dependent on the specified field of view angle.

2.1.8 Sizing and Positioning the Graph

gpSetMagnification	Resizes a plot.
gpSetPlotFit	Controls how a plot is sized to the frame.
gpSetTranslation	Translates a plot horizontally and/or vertically.

2.1.9 Animation

gpAnimate3DRotation	Animates the rotation of a plot about the X,Y and Z axes.
gpAnimateFieldZones	Cycles through the specified field zones one at a time to produce animation.
gpAnimateLineMaps	Cycles through the specified line maps one at a time to produce animation.
gpAnimateXAxisRotation	Animates the rotation of a plot about the X-axis.
gpAnimateYAxisRotation	Animates the rotation of a plot about the Y-axis.
gpAnimateZAxisRotation	Animates the rotation of a plot about the Z-axis.

2.1.10 Setting Frame Options

gpMoveFrame	Moves a frame to a specified location.
gpSetFrameBackgroundColor	Sets the background color for a frame.
gpSetFrameHeaderColor	Sets the header color for a frame.
gpSetFrameHeaderText	Sets the text to be displayed in all frame headers.
gpSetFrameName	Sets the name for a frame.
gpSetFrameStyle	Sets style options for a frame.
gpSetTileMargins	Sets the margins between the frames and the edges of the paper for the **gpTile** command.
gpTile	Tiles frames in the workspace.

2.1.11 Adding Text

gpAddTextObject	Adds a text object to a frame.
gpAddTextObject3D	Adds a text object to a frame at a specified position in 3D grid coordinates.
gpRotateTextObject	Rotates a text object from the horizontal.
gpSetDefaultTextFont	Sets the default font to be used for text objects.
gpSetTextObjectAnchorLocation	Sets the anchor location of a text object.
gpSetTextObjectBoxColor	Sets the colors of the outline and background of the box surrounding a text object.
gpSetTextObjectBoxStyle	Sets the style of the box surrounding a text object.
gpSetTextObjectBoxThickness	Sets the thickness of the outline of the box surrounding a text object.
gpSetTextObjectBoxType	Sets the type of the box surrounding a text object.
gpSetTextObjectColor	Sets the color of the text in a text object.
gpSetTextObjectFont	the type and size of the font used in a text object.
gpSetTextObjectLineSpacing	Sets the spacing between the lines of text in a text object.
gpSetTextObjectMargin	Sets the size of the margin between the text in a text object and the surrounding box.

2.1.12 Adding Arrows

gpAddArrow	Adds an arrow to a frame.
gpSetArrowColor	Sets the color of an arrow.
gpSetArrowHeadAngle	Sets the angle that one side of an arrow head makes with the arrow line segment.
gpSetArrowHeadAttachment	Specifies to which end of an arrow the arrow head is attached.
gpSetArrowHeadSize	Sets the size of an arrow.
gpSetArrowHeadStyle	Sets the style of the head of an arrow.
gpSetArrowHeadType	Sets the type of arrow head to use for an arrow.
gpSetArrowLinePattern	Sets the pattern of the arrow line segment.
gpSetArrowLinePatternLength	Sets the length of the arrow line segment pattern.
gpSetArrowLineThickness	Sets the line thickness of the arrow line segment.

2.1.13 Adding Line Segments

gpAddLineSegment	Adds a line segment to a frame.
gpAddLineSegment3D	Adds a line segment to a frame at a specified position in 3D grid coordinates.
gpSetLineSegmentColor	Sets the color of a line segment.
gpSetLineSegmentPattern	Sets the pattern of the line segment.
gpSetLineSegmentPatternLength	Sets the length of the line segment pattern.
gpSetLineSegmentThickness	Sets the thickness of the line segment.

2.1.14 Adding Other Geometries

gpAddCircle	Adds a circular geometry to a frame.
gpAddEllipse	Adds an elliptical geometry to a frame.
gpAddRectangle	Adds a rectangular geometry to a frame.
gpAddSquare	Adds a square geometry to a frame.
gpFillGeometry	Specifies whether or not a geometry object is filled.
gpSetGeometryColor	Sets the color of a geometry object.
gpSetGeometryFillColor	Sets the fill color of a geometry object.
gpSetGeometryLinePattern	Sets the line pattern of the geometry object.
gpSetGeometryLinePatternLength	Sets the length of the geometry object line pattern.
gpSetGeometryLineThickness	Sets the line thickness of the geometry object.
gpSetGeometryNumSides	Sets the number of sides approximating a circular or elliptical geometry object.

2.1.15 Printing and Exporting

gpExportBMP	Exports a graph to a Windows Bitmap (BMP) format graphics file.
gpExportEPS	Exports a graph to an Encapsulated PostScript (EPS) format graphics file.
gpExportHPGL	Exports a graph to an HP-GL format graphics file.
gpExportHPGL2	Exports a graph to an HP-GL/2 format graphics file.

gpExportJPEG	Exports a graph to a Joint Photographic Experts Group (JPEG) format graphics file.
gpExportPNG	Exports a graph to a Portable Network Graphics (PNG) file.
gpExportPS	Exports a graph to a PostScript (PS) format graphics file.
gpExportRAS	Exports a graph to a Sun Raster (RAS) format graphics file.
gpExportTIFF	Exports a graph to a Tagged Image File Format (TIFF) graphics file.
gpExportWMF	Exports a graph to a Windows Meta File (WMF) format graphics file.
gpExportXWD	Exports a graph to an X-Windows Format (XWD) graphics file.
gpPrint	Prints a graph.
gpPrintToFile	Specifies a print output file name or deactivates printing to a file.
gpSetExportImageWidth	Specifies the width of the exported image in pixels.
gpSetExportJPEGQuality	Sets the quality of the output JPEG graphics file.
gpSetPaperOrientation	Sets the orientation of the page on which the graph is displayed.
gpSetPaperSize	Sets the size of the page on which the graph is displayed.
gpSetPrintFormat	Sets the print output format for printing on UNIX or Linux.
gpSetPrintPalette	Sets the print/export palette to monochrome or color.

2.1.16 Plotting the Graph

gpMake2DCartesianPlot	Writes a data file and draws a graph with one frame containing a 2D Cartesian plot.
gpMake2DContourPlot	Writes a data file and draws a graph with one frame containing a 2D Contour plot.
gpMake3DCartesianPlot	Writes a data file and draws a graph with one frame containing a 3D Cartesian plot.
gpMake3DContourPlot	Writes a data file and draws a graph with one frame containing a 3D Contour plot.
gpMakeBarPlot	Writes a data file and draws a graph with one frame containing a Bar plot.
gpMakeBoxPlot	Writes a data file and draws a graph with one frame containing a Box plot (box graph percentile plot).
gpMakeHistPlot	Writes a data file and draws a graph with one frame containing a Hist plot (frequency histogram).
gpMakeHistFPlot	Writes a data file and draws a graph with one frame containing a HistF plot (frequency histogram), given a vector of frequency counts.
gpMakeHistPPlot	Writes a data file and draws a graph with one frame containing a HistP plot (percentage histogram).
gpMakeLogLogPlot	Writes a data file and draws a graph with one frame containing a LogLog plot.
gpMakeLogXPlot	Writes a data file and draws a graph with one frame containing a LogX plot.
gpMakeLogYPlot	Writes a data file and draws a graph with one frame containing a LogY plot.
gpMakePolarLinePlot	Writes a data file and draws a graph with one frame containing a Polar Line plot.

gpMakeXYLinePlot	Writes a data file and draws a graph with one frame containing an XY Line plot.
gpPlot	Draws a graph.

2.1.17 Miscellaneous Commands

gpAddMacroFile	Adds one or more macro files to a plot control structure, to be run at the end of the macro file created with the structure data.
gpCleanUp	Deletes the data, macro and structure files associated with a graph.
gpFramePause	Pauses drawing for a specified number of seconds after drawing for a frame is completed.
gpGetErrorMessage	Returns the error message associated with a particular error code.
gpOpenLayoutFile	Opens a layout file in **GAUSSplot**.
gpRedefineColor	Sets the RGB values for a color in the color palette.
gpRunMacroFile	Runs a macro file in **GAUSSplot**.
gpSetErrorLevel	Sets the level of error handling in your program.
gpSetPaperOrientation	Sets the orientation of the page on which the graph is displayed.
gpSetPaperSize	Sets the size of the page on which the graph is displayed.
gpSetWorkspaceView	Sets workspace view options.
gpShowDrawing	Specifies whether or not drawing is shown.
gpWait	Specifies whether **GAUSSplot** will return to **GAUSS** immediately or wait until the **GAUSSplot** window has been closed.

2.2 Global Variables

GAUSSplot uses global variables to contain all of the available options for certain command input arguments. These options may be referenced by name or by index into the global variable in which they are contained.

2.2.1 GAUSSplot Colors

The **GAUSSplot** color palette contains 68 colors. The color names are stored in the 68×1 global vector **`_gp_colors`**, which is defined as follows:

Index	Color	Index	Color
1	Black	35	Custom27
2	White	36	Custom28
3	Red	37	Custom29
4	Yellow	38	Custom30
5	Green	39	Custom31
6	Cyan	40	Custom32
7	Blue	41	Custom33
8	Purple	42	Custom34
9	Custom1	43	Custom35
10	Custom2	44	Custom36
11	Custom3	45	Custom37
12	Custom4	46	Custom38
13	Custom5	47	Custom39
14	Custom6	48	Custom40
15	Custom7	49	Custom41
16	Custom8	50	Custom42
17	Custom9	51	Custom43
18	Custom10	52	Custom44
19	Custom11	53	Custom45
20	Custom12	54	Custom46
21	Custom13	55	Custom47
22	Custom14	56	Custom48
23	Custom15	57	Custom49

Index	Color	Index	Color
24	Custom16	58	Custom50
25	Custom17	59	Custom51
26	Custom18	60	Custom52
27	Custom19	61	Custom53
28	Custom20	62	Custom54
29	Custom21	63	Custom55
30	Custom22	64	Custom56
31	Custom23	65	Multi
32	Custom24	66	Multi2
33	Custom25	67	Multi3
34	Custom26	68	Multi4

Any of the solid colors (**_gp_colors**[1:64]) may be redefined using user-specified RGB values by calling **gpRedefineColor**.

Colors 'Multi', 'Multi2', 'Multi3', and 'Multi4' (**_gp_colors**[65:68]) may be used only for mesh, contour lines, vectors, and scatter symbols. They use the contour variable for the corresponding contour group (1 - 4) to determine the color displayed at each point. The default contour variable for all contour groups is the last variable in the data file. To specify a different contour variable, call **gpSetContourVariable**.

2.2.2 GAUSSplot Fonts

GAUSSplot includes nine fonts, which can be used for any text included in a graph. The font names are stored in the 11×1 global vector **_gp_fonts**, which is defined as follows:

Index	Font
1	Courier
2	CourierBold
3	Helv (helvetica)
4	HelvBold
5	Times
6	TimesBold
7	TimesItalic
8	TimesItalicBold
9	Greek
10	Math
11	UserDef

See Appendix A for a table that lists all of the available characters in **GAUSSplot**.

2.2.3 GAUSSplot Line Patterns

In **GAUSSplot**, there are six different kinds of line patterns which can be used for any lines, mesh, arrows, etc. in a graph. The line pattern names are stored in the 6×1 global vector **_gp_line_patterns**, which is defined as follows:

Index	Line Pattern
1	Solid
2	Dashed
3	DashDot
4	Dotted
5	LongDash
6	DashDotDot

2.2.4 GAUSSplot Symbols

GAUSSplot includes eleven preset symbol shapes to represent data points in a graph when the either the symbol or scatter layer is turned on (see **gpShowSymbolLayer** and

gpShowScatterLayer). The symbol names are stored in the 11×1 global vector **_gp_symbols**, which is defined as follows:

Index	Symbol Shape
1	Square
2	Del (delta)
3	Grad (gradient)
4	RTri (right triangle)
5	LTri (left triangle)
6	Diamond
7	Circle
8	Point
9	Cube
10	Sphere
11	Octahedron

Only **_gp_symbols**[1:7] may be used in Bar, XY Line and Polar Line plots.

2.2.5 GAUSSplot Error Messages

All of the error messages used by **GAUSSplot** are stored in the global variable **_gp_error_messages**, which is defined as follows:

Index	Error Message
1	Frame index out of range
2	Line Map index out of range
3	Cannot access variable names in structure file - Use variable indices
4	Invalid plot type
5	Invalid variable names
6	Invalid argument
7	Plot type not set

Index	Error Message
8	Could not delete all associated files
9	Arguments not conformable
10	Cannot open file for write
11	Cannot write to file: out of disk space
12	Number of variable names does not match number of variables
13	Contour Group index out of range
14	Index out of range
15	Invalid object name
16	Data array must have 3 or 4 dimensions
17	No data file specified
18	Variables do not match
19	Invalid curve fit type
20	Variables must be 2 or 3-dimensional
21	Cannot open structure file
22	Frame does not contain a field zone legend
23	Field zone legend does not contain a title
24	Argument out of range
25	Wrong object type
26	Too many frames, use gpPlot to plot the graph

2.2.6 GAUSSplot Debugging

GAUSSplot prints debugging information to `stderr` if **`_gp_debug`** is set to a scalar 1 in your program. By default, it is set to 0.

Command Reference 3

gp2DCartesianPlotCreate

PURPOSE Creates a **gpPlotControl** structure and initializes its members to default values for a 2D Cartesian plot.

LIBRARY gaussplot

FORMAT *gp* = **gp2DCartesianPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a 2D Cartesian plot.

REMARKS You may use *gp* to plot a graph using either **gpMake2DCartesianPlot** or **gpPlot**. Calling **gp2DCartesianPlotCreate** sets the plot type of the entire graph to 2D Cartesian. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gp2DCartesianPlotCreate** contains one frame by default.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
x = seqa(1,5,20);
y = seqa(10,2,20).*log(x');
x = x'.*ones(20,20);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gp2DCartesianPlotCreate;
ret = gpMake2DCartesianPlot(&gp,x,y,vnames);
```

This example illustrates how you may create a 2D Cartesian plot with one field zone, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, and the second variable (**V2**) with the Y-axis.

SOURCE gpcartesianplot.src

SEE ALSO **gpMake2DCartesianPlot**, **gpPlot**

gp2DContourPlotCreate

PURPOSE Creates a **gpPlotControl** structure and initializes its members to default values for a 2D Contour plot.

LIBRARY gaussplot

FORMAT *gp* **= gp2DContourPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a 2D

Contour plot.

REMARKS

You may use *gp* to plot a graph using either **gpMake2DContourPlot** or **gpPlot**. Calling **gp2DContourPlotCreate** sets the plot type of the entire graph to 2D Contour. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gp2DContourPlotCreate** contains one frame by default.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
p = pi;
x = seqa(0,p/4,15)';
y = seqa(0,p/4,15);

cv = cos(x).*sin(x-y);
x = x.*ones(15,15);
y = y.*ones(15,15);
string vnames = { "V1", "V2", "V3" };

struct gpPlotControl gp;
gp = gp2DContourPlotCreate;
ret = gpMake2DContourPlot(&gp,x,y,cv,vnames);
```

This example illustrates how you may create a 2D Contour plot with one field zone, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, the second variable (**V2**) with the Y-axis, and the third variable (**V3**) will be the contour variable.

SOURCE

gpcontourplot.src

SEE ALSO

gpMake2DContourPlot, **gpPlot**

gp3DCartesianPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a 3D Cartesian plot.

LIBRARY gaussplot

FORMAT *gp* = **gp3DCartesianPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a 3D Cartesian plot.

REMARKS You may use *gp* to plot a graph using either **gpMake3DCartesianPlot** or **gpPlot**. Calling **gp3DCartesianPlotCreate** sets the plot type of the entire graph to 3D Cartesian. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gp3DCartesianPlotCreate** contains one frame by default.

EXAMPLE
```
library gaussplot;
#include gp.sdf

// Generate data
p = pi;
x = seqa(0,p/4,15)';
y = seqa(0,p/4,15);

z = cos(x).*sin(x-y);
x = x.*ones(15,15);
y = y.*ones(15,15);
string vnames = { "V1", "V2", "V3" };

struct gpPlotControl gp;
```

```
gp = gp3DCartesianPlotCreate;
ret = gpMake3DCartesianPlot(&gp,x,y,z,vnames);
```

This example illustrates how you may create a 3D Cartesian plot with one field zone, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, the second variable (**V2**) with the Y-axis, and the third variable (**V3**) with the Z-axis.

SOURCE gpcartesianplot.src

SEE ALSO **gpMake3DCartesianPlot**, **gpPlot**

gp3DContourPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a 3D Contour plot.

LIBRARY gaussplot

FORMAT *gp* = **gp3DContourPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a 3D Contour plot.

REMARKS You may use *gp* to plot a graph using either **gpMake3DContourPlot** or **gpPlot**. Calling **gp3DContourPlotCreate** sets the plot type of the entire graph to 3D Contour. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gp3DContourPlotCreate** contains one frame by default.

EXAMPLE

```
library gaussplot;
```

```
#include gp.sdf

// Generate data
p = pi;
x = seqa(0,p/4,15)';
y = seqa(0,p/4,15);

z = cos(x).*sin(x-y);
cv = sin(x).*cos(x-y);
x = x.*ones(15,15);
y = y.*ones(15,15);
string vnames = { "V1", "V2", "V3", "V4" };

struct gpPlotControl gp;
gp = gp3DContourPlotCreate;
ret = gpMake3DContourPlot(&gp,x,y,z,cv,vnames);
```

This example illustrates how you may create a 3D Contour plot with one field zone, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, the second variable (**V2**) with the Y-axis, the third variable (**V3**) with the Z-axis, and the fourth variable (**V4**) will be the contour variable.

SOURCE gpcontourplot.src

SEE ALSO **gpMake3DContourPlot**, **gpPlot**

gpAdd2DCartesianFrame

PURPOSE Adds a 2D Cartesian plot frame to a **gpPlotControl** structure creted with **gpPlotCreate** or **gp2DCartesianPlotCreate**.

LIBRARY gaussplot

FORMAT **gpAdd2DCartesianFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS By default, **gpAdd2DCartesianFrame** creates a frame with one field zone and associates the first variable in the data file with the X-axis and the second variable with the Y-axis. Use **gpSetAxisMap** to use different variables for the axes.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE
```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAdd2DCartesianFrame(&gp);
gpAdd3DContourFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The preceding example creates a graph with two frames. The first contains a 2D Cartesian plot, and the second a 3D Contour plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least three variables.

SOURCE `gpcartesianplot.src`

SEE ALSO **gpPlotCreate**, **gp2DCartesianPlotCreate**, **gpPlot**

gpAdd2DContourFrame

PURPOSE Adds a 2D Contour plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gp2DContourPlotCreate**.

LIBRARY `gaussplot`

FORMAT **gpAdd2DContourFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS By default, **gpAdd2DContourFrame** creates a frame with one field zone and associates the first variable in the data file with the X-axis, the second variable with the Y-axis, and uses the last variable as the contour variable. Use **gpSetAxisMap** to use different variables for the axes, and **gpSetContourVariable** to change the contour variable.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE
```
library gaussplot;
#include gp.sdf
```

```
struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAdd2DContourFrame(&gp);
gpAdd3DContourFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a 2D Contour plot, and the second a 3D Contour plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least three variables.

SOURCE `gpcontourplot.src`

SEE ALSO **gpPlotCreate**, **gp2DContourPlotCreate**, **gpPlot**

gpAdd3DCartesianFrame

gpAdd3DCartesianFrame

PURPOSE Adds a 3D Cartesian plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gp3DCartesianPlotCreate**.

LIBRARY gaussplot

FORMAT **gpAdd3DCartesianFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS By default, **gpAdd3DCartesianFrame** creates a frame with one field zone and associates the first variable in the data file with the X-axis, the second variable with the Y-axis, and the third variable with the Z-axis. Use **gpSetAxisMap** to use different variables for the axes.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE
```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAdd2DContourFrame(&gp);
gpAdd3DCartesianFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
```

```
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a 2D Contour plot, and the second a 3D Cartesian plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least three variables.

SOURCE `gpcartesianplot.src`

SEE ALSO **gpPlotCreate**, **gp3DCartesianPlotCreate**, **gpPlot**

gpAdd3DContourFrame

PURPOSE Adds a 3D Contour plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gp3DContourPlotCreate**.

LIBRARY `gaussplot`

FORMAT **gpAdd3DContourFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS By default, **gpAdd3DContourFrame** creates a frame with one field zone and associates the first variable in the data file with the X-axis, the second variable with the Y-axis, the third variable with the Z-axis, and uses the last variable as the contour variable. Use **gpSetAxisMap** to use different variables for the axes and **gpSetContourVariable** to change the contour variable.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAdd2DContourFrame(&gp);
gpAdd3DContourFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a 2D Contour plot, and the second a 3D Contour plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least three variables.

SOURCE `gpcontourplot.src`

SEE ALSO **gpPlotCreate**, **gp3DContourPlotCreate**, **gpPlot**

gpAddArrow

PURPOSE Adds an arrow to a specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpAddArrow**(*pgp*, *framenum*, *name*, *xstart*, *ystart*, *xend*, *yend*, *coord*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name for the arrow object.
xstart	scalar, horizontal position at which the arrow begins.
ystart	scalar, vertical position at which the arrow begins.
xend	scalar, horizontal position at which the arrow ends.
yend	scalar, vertical position at which the arrow ends.
coord	scalar, position coordinate system, one of the following: **1** starting and ending position in frame coordinates, 0-100, where (0, 0) is lower left corner of frame. **2** starting and ending position in 2D grid coordinates, i.e., X and Y plot coordinates.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Specifying a name for the arrow object allows you to reference the object in other commands that set arrow options. The arrow will be drawn from the point (*xstart*, *ystart*) to the point (*xend*, *yend*) in the coordinates specified with *coord*.

EXAMPLE

```
framenum = 2;
xstart = 4;
xend = 6.5;
```

```
ystart = 6;
yend = 6.75;
coord = 2;
ret = gpAddArrow(&gp,framenum,"arrow1",xstart,ystart,xend,
                 yend,coord);
ret = gpSetArrowColor(&gp,framenum,"arrow1","blue");
```

This example adds an arrow to frame two, drawing the arrow from the point (4, 6) to the point (6.5, 6.75) in grid coordinates. It then sets the color of the arrow to blue.

SOURCE gparrow.src

SEE ALSO **gpSetArrowColor**, **gpSetArrowLineThickness**, **gpSetArrowHeadStyle**

gpAddBarFrame

PURPOSE Adds a Bar plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpBarPlotCreate**.

LIBRARY gaussplot

FORMAT **gpAddBarFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS By default, **gpAddBarFrame** creates a frame with one line map, associating the first variable in the data file with the X-axis, and the second variable with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAddBarFrame(&gp);
gpAddXYLineFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a Bar plot, and the second an XY Line plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least two variables.

SOURCE `gpbarplot.src`

SEE ALSO **gpPlotCreate**, **gpBarPlotCreate**, **gpPlot**

gpAddBoxFrame

PURPOSE Adds an Box plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpBoxPlotCreate**.

LIBRARY `gaussplot`

FORMAT **gpAddBoxFrame**(*pgp*, *grp*, *y*);

INPUT

pgp	pointer to a plot control structure.
grp	1×P vector or N×P matrix, containing the group numbers corresponding to each column of *y* data.
y	N×P matrix. Each column represents the set of *y* values for an individual percentile box symbol.

REMARKS If *grp* is N×P, each row of *grp* should be identical.

gpAddBoxFrame computes percentiles on the data in *y* and adds box symbols to represent those percentiles to the new frame. However, you will still need to write *grp* (or **reshape**(*grp*,N,P) if *grp* is 1×P) and *y* to a `.plt` data file before plotting your graph.

By default, **gpAddBoxFrame** creates a frame with P line maps, associating the first P variables in the data file with the X-axis, and the next P variables with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify different line maps.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE
```
library gaussplot;
#include gp.sdf
```

```
// Initialize plot control structure
struct gpPlotControl gp;
gp = gpBoxPlotCreate;
gpSetErrorLevel(&gp,2);

// Create data
rndseed 12345;
nboxes = 5;
npoints = 7;
x=reshape(seqa(1,1,nboxes)',npoints,nboxes);
y=rndn(npoints,nboxes);

// Write data file
struct gpData gdat;
gdat = gpSetPlotData(x~y,"");
call gpWritePlotData(&gdat,"gpbox.plt");

// Set data and macro files
gpSetMacroFile(&gp,"gpbox.mcr");
call gpSetDataFile(&gp,0,"gpbox.plt");

// Set up frame 1
call gpAddBoxFrame(&gp,x,y);
call gpMoveFrame(&gp,1,1,0.25,9,3.925);

// Set up frame 2
call gpSetBoxMinMaxMethod(&gp,2,2);
call gpAddBoxFrame(&gp,x,y);
call gpShowSymbolLayer(&gp,2,1);
call gpSetSymbolSize(&gp,2,seqa(1,1,nboxes),1);
call gpMoveFrame(&gp,2,1,4.325,9,3.925);

// Plot the graph
call gpPlot(&gp);
```

The above example creates a graph with two frames, both containing Box plots. The second frame displays symbols at each data point as well as the percentile

box symbols.

SOURCE gpxylineplot.src

SEE ALSO **gpPlotCreate**, **gpBoxPlotCreate**, **gpPlot**

gpAddCircle

PURPOSE Adds a circular geometry object to the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpAddCircle**(*pgp*, *framenum*, *name*, *xstart*, *ystart*, *xradius*, *coord*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name for the line segment object.
xstart	scalar, horizontal position of the center of the circle.
ystart	scalar, vertical position of the center of the circle.
xradius	scalar, length of the radius of the circle in X plot coordinates or horizontal frame coordinates.
coord	scalar, position coordinate system, one of the following: **1** starting position, width, and height in frame coordinates, 0-100, where (0, 0) is lower left corner of frame. **2** starting position, width and height in 2D grid coordinates, i.e., X and Y plot coordinates.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Specifying a name for the circular geometry object allows you to reference the object in other commands that set options for the object. The following

commands may be used to set options for circles and other types of geometries that have been added to your graph:

gpSetGeometryColor
gpSetGeometryLineThickness
gpSetGeometryLinePattern
gpSetGeometryLinePatternLength
gpFillGeometry
gpSetGeometryFillColor
gpSetGeometryNumSides

By default, a circular geometry object is estimated by 72 sides. You may call **gpSetGeometryNumSides** after adding a circle to change the number of sides that the circle is estimated by.

You may also use **gpSetGeometryNumSides** in conjunction with **gpAddCircle** to create a regular polygon with N sides.

EXAMPLE

```
framenum = 1;
xstart = 3;
ystart = 0;
xradius = 2;
coord = 2;
ret = gpAddCircle(&gp,framenum,"circ1",xstart,ystart,xradius,
                  coord);
ret = gpSetGeometryColor(&gp,framenum,"circ1","purple");
```

This example adds a circle with radius of 2 to frame one, centered on the point (3,0). It then sets the color of the circle to purple.

```
framenum = 2;
xstart = 1;
ystart = 4;
xradius = 1;
coord = 2;
```

```
numsides = 6;
ret = gpAddCircle(&gp,framenum,"hex1",xstart,ystart,xradius,
                  coord);
ret = gpSetGeometryNumSides(&gp,framenum,"hex1",numsides);
```

This example adds a hexagon with radius of 1 to frame two, centered on the point (1,4).

SOURCE gpgeometry.src

SEE ALSO **gpAddEllipse**, **gpSetGeometryColor**, **gpSetGeometryLineThickness**

gpAddContourLegend

PURPOSE Adds a contour legend for the specified contour group(s).

LIBRARY gaussplot

FORMAT *ret* = **gpAddContourLegend**(*pgp*, *framenum*, *contourgroupnum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar or N×1 vector, contour group number(s).

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS All contour options are set within a contour group. Each field zone that is displayed as a contour plot must be associated with a particular contour group, which controls the contour options.

Contour plots can be displayed with contour lines and/or contour flooding. If both lines and flooding are used in a field zone, they may be associated with

different contour groups. By default a contour plot is displayed with contour flooding only, using contour group one to control all options.

EXAMPLE

```
framenum = 1;
contourgroupnum = 1;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
```

SOURCE gpcontourplot.src

SEE ALSO **gpMoveContourLegend**, **gpSetContourLegendBoxStyle**, **gpSetContourLegendOrientation**

gpAddEllipse

PURPOSE Adds an elliptical geometry object to the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpAddEllipse**(*pgp*, *framenum*, *name*, *xstart*, *ystart*, *xhaxis*, *yvaxis*, *coord*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name for the line segment object.
xstart	scalar, horizontal position of the center of the ellipse.
ystart	scalar, vertical position of the center of the ellipse.
xhaxis	scalar, length of the horizontal axis of the ellipse in X plot coordinates or horizontal frame coordinates.
yvaxis	scalar, length of the vertical axis of the ellipse in X plot coordinates or horizontal frame coordinates.

coord scalar, position coordinate system, one of the following:

1 starting position, width, and height in frame coordinates, 0-100, where (0, 0) is lower left corner of frame.

2 starting position, width and height in 2D grid coordinates, i.e., X and Y plot coordinates.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Specifying a name for the elliptical geometry object allows you to reference the object in other commands that set options for the object. The following commands may be used to set options for ellipses and other types of geometries that have been added to your graph:

gpSetGeometryColor

gpSetGeometryLineThickness

gpSetGeometryLinePattern

gpSetGeometryLinePatternLength

gpFillGeometry

gpSetGeometryFillColor

gpSetGeometryNumSides

By default, an elliptical geometry object is estimated by 72 sides. You may call **gpSetGeometryNumSides** after adding an ellipse to change the number of sides that the ellipse is estimated by.

You may also use **gpSetGeometryNumSides** in conjunction with **gpAddEllipse** to create a non-regular polygon with N sides.

EXAMPLE
```
framenum = 1;
xstart = 1;
ystart = 2;
xhaxis = 2.25;
yvaxis = 1.5;
coord = 2;
```

```
ret = gpAddEllipse(&gp,framenum,"ell1",xstart,ystart,
                        xhaxis,yvaxis,coord);
ret = gpSetGeometryColor(&gp,framenum,"ell1","blue");
```

This example adds an ellipse with a horizontal axis of 2.25 and a vertical axis of 1.5 to frame one, centered on the point (1,2). It then sets the color of the ellipse to blue.

```
framenum = 2;
xstart = 2;
ystart = 3;
xhaxis = 1;
yvaxis = 1.5;
coord = 2;
numsides = 8;
ret = gpAddEllipse(&gp,framenum,"oct1",xstart,ystart,
                        xhaxis,yvaxis,coord);
ret = gpSetGeometryNumSides(&gp,framenum,"oct1",numsides);
```

This example adds a non-regular octogon with a horizontal axis of 1 and a vertical axis of 1.5 to frame two, centered on the point (2,3).

SOURCE gpgeometry.src

SEE ALSO **gpAddCircle**, **gpSetGeometryColor**, **gpSetGeometryLineThickness**

gpAddFieldZoneLegendFrame

PURPOSE Adds a field zone legend frame as an inset in the specified frame.

LIBRARY gaussplot

gpAddFieldZoneLegendFrame

FORMAT *ret* = **gpAddFieldZoneLegendFrame**(*pgp*, *framenum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A field zone legend identifies each field zone in the specified frame. If the contour layer is the only layer activated in a field zone, then that field zone is not included in the legend. Call **gpAddContourLegend** to add a contour legend to the frame.

You must use the **GAUSSplot** API to add and set options for field zone legends. They are not available from the GUI.

A field zone legend is contained in its own frame, which is positioned as an inset in the parent frame (the parent frame is the frame specified by *framenum*). When the graph is plotted in the **GAUSSplot** GUI, the legend may be the active frame. If this is the case, then you will need to select the parent frame with the mouse (or select 'Push Current Frame Back' from the 'Frame' menu) before you may make changes to the parent frame. When you select the parent frame, the legend will be pushed back behind it. Select 'Push Current Frame Back' from the 'Frame' menu with the parent frame selected to make the legend reappear.

Because information about the data file is needed to create the field zone legend, you must keep the associated structure file together with your data file if you wish to use field zone legends. **gpAddFieldZoneLegendFrame** accesses the structure file to obtain the needed information about the data file. See Section 1.8 for more information on structure files.

The field zone names are used to identify the field zones in the legend. Field zone names come from the names of the corresponding zones in the data file. If you write the data file yourself, you may specify zone names with **gpAddZone**. Otherwise, default names ("Zone 1", "Zone 2",. . ., "Zone N") are used. Call **gpSetFieldZoneNames** to rename the field zones in the frame before adding a legend.

If one or more of the field zones in the specified frame are represented by 3D scatter symbols (i.e., the scatter layer is activated and the scatter shape is set to 'cube', 'sphere' or 'octahedron'), then 2D representations of the scatter symbols will be used in the field zone legend.

This command applies only to 2D and 3D Cartesian plots. Graphs containing field zone legends must be plotted with **gpPlot**. **gpMake3DCartesianPlot** does not support field zone legends.

EXAMPLE
```
framenum = 1;
ret = gpAddFieldZoneLegendFrame(&gp,framenum);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetFieldZoneNames**, **gpMoveFieldZoneLegend**, **gpSetFieldZoneLegendBoxStyle**, **gpSetFieldZoneLegendLineSpacing**

gpAddHistFrame

PURPOSE Adds a Hist plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpHistPlotCreate**.

LIBRARY gaussplot

FORMAT **gpAddHistFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS A Hist plot is a frequency histogram.

By default, **gpAddHistFrame** creates a frame with one line map, associating the first variable in the data file with the X-axis, and the second variable with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Call **gpCreateHistData** to generate from your data the vectors of category midpoints and frequency counts that are needed to plot the graph, and write the second and third returns from **gpCreateHistData** to your **GAUSSplot** data file.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Initialize plot control structure
struct gpPlotControl gp;
gp = gpPlotCreate;

// Add frames
gpAddHistFrame(&gp);
gpAddXYLineFrame(&gp);

// Generate data and write data file for first frame
struct gpData gdat;
x = round(rndn(4000,1)*100);
{ b,m,freq } = gpCreateHistData(x,50);
string vnames = { "Cat", "Freq" };
gdat = gpSetPlotData(m~freq,vnames);
ret = gpWritePlotData(&gdat,"mydata_hist.plt");

// Set data files for both frames
ret = gpSetDataFile(&gp,1,"mydata_hist.plt");
ret = gpSetDataFile(&gp,2,"mydata_xyline.plt");

// Position frame 1
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
```

```
ret = gpMoveFrame(&gp,1,xpos,ypos,width,height);

// Position frame 2
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,2,xpos,ypos,width,height);

// Plot the graph
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a Hist plot, and the second an XY Line plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata_xyline.plt` already exists and contains at least two variables.

SOURCE `gphistplot.src`

SEE ALSO **gpPlotCreate**, **gpHistPlotCreate**, **gpPlot**

gpAddHistFFrame

PURPOSE Adds a HistF plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpHistFPlotCreate**.

LIBRARY `gaussplot`

FORMAT **gpAddHistFFrame**(*pgp*);

INPUT *pgp* pointer to a plot control structure.

REMARKS A HistF plot is a frequency histogram generated from a vector of frequency

counts.

By default, **gpAddHistFFrame** creates a frame with one line map, associating the first variable in the data file with the X-axis, and the second variable with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Initialize plot control structure
struct gpPlotControl gp;
gp = gpPlotCreate;

// Add frames
gpAddHistFFrame(&gp);
gpAddXYLineFrame(&gp);

// Generate data and write data file for first frame
struct gpData gdat;
f = { 10, 5, 0, 35, 11, 18, 55, 20 };
c = { 10, 20, 30, 40, 50, 60, 70, 80 };
string vnames = { "Cat", "Freq" };
gdat = gpSetPlotData(c~f,vnames);
ret = gpWritePlotData(&gdat,"mydata_histf.plt");

// Set data files for both frames
ret = gpSetDataFile(&gp,1,"mydata_histf.plt");
ret = gpSetDataFile(&gp,2,"mydata_xyline.plt");

// Position frame 1
xpos = 0.25;
ypos = 0.25;
```

```
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,1,xpos,ypos,width,height);

// Position frame 2
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,2,xpos,ypos,width,height);

// Plot the graph
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a HistF plot, and the second an XY Line plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata_xyline.plt` already exists and contains at least two variables.

SOURCE `gphistplot.src`

SEE ALSO **gpPlotCreate**, **gpHistFPlotCreate**, **gpPlot**

gpAddHistPFrame

PURPOSE Adds a HistP plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpHistPPlotCreate**.

LIBRARY `gaussplot`

FORMAT **gpAddHistPFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS

A HistP plot is a percentage histogram.

By default, **gpAddHistPFrame** creates a frame with one line map, associating the first variable in the data file with the X-axis, and the second variable with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Call **gpCreateHistPData** to generate from your data the vectors of category midpoints and frequency percentages that are needed to plot the graph, and write the second and fourth returns from **gpCreateHistPData** to your **GAUSSplot** data file.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Initialize plot control structure
struct gpPlotControl gp;
gp = gpPlotCreate;

// Add frames
gpAddHistPFrame(&gp);
gpAddXYLineFrame(&gp);

// Generate data and write data file for first frame
struct gpData gdat;
x = round(rndn(4000,1)*100);
{ b,m,freqct,freqp } = gpCreateHistPData(x,50);
string vnames = { "Cat", "Prcnt" };
gdat = gpSetPlotData(m~freqp,vnames);
ret = gpWritePlotData(&gdat,"mydata_histp.plt");

// Set data files for both frames
ret = gpSetDataFile(&gp,1,"mydata_histp.plt");
ret = gpSetDataFile(&gp,2,"mydata_xyline.plt");
```

```
// Position frame 1
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,1,xpos,ypos,width,height);

// Position frame 2
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,2,xpos,ypos,width,height);

// Plot the graph
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a HistP plot, and the second an XY Line plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata_xyline.plt` already exists and contains at least two variables.

SOURCE `gphistplot.src`

SEE ALSO **gpPlotCreate**, **gpHistPPlotCreate**, **gpPlot**

gpAddLineLegend

PURPOSE Adds a line legend to a frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpAddLineLegend**(*pgp*, *framenum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A line legend identifies each line map in the frame using the names of the line maps. By default, line maps are given either the same name as the variable associated with the Y (or R) axis in the line map or "V*num*", where *num* is the index of the variable in the data file, if the variable names cannot be accessed from the structure file when the line maps are created. See Section 1.8 for information about structure files. Use **gpSetLineMapNames** to rename the line maps in a frame before adding a line legend.

This command applies only to Bar, XY Line and Polar Line plots.

EXAMPLE
```
framenum = 1;
ret = gpAddLineLegend(&gp,framenum);
```

SOURCE gplinemap.src

SEE ALSO **gpMoveLineLegend**, **gpSetLineLegendBoxStyle**, **gpSetLineLegendMargin**, **gpSetLineLegendLineSpacing**

gpAddLineSegment

PURPOSE Adds a line segment to the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpAddLineSegment**(*pgp*, *framenum*, *name*, *xstart*, *ystart*, *xend*, *yend*, *coord*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name for the line segment object.
xstart	scalar, horizontal position at which the line segment begins.
ystart	scalar, vertical position at which the line segment begins.
xend	scalar, horizontal position at which the line segment ends.
yend	scalar, vertical position at which the line segment ends.
coord	scalar, position coordinate system, one of the following: **1** starting and ending position in frame coordinates, 0-100, where (0, 0) is lower left corner of frame. **2** starting and ending position in 2D grid coordinates, i.e., X and Y plot coordinates.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

Specifying a name for the line segment object allows you to reference the object in other commands that set line segment options. The line segment will be drawn from the point (*xstart*, *ystart*) to the point (*xend*, *yend*) in the coordinates specified with *coord*.

To add a line segment at a specified position in 3D grid coordinates, call **gpAddLineSegment3D**.

EXAMPLE

```
framenum = 2;
xstart = 2;
xend = 3.2;
ystart = 4.75;
yend = 5.75;
coord = 2;
ret = gpAddLineSegment(&gp,framenum,"line1",xstart,ystart,
                       xend,yend,coord);
ret = gpSetLineSegmentColor(&gp,framenum,"line1","blue");
```

This example adds a line segment to frame two, drawing the line segment from the point (2, 4.75) to the point (3.2, 5.75) in grid coordinates. It then sets the color of the line segment to blue.

SOURCE gplinesegment.src

SEE ALSO **gpAddLineSegment3D**, **gpSetLineSegmentColor**, **gpSetLineSegmentThickness**

gpAddLineSegment3D

PURPOSE Adds a line segment to a frame at a specified position in 3D grid coordinates.

LIBRARY gaussplot

FORMAT *ret* = **gpAddLineSegment3D**(*pgp*, *framenum*, *name*, *xstart*, *ystart*, *zstart*, *xend*, *yend*, *zend*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name for the line segment object.
xstart	scalar, X-coordinate of the point at which the line segment begins.
ystart	scalar, Y-coordinate of the point at which the line segment begins.
zstart	scalar, Z-coordinate of the point at which the line segment begins.
xend	scalar, X-coordinate of the point at which the line segment ends.
yend	scalar, Y-coordinate of the point at which the line segment ends.
zend	scalar, Z-coordinate of the point at which the line segment ends.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpAddLineSegment3D** allows you to specify a 3D grid coordinate position for the line segment. Call **gpAddLineSegment** to add a line segment at a specified position in frame coordinates or 2D grid coordinates.

Specifying a name for the line segment object allows you to reference the object in other commands that set line segment options. The line segment will be drawn from the point (*xstart*, *ystart*, *zstart*) to the point (*xend*, *yend*, *zend*) in 3D grid coordinates.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 2;
xstart = 2;
xend = 3.2;
ystart = 4.75;
yend = 5.75;
zstart = 3;
zend = 5;
ret = gpAddLineSegment3D(&gp,framenum,"line1",xstart,ystart,
                         zstart,xend,yend,zend);
ret = gpSetLineSegmentColor(&gp,framenum,"line1","blue");
```

This example adds a line segment to frame two, drawing the line segment from the point (2, 4.75, 3) to the point (3.2, 5.75, 5) in 3D grid coordinates. It then sets the color of the line segment to blue.

SOURCE gplinesegment.src

SEE ALSO **gpAddLineSegment**, **gpSetLineSegmentColor**, **gpSetLineSegmentThickness**

gpAddLogLogFrame

PURPOSE Adds a LogLog plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpLogLogPlotCreate**.

LIBRARY gaussplot

gpAddLogLogFrame

FORMAT **gpAddLogLogFrame**(*pgp*);

INPUT *pgp* pointer to a plot control structure.

REMARKS A LogLog plot is simply an XY Line plot that uses log scaling on the X and Y axes.

By default, **gpAddLogLogFrame** creates a frame with one line map, associating the first variable in the data file with the X-axis, and the second variable with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE
```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAddLogLogFrame(&gp);
gpAddXYLineFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
```

```
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a LogLog plot, and the second an XY Line plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least two variables.

SOURCE `gplogplot.src`

SEE ALSO **gpPlotCreate**, **gpLogLogPlotCreate**, **gpPlot**

gpAddLogXFrame

PURPOSE Adds a LogX plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpLogXPlotCreate**.

LIBRARY `gaussplot`

FORMAT **gpAddLogXFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS A LogX plot is simply an XY Line plot that uses log scaling on the X-axis.

By default, **gpAddLogXFrame** creates a frame with one line map, associating the first variable in the data file with the X-axis, and the second variable with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAddLogXFrame(&gp);
gpAddXYLineFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a LogX plot, and the second an XY Line plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least two variables.

SOURCE `gplogplot.src`

SEE ALSO **gpPlotCreate**, **gpLogXPlotCreate**, **gpPlot**

gpAddLogYFrame

PURPOSE Adds a LogY plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpLogYPlotCreate**.

LIBRARY gaussplot

FORMAT **gpAddLogYFrame**(*pgp*);

INPUT *pgp* pointer to a plot control structure.

REMARKS A LogY plot is simply an XY Line plot that uses log scaling on the Y-axis.

By default, **gpAddLogYFrame** creates a frame with one line map, associating the first variable in the data file with the X-axis, and the second variable with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE
```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAddLogYFrame(&gp);
gpAddXYLineFrame(&gp);

framenum = 1;
xpos = 0.25;
```

```
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a LogY plot, and the second an XY Line plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least two variables.

SOURCE `gplogplot.src`

SEE ALSO **gpPlotCreate**, **gpLogYPlotCreate**, **gpPlot**

gpAddMacroFile

PURPOSE Adds one or more macro files to a plot control structure, to be run at the end of the macro file created with the structure data.

LIBRARY `gaussplot`

FORMAT **gpAddMacroFile(***pgp*, *macrofile***);**

INPUT *pgp* pointer to a plot control structure.

macrofile string or N×1 string array, name(s) of macro file(s) created with **GAUSSplot**.

REMARKS When you call **gpPlot** or any of the other plotting commands (see Section 2.1.16) to plot a graph based on the information in a plot control structure, **GAUSSplot** writes the information out as a macro file and then runs the macro file to produce the graph. Call this command to run one or more previously created macro files at the end of the macro file that produces your graph.

The **gpAddMacroFile** command allows you to create a graph using **GAUSSplot** commands, modify it in the **GAUSSplot** GUI in 'Macro Record' mode (by selecting the "Macro"→"Record" option from the **GAUSSplot** 'File' menu), and then add those modifications to the program that created the graph. See Section 1.2.1 for detailed instructions.

EXAMPLE
```
gpAddMacroFile(&gp,"macro.mcr");
```

SOURCE gp.src

SEE ALSO **gpRunMacroFile**

gpAddPolarLineFrame

PURPOSE Adds a Polar Line plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpPolarLinePlotCreate**.

LIBRARY gaussplot

FORMAT **gpAddPolarLineFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS By default, **gpAddPolarLineFrame** creates a frame with one line map, associating the first variable in the data file with the Theta-axis, and the second

variable with the R-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAddPolarLineFrame(&gp);
gpAddBarFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a Polar Line plot, and the second a Bar plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least two variables.

SOURCE gppolarlineplot.src

SEE ALSO **gpPlotCreate**, **gpPolarLinePlotCreate**, **gpPlot**

gpAddRectangle

PURPOSE Adds a rectangular geometry object to the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpAddRectangle**(*pgp*, *framenum*, *name*, *xstart*, *ystart*, *xlen*, *ylen*, *coord*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name for the line segment object.
xstart	scalar, horizontal position of the bottom left corner of the rectangle.
ystart	scalar, vertical position of the bottom left corner of the rectangle.
xlen	scalar, width of the rectangle in X plot coordinates or horizontal frame coordinates.
ylen	scalar, height of the rectangle in Y plot coordinates or vertical frame coordinates.
coord	scalar, position coordinate system, one of the following: **1** starting position, width, and height in frame coordinates, 0-100, where (0, 0) is lower left corner of frame. **2** starting position, width and height in 2D grid coordinates, i.e., X and Y plot coordinates.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Specifying a name for the rectangular geometry object allows you to reference the object in other commands that set options for the object. The following commands may be used to set options for rectangles and other types of geometries that have been added to your graph:

gpSetGeometryColor

gpSetGeometryLineThickness

gpSetGeometryLinePattern

gpSetGeometryLinePatternLength

gpFillGeometry

gpSetGeometryFillColor

EXAMPLE

```
framenum = 2;
xstart = 4;
xlen = 1;
ystart = 0;
ylen = 5;
coord = 2;
ret = gpAddRectangle(&gp,framenum,"rec1",xstart,ystart,
                        xlen,ylen,coord);
ret = gpSetGeometryColor(&gp,framenum,"rec1","green");
```

This example adds a rectangle with a width of 1 and a height of 5 to frame two, beginning (with the bottom left corner) at the point (4,0). It then sets the color of the rectangle to green.

SOURCE gpgeometry.src

SEE ALSO **gpAddSquare**, **gpSetGeometryColor**, **gpSetGeometryLineThickness**

gpAddSquare

PURPOSE Adds a square geometry object to the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpAddSquare**(*pgp*, *framenum*, *name*, *xstart*, *ystart*, *xlen*, *coord*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name for the line segment object.
xstart	scalar, horizontal position of the bottom left corner of the square.
ystart	scalar, vertical position of the bottom left corner of the square.
xlen	scalar, length of one side of the square in X plot coordinates or horizontal frame coordinates.
coord	scalar, position coordinate system, one of the following: **1** starting position, width, and height in frame coordinates, 0-100, where (0, 0) is lower left corner of frame. **2** starting position, width and height in 2D grid coordinates, i.e., X and Y plot coordinates.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Specifying a name for the square geometry object allows you to reference the object in other commands that set options for the object. The following commands may be used to set options for squares and other types of geometries that have been added to your graph:

gpSetGeometryColor

gpSetGeometryLineThickness

gpSetGeometryLinePattern

gpSetGeometryLinePatternLength

gpFillGeometry

gpSetGeometryFillColor

EXAMPLE

```
framenum = 2;
xstart = 2;
ystart = 1;
xlen = 3;
coord = 2;
ret = gpAddSquare(&gp,framenum,"sq1",xstart,ystart,
                        xlen,coord);
ret = gpSetGeometryLineThickness(&gp,framenum,"sq1",0.4);
```

This example adds a square with sides of length 3 to frame two, beginning (with the bottom left corner) at the point (2,1). It then sets the line thickness of the square to 0.4.

SOURCE gpgeometry.src

SEE ALSO **gpAddRectangle**, **gpSetGeometryColor**, **gpSetGeometryLineThickness**

gpAddTextObject

PURPOSE Adds a text object to the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpAddTextObject**(*pgp*, *framenum*, *name*, *xpos*, *ypos*, *coord*, *text*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	name	string, name for the text object.
	xpos	scalar, X position.
	ypos	scalar, Y position.
	coord	scalar, position coordinate system, one of the following: **1** frame coordinates, 0-100, where (0,0) is the lower left corner of the frame. **2** 2D grid coordinates, i.e., X and Y plot coordinates.
	text	string or N×1 string array, text.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS To add a text object to a frame at a specified position in 3D plot coordinates, call **gpAddTextObject3D**.

Specifying a name for the text object with the *name* argument allows you to reference the object in other commands that set text options. You may set *name* to a null string if you do not need to reference the text object later.

To add multi-line text to the frame, input an N×1 string array for *text*; this will create N lines of text in the frame.

By default, the text in a text object is written in a helvetica bold font. To change the font for a text object, call **gpSetTextObjectFont**. To use one font inside another (e.g. to insert a Greek or mathematic symbol into a normal text string), use one of the following substrings:

<courier> ... < /courier>	Use the courier font for the enclosed text.
<helvetica> ... < /helvetica>	Use the helvetica font for the enclosed text.
<times> ... < /times>	Use the times font for the enclosed text.
<greek> ... < /greek>	Use the Greek font for the enclosed text.
<math> ... < /math>	Use the math font for the enclosed text.
<userdef> ... < /userdef>	Use the user-defined font for the enclosed text.

For example:

```
y = r*sin(<greek>q</greek>)
```

produces the string:

y = r*sin(θ)

See Appendix A to see a complete list of the available characters in **GAUSSplot**.

The following formatting substrings are also supported:

<b> ... < /b>	Bold the enclosed text (not always supported).
<i> ... < /i>	Italicize the enclosed text (not always supported).
<sub> ... < /sub>	Subscript the enclosed text.
<sup> ... < /sup>	Superscript the enclosed text.
<verbatim> ... < /verbatim>	Write the enclosed text verbatim.

These font and formatting substrings can be nested inside one another or used together, for example:

```
Use <courier><b>courier bold</courier></b> for emphasis.
```

sets the text “courier bold” in a courier bold font.

EXAMPLE

```
framenum = 2;
xpos = 4.5;
ypos = 10;
coord = 2;
ret = gpAddTextObject(&gp,framenum,"text1",xpos,ypos,
                              coord,"f(x) = log(x+1)");
ret = gpSetTextObjectColor(&gp,framenum,"text1","blue");
```

This example adds a text object to frame two, placing the object so that its lower left corner is at the point (4.5, 10), and changes the text color to blue.

SOURCE gptext.src

SEE ALSO **gpAddTextObject3D**, **gpSetTextObjectColor**, **gpSetTextObjectFont**, **gpSetTextObjectAnchorLocation**

gpAddTextObject3D

PURPOSE Adds a text object to a frame at a specified position in 3D grid coordinates.

LIBRARY gaussplot

FORMAT *ret* = **gpAddTextObject3D**(*pgp*, *framenum*, *name*, *xpos*, *ypos*, *zpos*, *text*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name for the text object.
xpos	scalar, X position in X plot coordinates.
ypos	scalar, Y position in Y plot coordinates.
zpos	scalar, Z position in Z plot coordinates:
text	string or N×1 string array, text.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpAddTextObject3D** allows you to specify a 3D grid coordinate position for the text object. Call **gpAddTextObject** to add a text object at a specified position in frame coordinates or 2D grid coordinates.

Setting a name for the text object with the *name* argument allows you to reference the object in other commands that set text options. You may set *name* to a null string if you do not need to reference the text object later.

To add multi-line text to the frame, input an N×1 string array for *text*; this will create N lines of text in the frame.

By default, the text in a text object is written in a helvetica bold font. To change the font for a text object, call **gpSetTextObjectFont**. See **gpAddTextObject** for information on using one font inside another.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 2;
xpos = 4.5;
ypos = 10;
zpos = 4;
ret = gpAddTextObject3D(&gp,framenum,"text1",xpos,
                            ypos,zpos,"f(x,y) = sin(x)*cos(y)");
ret = gpSetTextObjectColor(&gp,framenum,"text1","blue");
```

This example adds a text object to frame two at the point (4.5, 10, 4) and changes the text color to blue.

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectColor**, **gpSetTextObjectFont**, **gpSetTextObjectAnchorLocation**

gpAddXYLineFrame

PURPOSE Adds an XY Line plot frame to a **gpPlotControl** structure created with **gpPlotCreate** or **gpXYLinePlotCreate**.

LIBRARY gaussplot

FORMAT **gpAddXYLineFrame(***pgp***);**

INPUT *pgp* pointer to a plot control structure.

REMARKS By default, **gpAddXYLineFrame** creates a frame with one line map, associating the first variable in the data file with the X-axis, and the second variable with the Y-axis. Use **gpSetLineMaps** to use different variables or to specify multiple line maps.

Frames are numbered in the order in which they are added, beginning with one.

Use **gpPlot** to draw the graph.

EXAMPLE
```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAddXYLineFrame(&gp);
gpAddBarFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains an XY

Line plot, and the second a Bar plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata.plt` already exists and contains at least two variables.

SOURCE `gpxylineplot.src`

SEE ALSO **gpPlotCreate**, **gpXYLinePlotCreate**, **gpPlot**

gpAddZone

PURPOSE Adds the data for one zone to a **gpData** structure.

LIBRARY `gaussplot`

FORMAT *ret* = **gpAddZone**(*pgdat*, *zonename*, *data*);

INPUT

pgdat	pointer to a **GAUSSplot** data structure initialized by **gpInitPlotData**.
zonename	string, the name of the zone.
data	N×K matrix containing K variables that are N×1. - or - 3-dimensional K×N×P array containing K variables that are N×P. - or - 4-dimensional K×L×N×P array containing K variables that are L×N×P.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS If *zonename* is a null string, a default zone name will be created. For an explanation of the structure of a `.plt` data file, see Section 1.3.

For Bar, XY Line and Polar Line plots, *data* should be either an N×K matrix or a 3-dimensional array containing K variables that are N×1. For 2D Cartesian

and 2D Contour plots, *data* should be a 3-dimensional array containing K variables that are N×P. For 3D Cartesian and 3D Contour plots, *data* may be either a 3-dimensional array containing K 2-dimensional variables (to plot a surface) or a 4-dimensional array containing K 3-dimensional variables (to plot a volume). See Section 1.3.3 for more information on inputting data.

EXAMPLE

```
library gaussplot;
#include gp.sdf

p = pi;
x1 = seqa(0,p/4,15);
y1 = sin(x1);
a1 = aconcat(x1,y1,3);

x2 = seqa(0,p/6,20);
y2 = cos(x2);
a2 = aconcat(x2,y2,3);

struct gpData gdat;
string vnames = { "X", "Y" };
gdat = gpInitPlotData(vnames);
ret = gpAddZone(&gdat,"Zone 1",a1);
ret = gpAddZone(&gdat,"Zone 2",a2);
ret = gpWritePlotData(&gdat,"mydata.plt");
```

SOURCE gpwritedata.src

SEE ALSO **gpInitPlotData**, **gpWritePlotData**, **gpSetPlotData**

gpAnimate3DRotation

PURPOSE Animates the rotation of a plot about the X, Y and Z axes.

LIBRARY gaussplot

gpAnimate3DRotation

FORMAT *ret* = **gpAnimate3DRotation**(*pgp*, *framenum*, *iter*, *xstep*, *ystep*, *zstep*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
iter	scalar, number of iterations in animation loop.
xstep	scalar, $-180 \leq xstep \leq 180$, rotation about the X-axis in degrees.
ystep	scalar, $-180 \leq ystep \leq 180$, rotation about the Y-axis in degrees.
zstep	scalar, $-180 \leq zstep \leq 180$, rotation about the Z-axis in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS In each iteration of the animation loop, the plot will be rotated *xstep* degrees about the X-axis, *ystep* degrees about the Y-axis, and *zstep* degrees about the Z-axis in that order. If *xstep*, *ystep* or *zstep* is set to zero, then the plot will not be rotated about the corresponding axis.

To animate rotation about one axis, call **gpAnimateXAxisRotation**, **gpAnimateYAxisRotation** or **gpAnimateZAxisRotation**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
iter = 50;
xstep = 5;
ystep = 3;
zstep = 4;
ret = gpAnimate3DRotation(&gp,framenum,iter,xstep,ystep,
                          zstep);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAnimateXAxisRotation**, **gpAnimateYAxisRotation**, **gpAnimateZAxisRotation**, **gpAnimateFieldZones**

gpAnimateFieldZones

PURPOSE Cycles through the specified field zones one at a time to produce animation.

LIBRARY gaussplot

FORMAT *ret* = **gpAnimateFieldZones**(*pgp*, *framenum*, *iter*, *first*, *last*, *skip*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
iter	scalar, number of iterations in the animation loop.
first	scalar, number of the first field zone in the animation loop.
last	scalar, number of the last field zone in the animation loop.
skip	scalar, show only one in every *skip* field zones between *first* and *last*.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Set *skip* to 1 if you do not want to skip any field zones.

This command does NOT apply to frames that contain Bar, XY Line or Polar Line plots. To animate line maps in a frame containing one of these plot types, call **gpAnimateLineMaps**.

EXAMPLE

```
framenum = 1;
iter = 10;
first = 2;
last = 200;
skip = 2;
```

```
ret = gpAnimateFieldZones(&gp,framenum,iter,first,last,skip);
```

This example produces an animation loop that runs 10 times, cycling through all of the even-numbered field zones starting with 2 and ending with 200.

SOURCE gpfieldzone.src

SEE ALSO **gpAnimate3DRotation**, **gpAnimateLineMaps**

gpAnimateLineMaps

PURPOSE Cycles through the specified line maps one at a time to produce animation.

LIBRARY gaussplot

FORMAT *ret* = **gpAnimateLineMaps**(*pgp*, *framenum*, *iter*, *first*, *last*, *skip*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
iter	scalar, number of iterations in the animation loop.
first	scalar, number of the first line map in the animation loop.
last	scalar, number of the last line map in the animation loop.
skip	scalar, show only one in every *skip* line maps between *first* and *last*.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Set *skip* to 1 if you do not want to skip any line maps.

This command applies ONLY to frames that contain Bar, XY Line and Polar Line plots. To animate field zones in a frame containing one of the other plot types, call **gpAnimateFieldZones**.

EXAMPLE

```
framenum = 1;
iter = 10;
first = 1;
last = 131;
skip = 2;
ret = gpAnimateLineMaps(&gp,framenum,iter,first,last,skip);
```

This example produces an animation loop that runs 10 times, cycling through all of the odd-numbered line maps starting with 1 and ending with 131.

SOURCE gplinemap.src

SEE ALSO **gpAnimateFieldZones**

gpAnimateXAxisRotation

PURPOSE Animates the rotation of a plot about the X-axis.

LIBRARY gaussplot

FORMAT *ret* = **gpAnimateXAxisRotation**(*pgp*, *framenum*, *iter*, *xstep*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
iter	scalar, number of iterations in animation loop.
xstep	scalar, $-180 \leq xstep \leq 180$, rotation about the X-axis in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS In each iteration of the animation loop, the plot will be rotated *xstep* degrees about the X-axis.

To animate rotation about the X, Y and Z axes, call **gpAnimate3DRotation**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
iter = 72;
xstep = 5;
ret = gpAnimateXAxisRotation(&gp,framenum,iter,xstep);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAnimate3DRotation**

gpAnimateYAxisRotation

PURPOSE Animates the rotation of a plot about the Y-axis.

LIBRARY gaussplot

FORMAT *ret* = **gpAnimateYAxisRotation**(*pgp*, *framenum*, *iter*, *ystep*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
iter	scalar, number of iterations in animation loop.
ystep	scalar, $-180 \leq ystep \leq 180$, rotation about the Y-axis in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS In each iteration of the animation loop, the plot will be rotated *ystep* degrees about the Y-axis.

To animate rotation about the X, Y and Z axes, call **gpAnimate3DRotation**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
iter = 72;
ystep = 5;
ret = gpAnimateYAxisRotation(&gp,framenum,iter,ystep);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAnimate3DRotation**

gpAnimateZAxisRotation

PURPOSE Animates the rotation of a plot about the Z-axis.

LIBRARY gaussplot

FORMAT *ret* = **gpAnimateZAxisRotation**(*pgp*, *framenum*, *iter*, *zstep*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
iter	scalar, number of iterations in animation loop.
zstep	scalar, $-180 \leq zstep \leq 180$, rotation about the Z-axis in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS In each iteration of the animation loop, the plot will be rotated *zstep* degrees about the Z-axis.

To animate rotation about the X, Y and Z axes, call **gpAnimate3DRotation**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
iter = 72;
zstep = 5;
ret = gpAnimateZAxisRotation(&gp,framenum,iter,zstep);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAnimate3DRotation**

gpBarPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a Bar plot.

LIBRARY gaussplot

FORMAT *gp* **= gpBarPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a Bar plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeBarPlot** or **gpPlot**. Calling **gpBarPlotCreate** sets the plot type of the entire graph to Bar plot. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpBarPlotCreate** contains one frame by default.

EXAMPLE
```
library gaussplot;
#include gp.sdf

p = pi;
```

```
x = seqa(0,p/4,15);
y = sin(x);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpBarPlotCreate;
ret = gpMakeBarPlot(&gp,x,y,vnames);
```

This example illustrates how you may create a bar plot with one line map, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, and the second variable (**V2**) with the Y-axis.

SOURCE gpbarplot.src

SEE ALSO **gpMakeBarPlot**, **gpPlot**

gpBoxPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a Box plot (box graph percentile plot).

LIBRARY gaussplot

FORMAT *gp* = **gpBoxPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a Box plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeBoxPlot** or **gpPlot**. Calling **gpBoxPlotCreate** sets the plot type of the entire graph to Box. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

If you use **gpBoxPlotCreate** to initialize a plot control structure for a

multiple-frame graph, you will need to call **gpAddBoxFrame** for each frame in the graph. A graph initialized with **gpBoxPlotCreate** does NOT contain one frame by default. However, there is no need to call **gpAddBoxFrame** if you are using **gpMakeBoxPlot** to plot the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpBoxPlotCreate;

rndseed 12345;
nboxes = 5;
npoints = 7;
x=seqa(1,1,nboxes)';
y=rndn(npoints,nboxes);

call gpMakeBoxPlot(&gp,x,y);
```

This example illustrates how you may create a Box Plot with five line maps, using defaults for all style and display options. **gpMakeBoxPlot**

SOURCE gphistplot.src

SEE ALSO **gpMakeBoxPlot**, **gpPlot**

gpCleanUp

PURPOSE Deletes the data, macro, and structure files associated with a graph.

LIBRARY gaussplot

FORMAT *ret* = **gpCleanUp**(*pgp*);

INPUT *pgp* pointer to a plot control structure.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Call **gpCleanUp** after creating a graph to delete the data, macro, and structure files associated with the graph (see Section 1.8 for more information on structure files). This is especially important if you have not specified names for the data and macro files, causing **GAUSSplot** to generate temporary file names.

The names of these files are stored in the **gpPlotControl** structure associated with the graph. Thus if you want to call **gpCleanUp**, you must do so before deleting the **gpPlotControl** structure from the **GAUSS** workspace.

EXAMPLE
```
ret = gpCleanUp(&gp);
```

SOURCE gp.src

SEE ALSO **gpPlot**

gpCreateHistData

PURPOSE Generates vectors of category breakpoints, category midpoints, and frequency counts for a Hist plot (frequency histogram).

LIBRARY gaussplot

FORMAT { *b*, *m*, *freq*, } = **gpCreateHistData**(*x*, *v*);

INPUT *x* N×1 vector, vector of data.

v L×1 vector, the breakpoints to be used to compute the frequencies
– or –
scalar, the number of categories.

OUTPUT

b	P×1 vector, the breakpoints used for each category.
m	P×1 vector, the midpoints of each category.
freq	P×1 vector, computed frequency counts.

REMARKS

If *v* is an L×1 vector, a final breakpoint equal to the maximum value of *x* will be added to the vector of breakpoints if the maximum breakpoint value is smaller. If *v* is a scalar, this command generates *v* evenly-spaced categories. The breakpoints of the categories are written to *b* and the midpoints of the categories to *m*.

Each time an element falls into one of the categories specified in *b*, the corresponding element of *freq* will be incremented by one. The categories are interpreted as follows:

$$
\begin{array}{lcccccl}
freq[1] & = & & & x & <= & b[1] \\
freq[2] & = & b[1] & < & x & <= & b[2] \\
freq[3] & = & b[2] & < & x & <= & b[3] \\
 & \vdots & & & & & \\
freq[P] & = & b[P-1] & < & x & <= & b[P]
\end{array}
$$

To create your Hist plot, write *m* and *freq* to your `.plt` data file, and then call **gpAddHistPlotFrame** to add a Hist Plot frame and **gpPlot** to generate the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Initialize plot control structure
struct gpPlotControl gp;
gp = gpPlotCreate;

// Add frames
gpAddHistFrame(&gp);
```

```
gpAddXYLineFrame(&gp);

// Generate data and write data file for first frame
struct gpData gdat;
x = round(rndn(4000,1)*100);
{ b,m,freq } = gpCreateHistData(x,50);
string vnames = { "Cat", "Freq" };
gdat = gpSetPlotData(m~freq,vnames);
ret = gpWritePlotData(&gdat,"mydata_hist.plt");

// Set data files for both frames
ret = gpSetDataFile(&gp,1,"mydata_hist.plt");
ret = gpSetDataFile(&gp,2,"mydata_xyline.plt");

// Position frame 1
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,1,xpos,ypos,width,height);

// Position frame 2
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,2,xpos,ypos,width,height);

// Plot the graph
ret = gpPlot(&gp);
```

This example creates a graph with two frames, the first containing a Hist plot, and the second an XY Line plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame. This example assumes that `mydata_xyline.plt` already exists and contains at least two variables.

SOURCE gphistplot.src

SEE ALSO **gpHistPlotCreate**, **gpPlot**

gpCreateHistPData

PURPOSE Generates vectors of category breakpoints, category midpoints, frequency counts, and frequency percentages for a HistP plot (percentage histogram).

LIBRARY gaussplot

FORMAT { *b*, *m*, *freqct*, *freqp* } = **gpCreateHistPData**(*x*, *v*);

INPUT

x	N×1 vector, vector of data.
v	L×1 vector, the breakpoints to be used to compute the frequencies – or – scalar, the number of categories.

OUTPUT

b	P×1 vector, the breakpoints used for each category.
m	P×1 vector, the midpoints of each category.
freqct	P×1 vector, computed frequency counts.
freqp	P×1 vector, computed frequency percentages.

REMARKS If *v* is an L×1 vector, a final breakpoint equal to the maximum value of *x* will be added to the vector of breakpoints if the maximum breakpoint value is smaller. If *v* is a scalar, this command generates *v* evenly-spaced categories. The breakpoints of the categories are written to *b* and the midpoints of the categories to *m*.

Each time an element falls into one of the categories specified in *b*, the corresponding element of *freq* will be incremented by one. The categories are

interpreted as follows:

$$
\begin{array}{lllllll}
freq[1] & = & & & x & <= & b[1] \\
freq[2] & = & b[1] & < & x & <= & b[2] \\
freq[3] & = & b[2] & < & x & <= & b[3] \\
\quad . & & & & & & \\
\quad . & & & & & & \\
\quad . & & & & & & \\
freq[P] & = & b[P-1] & < & x & <= & b[P]
\end{array}
$$

To create your HistP plot, write *m* and *freqp* to your `.plt` data file, and then call **gpAddHistPPlotFrame** to add a HistP Plot frame and **gpPlot** to generate the graph.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

// Create random data
rndseed 454356;
x = round(rndn(4000,1)*100);
{ b, m, freq, freqp } = gpCreateHistPData(x,50);

// Set data in gpdata structure
struct gpData gdat;
string vnames = { "Cat", "Freq", "Prcnt" };
gdat = gpSetPlotData(m~freq~freqp,vnames);

// Write data file
ret = gpWritePlotData(&gdat,"hist.plt");
if (ret);
 print gpGetErrorMessage(ret);
endif;
```

```
// Set data file
call gpSetDataFile(&gp,0,"hist.plt");

// Set up frame 1
call gpAddHistFrame(&gp);
call gpSetLineMaps(&gp,1,1~2);
call gpMoveFrame(&gp,1,0.25,0.25,5.175,7.85);

// Set up frame 2
call gpAddHistPFrame(&gp);
call gpSetLineMaps(&gp,1,1~3);
call gpSetBarColor(&gp,2,1,"blue","blue");
call gpMoveFrame(&gp,2,5.525,0.25,5.175,7.85);

// Plot the graph
call gpPlot(&gp);
```

This example creates a graph with two frames, the first containing a Hist plot, and the second a HistP plot.

SOURCE gphistplot.src

SEE ALSO **gpHistPPlotCreate**, **gpPlot**

gpExportBMP

PURPOSE Exports a graph to a Windows Bitmap (BMP) format graphics file.

LIBRARY gaussplot

FORMAT *ret* = **gpExportBMP**(*pgp*, *area*, *filename*);

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

-1 export all frames.

- or -

scalar, number of frame to export.

filename string, name of BMP output file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportBMP** sets an option to export the graph to a BMP format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE
```
area = -1;
ret = gpExportBMP(&gp,area,"graphic.bmp");
```

This example exports all frames to the BMP graphics file `graphic.bmp`.

SOURCE `gp.src`

SEE ALSO **gpSetExportImageWidth**, **gpExportJPEG**, **gpExportPS**, **gpExportWMF**

gpExportEPS

PURPOSE Exports a graph to an Encapsulated PostScript (EPS) format graphics file.

LIBRARY `gaussplot`

FORMAT *ret* = **gpExportEPS**(*pgp*, *area*, *filename*);

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.
-1 export all frames.
- or -
scalar, number of frame to export.

filename string, name of EPS output file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportEPS** sets an option to export the graph to an EPS format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE
```
area = 3;
ret = gpExportEPS(&gp,area,"graphic.eps");
```

This example exports frame three to the Encapsulated PostScript graphics file `graphic.eps`.

SOURCE `gp.src`

SEE ALSO **gpSetPrintPalette**, **gpExportPS**

gpExportHPGL

PURPOSE Exports a graph to an HP-GL format graphics file.

LIBRARY `gaussplot`

FORMAT *ret* = **gpExportHPGL**(*pgp*, *area*, *filename*);

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

-1 export all frames.

- or -

scalar, number of frame to export.

filename string, name of HP-GL output file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportHPGL** sets an option to export the graph to an HP-GL format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE
```
area = -1;
ret = gpExportHPGL(&gp,area,"graphic.hp");
```

This example exports all frames to the HP-GL graphics file `graphic.hp`.

SOURCE `gp.src`

SEE ALSO **gpExportHPGL2**, **gpExportPS**

gpExportHPGL2

PURPOSE Exports a graph to an HP-GL/2 format graphics file.

LIBRARY `gaussplot`

FORMAT *ret* = **gpExportHPGL2**(*pgp*, *area*, *filename*);

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

-1 export all frames.
- or -
scalar, number of frame to export.

filename string, name of HP-GL/2 output file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportHPGL2** sets an option to export the graph to an HP-GL/2 format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE
```
area = -1;
ret = gpExportHPGL2(&gp,area,"graphic.hp");
```

This example exports all frames to the HP-GL/2 graphics file `graphic.hp2`.

SOURCE `gp.src`

SEE ALSO **gpSetPrintPalette**, **gpExportHPGL**, **gpExportPS**

gpExportJPEG

PURPOSE Exports a graph to a Joint Photographic Experts Group (JPEG) format graphics file.

LIBRARY `gaussplot`

FORMAT *ret* = **gpExportJPEG**(*pgp*, *area*, *filename*);

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

		-1 export all frames. - or - scalar, number of frame to export.
	filename	string, name of JPEG output file.

OUTPUT

	ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportJPEG** sets an option to export the graph to a JPEG format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

You may call **gpSetExportJPEGQuality** to set the quality of the exported JPEG file. The range for the output file quality is 1 - 100. By default, the output file quality is set to 75.

EXAMPLE
```
area = -2;
ret = gpExportJPEG(&gp,area,"graphic.jpg");
```

This example exports the entire workspace to the JPEG graphics file `graphic.jpg`.

SOURCE `gp.src`

SEE ALSO **gpSetExportJPEGQuality**, **gpSetExportImageWidth**, **gpExportBMP**, **gpExportPS**

gpExportPNG

PURPOSE Exports a graph to a Portable Network Graphics (PNG) file.

LIBRARY `gaussplot`

FORMAT *ret* **= gpExportPNG(***pgp***,** *area***,** *filename***);**

INPUT

pgp pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

-1 export all frames.

- or -

scalar, number of frame to export.

filename string, name of PNG output file.

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

Calling **gpExportPNG** sets an option to export the graph to a PNG format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE

```
area = 3;
ret = gpExportPNG(&gp,area,"graphic.png");
```

This example exports frame three to the PNG graphics file `graphic.png`.

SOURCE

`gp.src`

SEE ALSO

gpSetExportImageWidth, **gpExportBMP**, **gpExportJPEG**, **gpExportPS**

gpExportPS

PURPOSE

Exports a graph to a PostScript (PS) format graphics file.

LIBRARY

`gausssplot`

FORMAT

ret **= gpExportPS(***pgp***,** *area***,** *filename***);**

INPUT

pgp pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

-1 export all frames.

- or -

scalar, number of frame to export.

filename string, name of PS output file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportPS** sets an option to export the graph to a PostScript format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE
```
area = 3;
ret = gpExportPS(&gp,area,"graphic.ps");
```

This example exports frame three to the PostScript graphics file `graphic.ps`.

SOURCE `gp.src`

SEE ALSO **gpSetPrintPalette**, **gpExportEPS**, **gpExportHPGL**

gpExportRAS

PURPOSE Exports a graph to a Sun Raster (RAS) format graphics file.

LIBRARY `gaussplot`

FORMAT *ret* = **gpExportRAS**(*pgp*, *area*, *filename*);

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.
-1 export all frames.
- or -
scalar, number of frame to export.

filename string, name of RAS output file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportRAS** sets an option to export the graph to a Sun Raster format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE
```
area = -2;
ret = gpExportRAS(&gp,area,"graphic.ras");
```

This example exports the entire workspace to the Sun Raster graphics file `graphic.ras`.

SOURCE `gp.src`

SEE ALSO **gpSetExportImageWidth**, **gpExportBMP**, **gpExportJPEG**, **gpExportPS**

gpExportTIFF

PURPOSE Exports a graph to a Tagged Image File Format (TIFF) graphics file.

LIBRARY `gaussplot`

FORMAT *ret* **= gpExportTIFF(***pgp***,** *area***,** *filename***);**

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

-1 export all frames.

- or -

scalar, number of frame to export.

filename string, name of TIFF output file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportTIFF** sets an option to export the graph to a TIFF format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE
```
area = 2;
ret = gpExportTIFF(&gp,area,"graphic.tif");
```

This example exports frame two to the TIFF graphics file `graphic.tif`.

SOURCE `gp.src`

SEE ALSO **gpSetExportImageWidth**, **gpSetPrintPalette**, **gpExportBMP**, **gpExportJPEG**

gpExportWMF

PURPOSE Exports a graph to a Windows Meta File (WMF) format graphics file.

LIBRARY `gaussplot`

FORMAT *ret* **= gpExportWMF(***pgp***,** *area***,** *filename***);**

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

-1 export all frames.

- or -

scalar, number of frame to export.

filename string, name of WMF output file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportWMF** sets an option to export the graph to a WMF format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE
```
area = 2;
ret = gpExportWMF(&gp,area,"graphic.wmf");
```

This example exports frame two to the WMF graphics file `graphic.wmf`.

SOURCE `gp.src`

SEE ALSO **gpSetPrintPalette**, **gpExportBMP**, **gpExportJPEG**

gpExportXWD

PURPOSE Exports a graph to an X-Windows Format (XWD) graphics file.

LIBRARY `gaussplot`

FORMAT *ret* **= gpExportXWD(***pgp***,** *area***,** *filename***);**

INPUT *pgp* pointer to a plot control structure.

area scalar, area of the graph to export, one of the following:

-2 export entire workspace.

		-1 export all frames. - or - scalar, number of frame to export.
	filename	string, name of XWD output file.

OUTPUT

	ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpExportXWD** sets an option to export the graph to an X-Windows format graphics file once the graph has been created with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

EXAMPLE

```
area = 2;
ret = gpExportXWD(&gp,area,"graphic.xwd");
```

This example exports frame two to the X-Windows Format graphics file `graphic.xwd`.

SOURCE `gp.src`

SEE ALSO **gpSetExportImageWidth**, **gpExportBMP**, **gpExportJPEG**, **gpExportPS**

gpFillGeometry

PURPOSE Specifies whether or not a geometry object is filled.

LIBRARY `gaussplot`

FORMAT *ret* = **gpFillGeometry**(*pgp*, *framenum*, *name*, *fill*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a geometry object.

fill scalar, one of the following:

0 do not fill grid area.

1 fill grid area.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, geometries are not filled. If you call **gpFillGeometry** to specify that a geometry is to be filled, the default fill color is white. To change the fill color, call **gpSetGeometryFillColor**.

You may call **gpFillGeometry** to set specify whether or not a geometry created with any of the following commands is to be filled:

gpAddCircle
gpAddEllipse
gpAddRectangle
gpAddSquare

EXAMPLE
```
framenum = 1;
xstart = 0;
ystart = 0;
xradius = 1.5;
coord = 2;
thickness = 0.4;
fill = 1;
ret = gpAddCircle(&gp,framenum,"circ1",xstart,ystart,xradius,
                  coord);
ret = gpFillGeometry(&gp,framenum,"circ1",fill);
```

This example adds a circle of radius 1.5 to frame one, centered on the point (0,0). It then specifies that the circle is to be filled.

SOURCE gpgeometry.src

SEE ALSO **gpSetGeometryFillColor**, **gpSetGeometryColor**

gpFillGridArea

PURPOSE Specifies whether or not the grid area is filled.

LIBRARY gaussplot

FORMAT *ret* = **gpFillGridArea**(*pgp*, *framenum*, *fill*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fill	scalar, one of the following:
	0 do not fill grid area.
	1 fill grid area.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the grid area is filled for 3D Cartesian and 3D Contour plots and unfilled for all other plot types. The default fill color is white. To change the fill color, call **gpSetGridAreaFillColor**. Because grid area light source shading is used by default in 3D Cartesian and 3D Contour plots, the grid area appears as gray. To deactivate grid area light source shading, call **gpUseGridAreaLighting**.

EXAMPLE
```
framenum = 1;
fill = 1;
ret = gpFillGridArea(&gp,framenum,fill);
```

SOURCE gp.src

SEE ALSO **gpSetGridAreaFillColor**, **gpUseGridAreaLighting**, **gpPositionGrid**

gpFramePause

PURPOSE Pauses drawing for a specified number of seconds after drawing for a frame is completed.

LIBRARY `gaussplot`

FORMAT *ret* = **gpFramePause**(*pgp*, *framenum*, *pause*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
pause	scalar, number of seconds to pause drawing after the specified frame is completed.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The *pause* specified with **gpFramePause** is applicable only if each frame is shown as it is created. Thus this command is ignored unless **gpShowDrawing** is called with a *draw* argument of 2 (show each frame after the drawing for that frame is completed) or 3 (show all drawing). By default, *draw* is set to 1 (do not show drawing), and thus no pause is used.

EXAMPLE

```
draw = 2;
framenum = 2;
pause = 0.25;
gpShowDrawing(&gp,draw);
ret = gpFramePause(&gp,framenum,pause);
```

This example shows each frame after the drawing for that frame is completed, and pauses drawing for 0.25 seconds after the second frame is displayed.

SOURCE `gp.src`

SEE ALSO **gpShowDrawing**

gpGetErrorMessage

PURPOSE Returns the error message associated with a particular error code.

LIBRARY gaussplot

FORMAT *errmsg* = **gpGetErrorMessage**(*errcode*);

INPUT *errcode* scalar, error code.

OUTPUT *errmsg* string, error message.

REMARKS By default, if a **GAUSSplot** command fails, it returns an error code. Pass the value of the error code into **gpGetErrorMessage** to get the associated message.

To change the level of error handling, call **gpSetErrorLevel**.

EXAMPLE
```
library gaussplot;
#include gp.sdf

p = pi;
x = seqa(0,p/4,15);
y = sin(x);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpBarPlotCreate;
ret = gpMakeBarPlot(&gp,x,y,vnames);

if (ret);
    errmsg = gpGetErrorMessage(ret);
    print errmsg;
endif;
```

SOURCE gp.src

SEE ALSO **gpSetErrorLevel**

gpHistPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a Hist plot (frequency histogram).

LIBRARY gaussplot

FORMAT *gp* = **gpHistPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a Hist plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeHistPlot** or **gpPlot**. Calling **gpHistPlotCreate** sets the plot type of the entire graph to Hist. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpHistPlotCreate** contains one frame by default.

EXAMPLE
```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpHistPlotCreate;

x = round(rndn(4000,1)*100);
string vnames = { "Cat", "Freq" };
```

```
{ b,m,freq,ret } = gpMakeHistPlot(&gp,x,50,vnames);
```

This example illustrates how you may create a frequency histogram with one line map, using defaults for all style and display options. **gpMakeHistPlot** breaks up **x** into 50 evenly spaced categories, and associates the categories with the X-axis and the frequencies with the Y-axis.

SOURCE gphistplot.src

SEE ALSO **gpMakeHistPlot**, **gpPlot**

gpHistFPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a HistF plot (frequency histogram).

LIBRARY gaussplot

FORMAT *gp* = **gpHistFPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a HistF plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeHistFPlot** or **gpPlot**. Calling **gpHistFPlotCreate** sets the plot type of the entire graph to HistF. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpHistFPlotCreate** contains one frame by default.

EXAMPLE
```
library gaussplot;
#include gp.sdf
```

```
f = { 10, 5, 0, 35, 11, 18, 55, 20 };
c = { 10, 20, 30, 40, 50, 60, 70, 80 };
string vnames = { "Cat", "Freq" };

struct gpPlotControl gp;
gp = gpHistFPlotCreate;
{ ret } = gpMakeHistFPlot(&gp,c,f,vnames);
```

This example illustrates how you may create a frequency histogram with one line map, using defaults for all style and display options. **gpMakeHistFPlot** associates **c** with the X-axis and **f** with the Y-axis.

SOURCE gphistplot.src

SEE ALSO **gpMakeHistFPlot**, **gpPlot**

gpHistPPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a HistP plot (percentage histogram).

LIBRARY gaussplot

FORMAT *gp* = **gpHistPPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a HistP plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeHistPPlot** or **gpPlot**. Calling **gpHistPPlotCreate** sets the plot type of the entire graph to HistP. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpHistPPlotCreate** contains one frame by default.

EXAMPLE

```
library gaussplot;
#include gp.sdf

x = round(rndn(4000,1)*100);
string vnames = { "Cat", "Prcnt" };

struct gpPlotControl gp;
gp = gpHistPPlotCreate;
{ b,m,freqct,freqp,ret } = gpMakeHistPPlot(&gp,x,50,vnames);
```

This example illustrates how you may create a percentage histogram with one line map, using defaults for all style and display options. **gpMakeHistPPlot** breaks up **x** into 50 evenly spaced categories and calculates percentages of the frequencies for each category. It then associates the categories with the X-axis and the percentages with the Y-axis.

SOURCE gphistplot.src

SEE ALSO **gpMakeHistPPlot**, **gpPlot**

gpInitPlotData

PURPOSE Initializes a **GAUSSplot** data structure and writes specified variable names to it.

LIBRARY gaussplot

FORMAT *gdat* **= gpInitPlotData(***vnames***);**

INPUT *vnames* N×1 string array of variable names for the data.

OUTPUT *gdat* a **GAUSSplot** data structure containing *vnames*.

REMARKS This command creates a **GAUSSplot** data structure, which contains variable names, but no data. Use **gpAddZone** to add data to the structure. Once the data has been added, it can be written to a `.plt` data file with **gpWritePlotData**.

EXAMPLE
```
library gaussplot;
#include gp.sdf

struct gpData gdat;
string vnames = { "X1", "X2", "X3" };
gdat = gpInitPlotData(vnames);
```

SOURCE `gpwritedata.src`

SEE ALSO **gpAddZone**, **gpWritePlotData**, **gpSetPlotData**

gpLogLogPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a LogLog plot.

LIBRARY `gaussplot`

FORMAT *gp* **= gpLogLogPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a LogLog plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeLogLogPlot** or **gpPlot**. Calling **gpLogLogPlotCreate** sets the plot type of the entire graph to LogLog. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpLogLogPlotCreate** contains one frame by default.

EXAMPLE

```
library gaussplot;
#include gp.sdf

x = seqa(0.2,0.2,20);
y = log(x);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpLogLogPlotCreate;
ret = gpMakeLogLogPlot(&gp,x,y,vnames);
```

This example illustrates how you may create a LogLog plot with one line map, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, and the second variable (**V2**) will be associated with the Y-axis.

SOURCE gplogplot.src

SEE ALSO **gpMakeLogLogPlot**, **gpPlot**

gpLogXPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a LogX plot.

LIBRARY gaussplot

FORMAT *gp* = **gpLogXPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a LogX plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeLogXPlot** or **gpPlot**. Calling **gpLogXPlotCreate** sets the plot type of the entire graph to LogX. To

create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpLogXPlotCreate** contains one frame by default.

EXAMPLE

```
library gaussplot;
#include gp.sdf

x = seqa(0.2,0.2,20);
y = log(x);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpLogXPlotCreate;
ret = gpMakeLogXPlot(&gp,x,y,vnames);
```

This example illustrates how you may create a LogX plot with one line map, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, and the second variable (**V2**) will be associated with the Y-axis.

SOURCE gplogplot.src

SEE ALSO **gpMakeLogXPlot**, **gpPlot**

gpLogYPlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a LogY plot.

LIBRARY gaussplot

FORMAT *gp* **= gpLogYPlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a LogY plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeLogYPlot** or **gpPlot**. Calling **gpLogYPlotCreate** sets the plot type of the entire graph to LogY. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpLogYPlotCreate** contains one frame by default.

EXAMPLE
```
library gaussplot;
#include gp.sdf

x = seqa(0.2,0.2,20);
y = log(x);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpLogYPlotCreate;
ret = gpMakeLogYPlot(&gp,x,y,vnames);
```

This example illustrates how you may create a LogY plot with one line map, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, and the second variable (**V2**) will be associated with the Y-axis.

SOURCE gplogplot.src

SEE ALSO **gpMakeLogYPlot**, **gpPlot**

gpMake2DCartesianPlot

PURPOSE Writes a data file and draws a graph with one frame containing a 2D Cartesian

plot.

LIBRARY `gaussplot`

FORMAT *ret* = **gpMake2DCartesianPlot**(*pgp*, *x*, *y*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gp2DCartesianPlotCreate**.
x	N×P matrix, the variable to be associated with the X-axis.
y	N×P matrix, the variable to be associated with the Y-axis.
vnames	2×1 string array, variable names.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a 2D Cartesian plot easily, without first writing the plot data to a `.plt` data file. It writes the variables *x* and *y* to a `.plt` data file with one zone, using the names in *vnames*, and then creates a 2D Cartesian plot with that data. If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically. **gpMake2DCartesianPlot** creates a graph with one frame containing only one field zone. For more complicated graphs, use **gpPlot**.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMake2DCartesianPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
```

```
x = seqa(1,5,20);
y = seqa(10,2,20).*log(x');
x = x'.*ones(20,20);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gp2DCartesianPlotCreate;
ret = gpMake2DCartesianPlot(&gp,x,y,vnames);
```

SOURCE gpcartesianplot.src

SEE ALSO **gp2DCartesianPlotCreate**, **gpPlot**

gpMake2DContourPlot

PURPOSE Writes a data file and draws a graph with one frame containing a 2D Contour plot.

LIBRARY gaussplot

FORMAT *ret* = **gpMake2DContourPlot**(*pgp*, *x*, *y*, *cv*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gp2DContourPlotCreate**.
x	N×P matrix, the variable to be associated with the X-axis.
y	N×P matrix, the variable to be associated with the Y-axis.
cv	N×P matrix, the contour variable.
vnames	3×1 string array, variable names.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a 2D Contour plot easily, without first writing the plot data to a `.plt` data file. It writes the variables *x*, *y* and *cv* to a

.plt data file with one zone, using the names in *vnames*, and then creates a 2D Contour plot with that data. If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically. **gpMake2DContourPlot** creates a graph with one frame containing only one field zone. For more complicated graphs, use **gpPlot**.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMake2DContourPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with gp and have .plt and .mcr extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit gauss.cfg to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
p = pi;
x = seqa(0,p/4,15)';
y = seqa(0,p/4,15);

cv = cos(x).*sin(x-y);
x = x.*ones(15,15);
y = y.*ones(15,15);
string vnames = { "V1", "V2", "V3" };

struct gpPlotControl gp;
gp = gp2DContourPlotCreate;
ret = gpMake2DContourPlot(&gp,x,y,cv,vnames);
```

SOURCE gpcontourplot.src

SEE ALSO **gp2DContourPlotCreate**, **gpPlot**

gpMake3DCartesianPlot

PURPOSE Writes a data file and draws a graph with one frame containing a 3D Cartesian plot.

LIBRARY `gaussplot`

FORMAT *ret* = **gpMake3DCartesianPlot**(*pgp*, *x*, *y*, *z*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gp3DCartesianPlotCreate**.
x	N×P matrix, the variable to be associated with the X-axis.
y	N×P matrix, the variable to be associated with the Y-axis.
z	N×P matrix, the variable to be associated with the Z-axis.
vnames	3×1 string array, variable names.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a 3D Cartesian plot easily, without first writing the plot data to a `.plt` data file. It writes the variables *x*, *y*, and *z* to a `.plt` data file with one zone, using the names in *vnames*, and then creates a 3D Cartesian plot with that data. If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically. **gpMake3DCartesianPlot** creates a graph with one frame containing only one field zone. For more complicated graphs, use **gpPlot**.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMake3DCartesianPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in

the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
p = pi;
x = seqa(0,p/4,15)';
y = seqa(0,p/4,15);

z = cos(x).*sin(x-y);
x = x.*ones(15,15);
y = y.*ones(15,15);
string vnames = { "V1", "V2", "V3" };

struct gpPlotControl gp;
gp = gp3DCartesianPlotCreate;
ret = gpMake3DCartesianPlot(&gp,x,y,z,vnames);
```

SOURCE gpcartesianplot.src

SEE ALSO **gp3DCartesianPlotCreate**, **gpPlot**

gpMake3DContourPlot

PURPOSE Writes a data file and draws a graph with one frame containing a 3D Contour plot.

LIBRARY gaussplot

FORMAT *ret* = **gpMake3DContourPlot**(*pgp*, *x*, *y*, *z*, *cv*, *vnames*);

INPUT *pgp* pointer to a plot control structure created with

		gp3DContourPlotCreate.
	x	N×P matrix, the variable to be associated with the X-axis.
	y	N×P matrix, the variable to be associated with the Y-axis.
	z	N×P matrix, the variable to be associated with the Z-axis.
	cv	N×P matrix, the contour variable.
	vnames	4×1 string array, variable names.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a 3D Contour plot easily, without first writing the plot data to a `.plt` data file. It writes the variables *x*, *y*, *z* and *cv* to a `.plt` data file with one zone, using the names in *vnames*, and then creates a 3D Contour plot with that data. If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically. **gpMake3DContourPlot** creates a graph with one frame containing only one field zone. For more complicated graphs, use **gpPlot**.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMake3DContourPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

EXAMPLE
```
library gaussplot;
#include gp.sdf

// Generate data
p = pi;
x = seqa(0,p/4,15)';
y = seqa(0,p/4,15);

z = cos(x).*sin(x-y);
```

```
cv = sin(x).*cos(x-y);
x = x.*ones(15,15);
y = y.*ones(15,15);
string vnames = { "V1", "V2", "V3", "V4" };

struct gpPlotControl gp;
gp = gp3DContourPlotCreate;
ret = gpMake3DContourPlot(&gp,x,y,z,cv,vnames);
```

SOURCE gpcontourplot.src

SEE ALSO **gp3DContourPlotCreate**, **gpPlot**

gpMakeAxesIndependent

PURPOSE Allows the axes of a plot to be sized independently of one another.

LIBRARY gaussplot

FORMAT *ret* **= gpMakeAxesIndependent(***pgp*, *framenum***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the axes of line plots are sized independently, 2D Cartesian and 2D Contour plots are XY-dependent with an XY-ratio of one, and 3D Cartesian and 3D Contour plots are XYZ-dependent with an XY-ratio of one and an XZ-ratio of one. This means that one unit on the X-axis must be the same length in the plot as one unit on the Y-axis, and for 3D plots, one unit on the X-axis must also be the same length as one unit on the Z-axis.

To specify different dependence ratios, call **gpMakeAxesXYDependent** or **gpMakeAxesXYZDependent**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gp3DCartesianPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
ret = gpMakeAxesIndependent(&gp,1);

ret = gpPlot(&gp);
```

SOURCE gp.src

SEE ALSO **gpMakeAxesXYDependent**, **gpMakeAxesXYZDependent**

gpMakeAxesXYDependent

PURPOSE Makes the X and Y axes dependent on each other, using the specified XY-ratio.

LIBRARY gaussplot

FORMAT *ret* = **gpMakeAxesXYDependent**(*pgp*, *framenum*, *xyratio*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
xyratio	scalar, ratio of the length of one unit on the X-axis to the length of one unit on the Y-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

By default, the axes of line plots are sized independently, 2D Cartesian and 2D Contour plots are XY-dependent with an XY-ratio of one, and 3D Cartesian and 3D Contour plots are XYZ-dependent with an XY-ratio of one and an XZ-ratio of one. This means that one unit on the X-axis must be the same length in the plot as one unit on the Y-axis, and for 3D plots, one unit on the X-axis must also be the same length as one unit on the Z-axis.

Changing the XY ratio has an inverse effect on the relative size of the Y-axis in the plot. Thus if you want to decrease the relative size of the Y-axis, then you should increase the XY-ratio. Similarly, to increase the relative size of the Y-axis, decrease the XY-ratio.

Call **gpMakeAxesXYDependent** to change a plot to XY-dependent or specify a different XY-ratio. To specify different XY and XZ ratios for 3D plots, call **gpMakeAxesXYZDependent**. Or to allow the axes to be sized independently, call **gpMakeAxesIndependent**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gp2DCartesianPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
ret = gpMakeAxesXYDependent(&gp,1,10);

ret = gpPlot(&gp);
```

This example sets the XY-ratio for the first frame to 10, meaning that the length of one unit on the X-axis equals 10 times the length of one unit on the Y-axis.

SOURCE

gp.src

SEE ALSO

gpMakeAxesXYZDependent, **gpMakeAxesIndependent**

gpMakeAxesXYZDependent

PURPOSE Makes the X, Y and Z axes dependent on one another, using the specified XY and XZ ratios.

LIBRARY `gaussplot`

FORMAT *ret* = **gpMakeAxesXYZDependent**(*pgp*, *framenum*, *xyratio*, *xzratio*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
xyratio	scalar, ratio of the length of one unit on the X-axis to the length of one unit on the Y-axis.
xzratio	scalar, ratio of the length of one unit on the X-axis to the length of one unit on the Z-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the axes of line plots are sized independently, 2D Cartesian and 2D Contour plots are XY-dependent with an XY-ratio of one, and 3D Cartesian and 3D Contour plots are XYZ-dependent with an XY-ratio of one and an XZ-ratio of one. This means that one unit on the X-axis must be the same length in the plot as one unit on the Y-axis, and for 3D plots, one unit on the X-axis must also be the same length as one unit on the Z-axis.

Changing the XY and XZ ratios has an inverse effect on the relative sizes of the Y and Z axes in the plot. Thus if you want to decrease the relative sizes of the Y and Z axes, then you should increase the XY and XZ ratios. Similarly, to increase the relative sizes of the Y and Z axes, decrease the XY and XZ ratios.

This command applies only to 3D Cartesian and 3D Contour plots; call it to specify different XY and XZ ratios. To change a plot to XY-dependent, call **gpMakeAxesXYDependent**. Or to allow the axes to be sized independently, call **gpMakeAxesIndependent**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gp3DCartesianPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
ret = gpMakeAxesXYZDependent(&gp,1,10,0.5);

ret = gpPlot(&gp);
```

This example sets the XY-ratio for the first frame to 10, and the XZ-ratio to 0.5. This means that the length of one unit on the X-axis equals 10 times the length of one unit on the Y-axis and half the length of one unit on the Z-axis.

SOURCE gp.src

SEE ALSO **gpMakeAxesXYDependent**, **gpMakeAxesIndependent**

gpMakeBarPlot

PURPOSE Writes a data file and draws a graph with one frame containing a Bar plot.

LIBRARY gaussplot

FORMAT *ret* = **gpMakeBarPlot**(*pgp*, *x*, *y*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gpBarPlotCreate**.
x	N×1 vector or N×K matrix, the variable(s) to be associated with the X-axis.
y	N×1 vector or N×K matrix, the variable(s) to be associated with the Y-axis.

vnames M×1 string array or null string, variable names.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a Bar plot easily, without first writing the plot data to a `.plt` data file. It writes the variables contained in *x* and *y* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is an M×1 string array, then M must equal **cols**(*x*)+**cols**(*y*). If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakeBarPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakeBarPlot** plots a graph with one frame, containing one or more line maps. If both *x* and *y* are N×1 vectors, then only one line map will be created. If they are both N×K matrices, then K line maps will be created, and each column in *x* will be plotted against the corresponding column in *y*. If *x* is an N×1 vector and *y* is an N×K matrix, or vice versa, then the N×1 vector will be plotted against each of the columns in the N×K matrix. To create a more complicated graph with multiple frames, use **gpPlot**.

Observations containing missing values will be deleted before the data is written to the data file. If there are missing values, and *x* and/or *y* contain multiple columns, then each column will be written to a separate zone in the data file, since variables contained in the same zone must be equal in size. If the variables are written to multiple zones, then only the first two variable names in *vnames* will be used, since variable names are set once for all zones in the data file. See **gpMakeXYLinePlot** for more information.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
p = pi;
x = seqa(0,p/4,15);
y = sin(x);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpBarPlotCreate;
ret = gpMakeBarPlot(&gp,x,y,vnames);
```

SOURCE gpbarplot.src

SEE ALSO **gpBarPlotCreate**, **gpPlot**

gpMakeBoxPlot

PURPOSE Writes a data file and draws a graph with one frame containing a Box plot (box graph percentile plot).

LIBRARY gaussplot

FORMAT *ret* = **gpMakeBoxPlot**(*pgp*, *x*, *y*);

INPUT

pgp	pointer to a plot control structure created with **gpBoxPlot**.
x	1×P vector or N×P matrix, containing the group numbers corresponding to each column of *y* data. If scalar 0, a sequence from 1 to **cols**(*y*) will be generated automatically for the X-axis.
y	N×P matrix. Each column represents the set of *y* values for an individual percentile box symbol.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a Box plot easily, without first writing the plot data to a `.plt` data file.

If *grp* is N×P, each row of *grp* should be identical.

gpMakeBoxPlot computes percentiles on the data in *y* and draws box symbols to represent those percentiles. It then writes *grp* (or **reshape(***grp***,**N**,**P**)** if *grp* is 1×P) and *y* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is an M×1 string array, then M must equal **cols(***grp***)+cols(***y***)**. If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakeBoxPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakeBoxPlot** plots a graph with one frame, containing percentile box symbols for each column in *y*. By default, the data points (*grp*,*y*) are not displayed. You may call **gpShowSymbolLayer** before **gpMakeBoxPlot** to display symbols at the data points.

To create a more complicated graph with multiple frames, use **gpPlot**.

Each column of *y* corresponds to a different line map. The percentile box symbols and data points that correspond to a particular column of *y* are considered a part of the same line map. Thus the graph will contain as many line maps as there are columns in *y*.

Observations containing missing values will be deleted before the data is written to the data file. If there are missing values, and *x* and/or *y* contain

multiple columns, then each column will be written to a separate zone in the data file, since variables contained in the same zone must be equal in size. If the variables are written to multiple zones, then only the first two variable names in *vnames* will be used, since variable names are set once for all zones in the data file. See **gpMakeXYLinePlot** for more information.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpBoxPlotCreate;

rndseed 12345;
nboxes = 5;
npoints = 7;
x=seqa(1,1,nboxes)';
y=rndn(npoints,nboxes);

call gpMakeBoxPlot(&gp,x,y);
```

This example illustrates how you may create a Box Plot with five line maps, using defaults for all style and display options.

SOURCE gpboxplot.src

SEE ALSO **gpBoxPlotCreate**, **gpPlot**

gpMakeHistPlot

PURPOSE Writes a data file and draws a graph with one frame containing a Hist plot (frequency histogram).

LIBRARY gaussplot

FORMAT { *b*, *m*, *freq*, *ret* } = **gpMakeHistPlot**(*pgp*, *x*, *v*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gpHistPlotCreate**.
x	N×1 vector, vector of data.
v	L×1 vector, the breakpoints to be used to compute the frequencies – or – scalar, the number of categories.
vnames	2×1 string array or null string, variable names.

OUTPUT

b	P×1 vector, the breakpoints used for each category.
m	P×1 vector, the midpoints of each category.
freq	P×1 vector, computed frequency counts.
ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a Hist plot easily, without first writing the plot data to a `.plt` data file.

If *v* is an L×1 vector, a final breakpoint equal to the maximum value of *x* will be added to the vector of breakpoints if the maximum breakpoint value is smaller. If *v* is a scalar, this command generates *v* evenly-spaced categories. The breakpoints of the categories are stored in *b* and the midpoints of the categories in *m*.

Each time an element falls into one of the categories specified in *b*, the corresponding element of *freq* will be incremented by one. The categories are interpreted as follows:

$$
\begin{array}{lcllcl}
freq[1] & = & & x & <= & b[1] \\
freq[2] & = & b[1] < & x & <= & b[2] \\
freq[3] & = & b[2] < & x & <= & b[3] \\
\quad . & & & & & \\
\quad . & & & & & \\
\quad . & & & & & \\
freq[P] & = & b[P-1] < & x & <= & b[P]
\end{array}
$$

After computing the category and frequency vectors, **gpMakeHistPlot** writes *m* and *freq* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is a null string, the names **V1** and **V2** will be used.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakeHistPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakeHistPlot** plots a graph with one frame, containing one line map. To create a more complicated graph with multiple frames, use **gpPlot**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpHistPlotCreate;

x = round(rndn(4000,1)*100);
string vnames = { "Cat", "Freq" };

{ b,m,freq,ret } = gpMakeHistPlot(&gp,x,50,vnames);
```

SOURCE gphistplot.src

SEE ALSO **gpHistPlotCreate**, **gpPlot**

gpMakeHistFPlot

PURPOSE Writes a data file and draws a graph with one frame containing a HistF plot (frequency histogram), given a vector of frequency counts.

LIBRARY `gaussplot`

FORMAT *ret* = **gpMakeHistFPlot**(*pgp*, *c*, *f*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gpHistFPlotCreate**.
c	N×1 vector, numeric labels for categories. If this is a scalar 0, a sequence from 1 to **rows**(*f*) will be created.
f	N×1 vector, vector of frequency counts.
vnames	2×1 string array or null string, variable names.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a HistF plot easily, without first writing the plot data to a `.plt` data file.

gpMakeHistFPlot writes *c* and *f* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is a null string, the names **V1** and **V2** will be used.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakeHistFPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakeHistFPlot** plots a graph with one frame, containing one line map. To create a more complicated graph with multiple frames, use **gpPlot**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpHistFPlotCreate;

f = { 10, 5, 0, 35, 11, 18, 55, 20 };
c = { 10, 20, 30, 40, 50, 60, 70, 80 };
string vnames = { "Cat", "Freq" };

{ b,m,freq,ret } = gpMakeHistFPlot(&gp,x,50,vnames);
```

SOURCE gphistplot.src

SEE ALSO **gpHistFPlotCreate**, **gpPlot**

gpMakeHistPPlot

PURPOSE Writes a data file and draws a graph with one frame containing a HistP plot (percentage histogram).

LIBRARY gaussplot

FORMAT { *b*, *m*, *freqct*, *freqp*, *ret* } = **gpMakeHistPPlot**(*pgp*, *x*, *v*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gpHistPPlotCreate**.
x	N×1 vector, vector of data.
v	L×1 vector, the breakpoints to be used to compute the frequencies – or – scalar, the number of categories.

	vnames	2×1 string array or null string, variable names.
OUTPUT	*b*	P×1 vector, the breakpoints used for each category.
	m	P×1 vector, the midpoints of each category.
	freqct	P×1 vector, computed frequency counts.
	freqp	P×1 vector, computed frequency percentages.
	ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a HistP plot easily, without first writing the plot data to a `.plt` data file.

If *v* is an L×1 vector, a final breakpoint equal to the maximum value of *x* will be added to the vector of breakpoints if the maximum breakpoint value is smaller. If *v* is a scalar, this command generates *v* evenly-spaced categories. The breakpoints of the categories are stored in *b* and the midpoints of the categories in *m*.

Each time an element falls into one of the categories specified in *b*, the corresponding element of *freqct* will be incremented by one. The categories are interpreted as follows:

$$
\begin{array}{lcllcll}
freqct[1] & = & & & x & <= & b[1] \\
freqct[2] & = & b[1] & < & x & <= & b[2] \\
freqct[3] & = & b[2] & < & x & <= & b[3] \\
\quad . & & & & & & \\
\quad . & & & & & & \\
\quad . & & & & & & \\
freqct[P] & = & b[P-1] & < & x & <= & b[P]
\end{array}
$$

The frequency percentages are calculated from the frequency counts and stored in *freqp*.

After computing the category and frequency vectors, **gpMakeHistPPlot** writes *m* and *freqp* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is a null string, the names **V1** and **V2** will be used.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakeHistPPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with gp and have .plt and .mcr extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit gauss.cfg to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakeHistPPlot** plots a graph with one frame, containing one line map. To create a more complicated graph with multiple frames, use **gpPlot**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpHistPPlotCreate;

x = round(rndn(4000,1)*100);
string vnames = { "Cat", "Prcnt" };

{ b,m,freqct,freqp,ret } = gpMakeHistPPlot(&gp,x,50,vnames);
```

SOURCE gphistplot.src

SEE ALSO **gpHistPPlotCreate**, **gpPlot**

gpMakeLogLogPlot

PURPOSE Writes a data file and draws a graph with one frame containing a LogLog plot.

LIBRARY gaussplot

FORMAT *ret* = **gpMakeLogLogPlot**(*pgp*, *x*, *y*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gpLogLogPlot**.
x	N×1 vector or N×K matrix, the variable(s) to be associated with the X-axis.
y	N×1 vector or N×K matrix, the variable(s) to be associated with the Y-axis.
vnames	M×1 string array or null string, variable names.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A LogLog plot is simply an XY Line plot that uses log scaling on the X and Y axes.

This command allows you to create a LogLog plot easily, without first writing the plot data to a `.plt` data file. It writes the variables contained in *x* and *y* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is an M×1 string array, then M must equal **cols**(*x*)+**cols**(*y*). If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakeLogLogPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakeLogLogPlot** plots a graph with one frame, containing one or more line maps. If both *x* and *y* are N×1 vectors, then only one line map will be created. If they are both N×K matrices, then K line maps will be created, and each column in *x* will be plotted against the corresponding column in *y*. If *x* is an N×1 vector and *y* is an N×K matrix, or vice versa, then

the N×1 vector will be plotted against each of the columns in the N×K matrix. To create a more complicated graph with multiple frames, use **gpPlot**.

Observations containing missing values will be deleted before the data is written to the data file. If there are missing values, and *x* and/or *y* contain multiple columns, then each column will be written to a separate zone in the data file, since variables contained in the same zone must be equal in size. If the variables are written to multiple zones, then only the first two variable names in *vnames* will be used, since variable names are set once for all zones in the data file. See **gpMakeXYLinePlot** for more information.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
x = seqa(0.2,0.2,20);
y = log(x)~ln(x);
string vnames = { "V1", "V2", "V3" };

struct gpPlotControl gp;
gp = gpLogLogPlotCreate;
ret = gpMakeLogLogPlot(&gp,x,y,vnames);
```

SOURCE gplogplot.src

SEE ALSO **gpLogLogPlotCreate**, **gpPlot**

gpMakeLogXPlot

PURPOSE Writes a data file and draws a graph with one frame containing a LogX plot.

LIBRARY gaussplot

FORMAT *ret* = **gpMakeLogXPlot**(*pgp*, *x*, *y*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gpLogXPlot**.
x	N×1 vector or N×K matrix, the variable(s) to be associated with the X-axis.
y	N×1 vector or N×K matrix, the variable(s) to be associated with the Y-axis.
vnames	M×1 string array or null string, variable names.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

A LogX plot is simply an XY Line plot that uses log scaling on the X-axis.

This command allows you to create a LogX plot easily, without first writing the plot data to a `.plt` data file. It writes the variables contained in *x* and *y* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is an M×1 string array, then M must equal **cols(***x***)+cols(***y***)**. If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakeLogXPlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakeLogXPlot** plots a graph with one frame, containing one or more line maps. If both *x* and *y* are N×1 vectors, then only one line map will be created. If they are both N×K matrices, then K line maps will be created, and each column in *x* will be plotted against the corresponding column in *y*. If *x* is an N×1 vector and *y* is an N×K matrix, or vice versa, then the N×1 vector will be plotted against each of the columns in the N×K matrix. To create a more complicated graph with multiple frames, use **gpPlot**.

Observations containing missing values will be deleted before the data is

written to the data file. If there are missing values, and *x* and/or *y* contain multiple columns, then each column will be written to a separate zone in the data file, since variables contained in the same zone must be equal in size. If the variables are written to multiple zones, then only the first two variable names in *vnames* will be used, since variable names are set once for all zones in the data file. See **gpMakeXYLinePlot** for more information.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
x = seqa(0.2,0.2,20);
y = log(x)~ln(x);
string vnames = { "V1", "V2", "V3" };

struct gpPlotControl gp;
gp = gpLogXPlotCreate;
ret = gpMakeLogXPlot(&gp,x,y,vnames);
```

SOURCE gplogplot.src

SEE ALSO **gpLogXPlotCreate**, **gpPlot**

gpMakeLogYPlot

PURPOSE Writes a data file and draws a graph with one frame containing a LogY plot.

LIBRARY gaussplot

FORMAT *ret* = **gpMakeLogYPlot**(*pgp*, *x*, *y*, *vnames*);

INPUT *pgp* pointer to a plot control structure created with **gpLogYPlot**.

	x	N×1 vector or N×K matrix, the variable(s) to be associated with the X-axis.
	y	N×1 vector or N×K matrix, the variable(s) to be associated with the Y-axis.
	vnames	M×1 string array or null string, variable names.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A LogY plot is simply an XY Line plot that uses log scaling on the Y-axis.

This command allows you to create a LogY plot easily, without first writing the plot data to a `.plt` data file. It writes the variables contained in *x* and *y* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is an M×1 string array, then M must equal **`cols(`***x***`)+cols(`***y***`)`**. If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically.

If data and macro file names have not been specified by calls to **`gpSetDataFile`** and **`gpSetMacroFile`**, then **`gpMakeLogYPlot`** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **`gpCleanUp`**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **`gpCleanUp`**.

After writing the data file, **`gpMakeLogYPlot`** plots a graph with one frame, containing one or more line maps. If both *x* and *y* are N×1 vectors, then only one line map will be created. If they are both N×K matrices, then K line maps will be created, and each column in *x* will be plotted against the corresponding column in *y*. If *x* is an N×1 vector and *y* is an N×K matrix, or vice versa, then the N×1 vector will be plotted against each of the columns in the N×K matrix. To create a more complicated graph with multiple frames, use **`gpPlot`**.

Observations containing missing values will be deleted before the data is written to the data file. If there are missing values, and *x* and/or *y* contain

multiple columns, then each column will be written to a separate zone in the data file, since variables contained in the same zone must be equal in size. If the variables are written to multiple zones, then only the first two variable names in *vnames* will be used, since variable names are set once for all zones in the data file. See **gpMakeXYLinePlot** for more information.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
x = seqa(0.2,0.2,20);
y = log(x)~ln(x);
string vnames = { "V1", "V2", "V3" };

struct gpPlotControl gp;
gp = gpLogYPlotCreate;
ret = gpMakeLogYPlot(&gp,x,y,vnames);
```

SOURCE gplogplot.src

SEE ALSO **gpLogYPlotCreate**, **gpPlot**

gpMakePolarLinePlot

PURPOSE Writes a data file and draws a graph with one frame containing a Polar Line plot.

LIBRARY gaussplot

FORMAT *ret* = **gpMakePolarLinePlot**(*pgp*, *theta*, *r*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gpPolarLinePlotCreate**.

	theta	N×1 vector or N×K matrix, the variable(s) to be associated with the Theta-axis.
	r	N×1 vector or N×K matrix, the variable(s) to be associated with the R-axis.
	vnames	M×1 string array or null string, variable names.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create a Polar Line plot easily, without first writing the plot data to a `.plt` data file. It writes the variables contained in *theta* and *r* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is an M×1 string array, then M must equal **cols**(*x*)+**cols**(*y*). If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakePolarLinePlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakePolarLinePlot** plots a graph with one frame, containing one or more line maps. If both *theta* and *r* are N×1 vectors, then only one line map will be created. If they are both N×K matrices, then K line maps will be created, and each column in *theta* will be plotted against the corresponding column in *r*. If *theta* is an N×1 vector and *r* is an N×K matrix, or vice versa, then the N×1 vector will be plotted against each of the columns in the N×K matrix. To create a more complicated graph with multiple frames, use **gpPlot**.

Observations containing missing values will be deleted before the data is written to the data file. If there are missing values, and *theta* and/or *r* contain multiple columns, then each column will be written to a separate zone in the

data file, since variables contained in the same zone must be equal in size. If the variables are written to multiple zones, then only the first two variable names in *vnames* will be used, since variable names are set once for all zones in the data file. See **gpMakeXYLinePlot** for more information.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
theta1 = seqa(0,5,37);
theta2 = seqa(270,5,19)|seqa(5,5,18);
r1 = sin(theta1*pi/180);
r2 = cos(theta2*pi/180);
string vnames = { "Theta1", "Theta2", "R1", "R2" };

struct gpPlotControl gp;
gp = gpPolarLinePlotCreate;
ret = gpMakePolarLinePlot(&gp,theta1~theta2,r1~r2,vnames);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpPolarLinePlotCreate**, **gpPlot**

gpMakeVectorsTangent

PURPOSE Projects the vectors in the vector layer onto the 3D field zone surface.

LIBRARY gaussplot

FORMAT *ret* = **gpMakeVectorsTangent**(*pgp*, *framenum*, *fieldzonenum*, *tangent*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.

	fieldzonenum	scalar or N×1 vector, field zone number(s).
	tangent	scalar, one of the following: **0** do not project vectors onto the 3D field zone surface. **1** project vectors onto the 3D field zone surface.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Calling **gpMakeVectorsTangent** with a *tangent* argument set to 1 causes the component of the vectors normal to the 3D field zone surface to be removed, giving you vectors that are tangent to the surface.

By default, the vector layer is turned off. To activate the vector layer in a frame, call **gpShowVectorLayer**. When the vector layer is activated, plain arrow heads are used by default.

Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE
```
framenum = 1;
seton = 1;
fieldzonenum = 2;
tangent = 1;
ret = gpShowVectorLayer(&gp,framenum,seton);
ret = gpSetMakeVectorsTangent(&gp,framenum,fieldzonenum,tangent);
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowVectorLayer**, **gpSetVectorColor**, **gpSetVectorType**, **gpSetVectorVariables**

gpMakeXYLinePlot

PURPOSE Writes a data file and draws a graph with one frame containing an XY Line plot.

gpMakeXYLinePlot

LIBRARY `gaussplot`

FORMAT *ret* = **gpMakeXYLinePlot**(*pgp*, *x*, *y*, *vnames*);

INPUT

pgp	pointer to a plot control structure created with **gpXYLinePlotCreate**.
x	N×1 vector or N×K matrix, the variable(s) to be associated with the X-axis.
y	N×1 vector or N×K matrix, the variable(s) to be associated with the Y-axis.
vnames	M×1 string array or null string, variable names.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to create an XY Line plot easily, without first writing the plot data to a `.plt` data file. It writes the variables contained in *x* and *y* to a `.plt` data file with one zone, using the names in *vnames*. If *vnames* is an M×1 string array, then M must equal **cols**(*x*)+**cols**(*y*). If *vnames* is a null string, the names $V1, V2, \ldots, VM$ (or $V01, V02, \ldots, VM$, if $9 < M < 100$) will be generated automatically.

If data and macro file names have not been specified by calls to **gpSetDataFile** and **gpSetMacroFile**, then **gpMakeXYLinePlot** will generate temporary names for the data and macro files it creates. The temporary data and macro file names will begin with `gp` and have `.plt` and `.mcr` extensions respectively. The files will be placed on the **GAUSS** temporary file path (edit `gauss.cfg` to change this path). To delete these temporary files after running your program, call **gpCleanUp**. The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

After writing the data file, **gpMakeXYLinePlot** plots a graph with one frame, containing one or more line maps. If both *x* and *y* are N×1 vectors, then only one line map will be created. If they are both N×K matrices, then K line maps will be created, and each column in *x* will be plotted against the corresponding column in *y*. If *x* is an N×1 vector and *y* is an N×K matrix, or vice versa, then

the N×1 vector will be plotted against each of the columns in the N×K matrix. To create a more complicated graph with multiple frames, use **gpPlot**.

Observations containing missing values will be deleted before the data is written to the data file. If there are missing values, and *x* and/or *y* contain multiple columns, then each column will be written to a separate zone in the data file, since variables contained in the same zone must be equal in size. If the variables are written to multiple zones, then only the first two variable names in *vnames* will be used, since variable names are set once for all zones in the data file. For example, suppose that the variables for an XY Line plot are as follows:


```
x =
    0
    0.5
    1
    1.5
    2
    2.5
    3
    3.5

y =
    5       .
    7.2     .
    9       4
    3.5    15
    4.7    17
    12     20
    8      29
    13.4    .
```

with variable names set to “X” and “Y”. The generated data file would be laid out as follows:

Zone 1		
Variable	Observations	Data
X	8	x
Y	8	y[.,1]

Zone 2		
Variable	Observations	Data
X	5	x[3:7]
Y	5	y[3:7,2]

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
x = seqa(0.2,0.2,20);
y = log(x)~ln(x);
string vnames = { "V1", "V2", "V3" };

struct gpPlotControl gp;
gp = gpXYLinePlotCreate;
ret = gpMakeXYLinePlot(&gp,x,y,vnames);
```

SOURCE gpxylineplot.src

SEE ALSO **gpXYLinePlotCreate**, **gpPlot**

gpMoveContourLegend

PURPOSE Moves a contour legend to a specified location in the frame.

LIBRARY gaussplot

FORMAT *ret* = **gpMoveContourLegend**(*pgp*, *framenum*, *contourgroupnum*, *xpos*, *ypos*);

INPUT

pgp	pointer to a plot control structure
framenum	scalar, frame number
contourgroupnum	scalar, contour group number
xpos	scalar, horizontal location of the upper right-hand corner of the contour legend as a percentage of the frame width.
ypos	scalar, vertical location of the upper right-hand corner of the contour legend as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpMoveContourLegend** must be preceded by a call to **gpAddContourLegend**.

EXAMPLE

```
framenum = 1;
contourgroupnum = 1;
xpos = 85;
ypos = 80;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpMoveContourLegend(&gp,framenum,contourgroupnum,xpos,
                          ypos);
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendOrientation**, **gpSetContourLegendBoxStyle**

gpMoveFieldZoneLegend

PURPOSE Moves a field zone legend frame to a specified location in the parent frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpMoveFieldZoneLegend**(*pgp*, *framenum*, *xpos*, *ypos*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
xpos	scalar, horizontal location of the upper right-hand corner of the field zone legend frame as a percentage of the parent frame width.
ypos	scalar, vertical location of the upper right-hand corner of the field zone legend frame as a percentage of the parent frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpMoveFieldZoneLegend** must be preceded by a call to **gpAddFieldZoneLegendFrame**.

The parent frame is the frame that the field zone legend describes. The field zone legend frame is an inset inside the parent frame.

EXAMPLE

```
framenum = 1;
xpos = 97;
ypos = 15;

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpMoveFieldZoneLegend(&gp,framenum,xpos,ypos);
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendBoxStyle**, **gpSetFieldZoneLegendLineSpacing**

gpMoveFrame

PURPOSE Moves and resizes a frame in the workspace.

LIBRARY gaussplot

FORMAT *ret* = **gpMoveFrame**(*pgp*, *framenum*, *xpos*, *ypos*, *width*, *height*);

INPUT

pgp	pointer to a plot control structure
framenum	scalar, frame number
xpos	scalar, distance between the left-hand side of the frame and the left-hand edge of the workspace in paper ruler units.
ypos	scalar, distance between the top of the frame and the top edge of the workspace in paper ruler units.
width	scalar, width of the frame in paper ruler units.
height	scalar, height of the frame in paper ruler units.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all frames have a width of 9, a height of 8, and are located 1 unit from the left edge of the workspace, and 0.25 units from the top edge. Multiple frames will be stacked one on top of another unless a call is made to **gpMoveFrame**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;
```

```
ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAddBarFrame(&gp);
gpAddXYLineFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

This example creates a Bar plot frame and an XY Line plot frame, making the latter an inset in the lower left-hand corner of the former.

SOURCE gp.src

SEE ALSO **gpSetFrameName**, **gpSetFrameStyle**

gpMoveLineLegend

PURPOSE Moves a line legend to a specified location in the frame.

LIBRARY gaussplot

FORMAT *ret* = **gpMoveLineLegend**(*pgp*, *framenum*, *xpos*, *ypos*);

INPUT	*pgp*	pointer to a plot control structure.
	framenum	scalar, frame number.
	xpos	scalar, horizontal location of the upper right-hand corner of the line legend as a percentage of the frame width.
	ypos	scalar, vertical location of the upper right-hand corner of the line legend as a percentage of the frame height.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpMoveLineLegend** must be preceded by a call to **gpAddLineLegend**.

EXAMPLE
```
framenum = 1;
xpos = 85;
ypos = 80;

ret = gpAddLineLegend(&gp,framenum);
ret = gpMoveLineLegend(&gp,framenum,xpos,ypos);
```

SOURCE gplinemap.src

SEE ALSO **gpAddLineLegend**, **gpSetLineLegendBoxStyle**, **gpSetLineLegendMargin**, **gpSetLineLegendLineSpacing**

gpOpenLayoutFile

PURPOSE Opens a layout file in **GAUSSplot**.

LIBRARY gaussplot

FORMAT **gpOpenLayoutFile(***layoutfile***,** *datafile***,** *wait***);**

INPUT	*layoutfile*	string, the name of a layout file created in **GAUSSplot**.

datafile string or null string, data file name.

wait scalar, one of the following:

0 do not wait; return to **GAUSS** immediately.

1 wait until the **GAUSSplot** window has been closed to return to **GAUSS**.

REMARKS In **GAUSSplot** it is possible to save all of the information for a graph as a layout file. This allows you to create a graph using **GAUSSplot** commands, modify the graph using the buttons and menu options in the **GAUSSplot** GUI, and then save the information for the modified graph to a file that can be re-opened later.

The layout file will contain the name of the data file(s) used to create the graph. To re-open the layout file with the same data file(s), call **gpOpenLayoutFile**, setting *datafile* to a null string. If *datafile* contains a data file name, the layout file will be opened with the data from the specified file. The new data file must have at least as many zones and variables as the original data file(s).

EXAMPLE
```
wait = 0;
gpOpenLayoutFile("layout.lay","mydata.plt",wait);
```

SOURCE gp.src

gpPlot

PURPOSE Draws a graph.

LIBRARY gaussplot

FORMAT *ret* = **gpPlot**(*pgp*);

INPUT *pgp* pointer to a plot control structure.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS

This command creates a graph using the information contained in the structure that *pgp* is pointing at. You must call **gpSetDataFile** before **gpPlot** to set the data file(s) that will be used in the graph.

If a macro file name has not been specified by a call to **gpSetMacroFile**, then **gpPlot** will generate a temporary name for the macro file it creates. The temporary name will begin with gp and have an .mcr extension. The file will be placed on the **GAUSS** temporary file path (edit gauss.cfg to change this path). To delete this temporary file, as well as the data and structure files used for the graph, call **gpCleanUp** after running you program (see Section 1.8 for information on structure files).

The names of these files are stored in the structure that *pgp* is pointing at, so you must still have the structure in your **GAUSS** workspace when you call **gpCleanUp**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

// Generate data
p = pi;
x = seqa(0,p/4,15);
y1 = sin(x);
y2 = cos(x);
string vnames = { "V1", "V2", "V3" };
framenum = 1;
string linemaps = { "V1" "V2",
                    "V1" "V3" };

struct gpData gdat;
gdat = gpSetPlotData(x~y1~y2,vnames);
ret = gpWritePlotData(&gdat,"mydata.plt");

struct gpPlotControl gp;
gp = gpBarPlotCreate;

ret = gpSetDataFile(&gp,framenum,"mydata.plt");
```

```
ret = gpSetLineMaps(&gp,framenum,linemaps);
ret = gpPlot(&gp);
```

The above example creates a graph with one frame containing two Bar plot line maps. In the first line map, **V1** is associated with the X-axis and **V2** with the Y-axis. In the second line map, **V1** is again associated with the X-axis, and **V3** is associated with the Y-axis.

SOURCE gp.src

SEE ALSO **gpPlotCreate**, **gpOpenLayoutFile**

gpPlotCreate

PURPOSE Creates a plot control structure and initializes its members to generic default values for a graph with frames containing different plot types.

LIBRARY gaussplot

FORMAT *gp* = **gpPlotCreate;**

OUTPUT *gp* plot control structure with its members set to generic default values.

REMARKS Use **gpPlotCreate** to create a plot control structure that does not have a plot type set for the entire graph. Since the graph has no plot type, it contains no frames by default. Therefore, you must add frames to the graph, using one or more of the following commands:

gpAdd2DCartesianFrame
gpAdd2DContourFrame
gpAdd3DCartesianFrame
gpAdd3DContourFrame
gpAddBarFrame
gpAddPolarLineFrame
gpAddXYLineFrame

The frames will be numbered in the order in which they are added, beginning with one.

You must use **gpPlot** to plot a graph initialized with **gpPlotCreate**.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAdd2DContourFrame(&gp);
gpAdd3DContourFrame(&gp);

framenum = 1;
xpos = 0.25;
ypos = 0.25;
width = 10.5;
height = 8;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);

framenum = 2;
xpos = 2;
ypos = 4.6;
width = 3.5;
height = 2.5;
ret = gpMoveFrame(&gp,framenum,xpos,ypos,width,height);
ret = gpPlot(&gp);
```

The above example creates a graph with two frames. The first contains a 2D Contour plot, and the second a 3D Contour plot. The frames are moved and resized so that the second frame is an inset drawn in the lower left-hand corner of the first frame.

This example assumes that `mydata.plt` already exists and contains at least three variables. By default, a 2D Contour frame associates the first variable in the data file with the X-axis, the second variable with the Y-axis, and uses the last variable as the contour variable. Similarly, a 3D Contour frame associates the first three variables with the X, Y and Z axes, respectively, and uses the last variable as the contour variable.

SOURCE `gp.src`

SEE ALSO **gpPlot**

gpPolarLinePlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for a Polar Line plot.

LIBRARY `gaussplot`

FORMAT *gp* = **gpPolarLinePlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for a Polar Line plot.

REMARKS You may use *gp* to plot a graph using either **gpMakePolarLinePlot** or **gpPlot**. Calling **gpPolarLinePlotCreate** sets the plot type of the entire graph to Polar Line. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpPolarLinePlotCreate** contains one frame by default.

EXAMPLE

```
library gaussplot;
#include gp.sdf

theta = seqa(0,5,37);
r = sin(theta*pi/180);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpPolarLinePlotCreate;
ret = gpMakePolarLinePlot(&gp,theta,r,vnames);
```

This example illustrates how you may create a Polar Line plot with one line map, using defaults for all style and display options. The first variable (**V1**) will be associated with the Theta-axis, and the second variable (**V2**) with the R-axis.

SOURCE gppolarlineplot.src

SEE ALSO **gpMakePolarLinePlot**, **gpPlot**

gpPositionGrid

PURPOSE Positions the grid in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpPositionGrid**(*pgp*, *framenum*, *left*, *right*, *bottom*, *top*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.

	left	scalar, position of the left edge of the grid in frame units (0 - 100).
	right	scalar, position of the right edge of the grid in frame units (0 - 100).
	bottom	scalar, position of the bottom edge of the grid in frame units (0 - 100).
	top	scalar, position of the top edge of the grid in frame units (0 - 100).
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The bottom-left corner of the frame is position (0,0) in frame units.

The default position of the grid is as follows:

```
left = 13;
right = 88;
bottom = 11;
top = 88;
```

This command applies ONLY to Bar, XY Line, 2D Cartesian and 2D Contour plots.

EXAMPLE
```
framenum = 1;
left = 20;
right = 88;
bottom = 11;
top = 88;
ret = gpPositionGrid(&gp,framenum,left,right,bottom,top);
```

SOURCE gp.src

SEE ALSO **gpPositionXAxis**, **gpPositionYAxis**

gpPositionXAxis

PURPOSE Positions an X-axis line in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpPositionXAxis**(*pgp*, *framenum*, *axisnum*, *position*, *coord*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
position	scalar, X-axis position.
coord	scalar, position coordinate system, one of the following:

1 position X-axis at *position* on the corresponding Y-axis (the Y-axis with the same *axisnum*) in Y grid coordinates.

2 position X-axis *position* percent of the grid height below the grid minimum.

3 position X-axis *position* percent of the grid height above the grid maximum.

4 position X-axis at *position* on the vertical gridline as a percentage of the grid height.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one.

By default, the first X-axis in a frame is positioned at the grid minimum, and subsequent X axes are positioned at the grid maximum.

This command applies ONLY to Bar, XY Line, 2D Cartesian and 2D Contour plots.

EXAMPLE

```
framenum = 1;
axisnum = 1;
position = 50;
coord = 4;
ret = gpPositionXAxis(&gp,framenum,axisnum,position,coord);
```

This example vertically centers the first X-axis in the grid.

SOURCE gp.src

SEE ALSO **gpPositionGrid**, **gpShowXAxis**, **gpSetXAxisColor**

gpPositionYAxis

PURPOSE Positions a Y-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpPositionYAxis**(*pgp*, *framenum*, *axisnum*, *position*, *coord*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
position	scalar, Y-axis position.
coord	scalar, position coordinate system, one of the following:

1 position Y-axis at *position* on the corresponding X-axis (the X-axis with the same *axisnum*) in X grid coordinates.

2 position Y-axis *position* percent of the grid width to the left of the grid minimum.

3 position Y-axis *position* percent of the grid width to the right of the grid maximum.

4 position Y-axis at *position* on the horizontal gridline as a percentage of the grid width.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one.

By default, the first Y-axis in a frame is positioned at the grid minimum, and subsequent Y axes are positioned at the grid maximum.

This command applies ONLY to Bar, XY Line, 2D Cartesian and 2D Contour plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
position = 0;
coord = 3;
ret = gpPositionYAxis(&gp,framenum,axisnum,position,coord);
```

This example positions the first Y-axis at the grid maximum (on the right edge of the grid).

SOURCE gp.src

SEE ALSO **gpPositionGrid**, **gpShowYAxis**, **gpSetYAxisColor**

PURPOSE Prints the graph.

LIBRARY gaussplot

FORMAT **gpPrint**(*pgp*);

INPUT *pgp* pointer to a plot control structure.

REMARKS Calling **gpPrint** causes the graph to be printed as soon as it has been plotted with **gpPlot** or one of the **gpMake***PlotType***Plot** commands.

On Windows, the graph is printed to the default printer.

On UNIX and Linux, the graph is printed using the print spooler commands specified in the tecplot.cfg file found in the gaussplot subdirectory of your **GAUSS** installation directory. The tecplot.cfg file contains some sample print spooler commands that are commented out. You must set up the print spooler commands before you call **gpPrint** on UNIX or Linux. The syntax for the print spooler commands is as follows:

```
$!PRINTSETUP
  SPOOLER
    {
    print_format = "spooler_cmd"
    }
```

where *spooler_cmd* is the desired spooler command, and *print_format* is one of the following:

print_format	Used for
PSMONOSPOOLCMD	monochrome PostScript print output
PSCOLORSPOOLCMD	color PostScript print output
HPGLSPOOLCMD	HP-GL print output
HPGL2MONOSPOOLCMD	monochrome HP-GL/2 print output
HPGL2COLORSPOOLCMD	color HP-GL/2 print output

Use '@' as a place holder for the file name in your spooler command. For example:

```
$!PRINTSETUP
  SPOOLER
    {
    PSMONOSPOOLCMD = "lpr @; rm @"
    HPGL2COLORSPOOLCMD = "lpr -P hp5550 @; rm @"
    }
```

specifies that the spooler command "lpr @; rm @" is to be used for all monochrome PostScript print output, and the spooler command "lpr -P hp5550 @; rm @" is to be used for all color HP-GL/2 print output.

The default format for print output on UNIX and Linux is PostScript. We recommend that you always use PostScript as your print format. However, if you want to change the print output format to HP-GL or HP-GL/2, you may do so by calling **gpSetPrintFormat**.

By default, the graph is printed in color. To print the graph as a monochrome image, call **gpSetPrintPalette**. If you want to print to a file instead of a printer, call **gpPrintToFile** before calling **gpPrint**.

EXAMPLE

```
monochrome = 1;
ret = gpSetPrintPalette(&gp,monochrome);
gpPrint(&gp);
```

SOURCE gp.src

SEE ALSO **gpSetPrintPalette**, **gpSetPrintFormat**, **gpPrintToFile**, **gpSetPaperOrientation**

gpPrintToFile

PURPOSE Specifies a print output file name or sets print output to go to printer.

LIBRARY gaussplot

FORMAT **gpPrintToFile(***pgp*, *filename*);

INPUT

pgp	pointer to a plot control structure.
filename	string, name of print output file.

REMARKS If *filename* contains a file name, then all subsequent print jobs (specified by the **gpPrint** command) will be sent to that file, instead of a printer, until another call to **gpPrintToFile** is encountered. If *filename* is a null string, all subsequent print jobs will be sent to a printer.

On UNIX/Linux, the type of output file that is created depends on the print output format specified with **gpSetPrintFormat**. By default, the ouptut file is PostScript.

On Windows, printing to a file generates a `.prn` file containing printer commands. To print a `.prn` file created by a call to **gpPrintToFile**, type the following at a command prompt:

```
copy file_name lpt1 /b
```

Calling **gpPrintToFile** with a file name does not cause the graph to be printed to the file. It merely specifies that all subsequent print jobs be sent to the file. The command must be succeeded by a call to **gpPrint**.

By default, **gpPrint** sends output to a printer, not to a file.

EXAMPLE

```
gpSetPrintToFile(&gp,"print.ps");
gpPrint(&gp);
```

SOURCE gp.src

SEE ALSO **gpPrint**, **gpSetPrintFormat**, **gpSetPrintPalette**

gpRedefineColor

PURPOSE Sets the RGB values for a color in the color palette.

LIBRARY `gaussplot`

FORMAT *ret* = **gpRedefineColor**(*pgp*, *color*, *rvalue*, *gvalue*, *bvalue*);

INPUT

pgp	pointer to a plot control structure.
color	string or N×1 string array, name(s) of color(s) to redefine. – or – scalar or N×1 vector, indices of color(s) in color list.
rvalue	scalar or N×1 vector, $0 \leq rvalue \leq 255$, red value(s).
gvalue	scalar or N×1 vector, $0 \leq gvalue \leq 255$, green value(s).
bvalue	scalar or N×1 vector, $0 \leq bvalue \leq 255$, blue value(s).

Output

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS When you redefine a color in the color palette, all instances of that color in the graph will use the new definition. To redefine multiple colors with one call to **gpRedefineColor**, use N×1 vectors for all of the arguments. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
fieldzonenum = 1;
ret = gpRedefineColor(&gp,"custom6",55,127,199);
ret = gpSetMeshColor(&gp,framenum,fieldzonenum,"custom6");
```

SOURCE `gp.src`

gpRotateAxes

PURPOSE Rotates a plot about the X, Y and Z axes.

LIBRARY `gaussplot`

FORMAT *ret* = **gpRotateAxes**(*pgp*, *framenum*, *xrotation*, *yrotation*, *zrotation*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
xrotation	scalar, $-180 \leq xrotation \leq 180$, rotation about the X-axis in degrees.
yrotation	scalar, $-180 \leq yrotation \leq 180$, rotation about the Y-axis in degrees.
zrotation	scalar, $-180 \leq zrotation \leq 180$, rotation about the Z-axis in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is a convenience function that allows you to rotate a plot about the X, Y and Z axes with a single command. If *xrotation*, *yrotation*, or *zrotation* is set to zero, then the plot will not be rotated about the corresponding axis. To rotate the plot about one axis, call **gpRotateXAxis**, **gpRotateYAxis** or **gpRotateZAxis**.

A call to **gpRotateAxes** rotates the plot *xrotation* degrees about the X-axis, *yrotation* degrees about the Y-axis, and *zrotation* degrees about the Z-axis in that order.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
xrotation = 45;
```

```
yrotation = 90;
zrotation = 180;
ret = gpRotateAxes(&gp,framenum,xrotation,yrotation,zrotation);
```

SOURCE gpfieldzone.src

SEE ALSO **gpRotateXAxis**, **gpRotateYAxis**, **gpRotateZAxis**

gpRotateTextObject

PURPOSE Rotates a text object from the horizontal.

LIBRARY gaussplot

FORMAT *ret* = **gpRotateTextObject**(*pgp*, *framenum*, *name*, *angle*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
angle	scalar, $-360 \leq angle \leq 360$, angle in degrees to rotate the text object from the horizontal.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpRotateTextObject** must be preceded by a call to **gpAddTextObject**. By default, the angle is set to 0, i.e., the text object is drawn horizontally.

Inputting a negative value for *angle* rotates the text object clockwise, while inputting a positive value rotates the text object counter-clockwise.

EXAMPLE

```
framenum = 1;
```

```
objectname = "text1";
angle = 15;

ret = gpAddTextObject(&gp,framenum,objectname,"f(x) = sin(x)");
ret = gpRotateTextObject(&gp,framenum,objectname,angle);
```

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectBoxStyle**

gpRotateXAxis

PURPOSE Rotates a plot about the X-axis.

LIBRARY gaussplot

FORMAT *ret* = **gpRotateXAxis**(*pgp*, *framenum*, *xrotation*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
xrotation	scalar, $-180 \leq xrotation \leq 180$, rotation about the X-axis in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpRotateXAxis** rotates the plot *xrotation* degrees about the X-axis. To rotate the plot about the X, Y and Z axes, call **gpRotateAxes**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
xrotation = -45;
ret = gpRotateXAxis(&gp,framenum,xrotation);
```

SOURCE gpfieldzone.src

SEE ALSO **gpRotateAxes**

gpRotateYAxis

PURPOSE Rotates a plot about the Y-axis.

LIBRARY gaussplot

FORMAT *ret* = **gpRotateYAxis**(*pgp*, *framenum*, *yrotation*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
yrotation	scalar, $-180 \leq yrotation \leq 180$, rotation about the Y-axis in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpRotateYAxis** rotates the plot *yrotation* degrees about the Y-axis. To rotate the plot about the X, Y and Z axes, call **gpRotateAxes**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
yrotation = 130;
ret = gpRotateYAxis(&gp,framenum,yrotation);
```

SOURCE gpfieldzone.src

SEE ALSO **gpRotateAxes**

gpRotateZAxis

PURPOSE Rotates a plot about the Z-axis.

LIBRARY `gaussplot`

FORMAT *ret* **= gpRotateZAxis(***pgp***,** *framenum***,** *zrotation***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
zrotation	scalar, $-180 \leq zrotation \leq 180$, rotation about the Z-axis in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpRotateZAxis** rotates the plot *zrotation* degrees about the Z-axis. To rotate the plot about the X, Y and Z axes, call **gpRotateAxes**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
zrotation = -75;
ret = gpRotateZAxis(&gp,framenum,zrotation);
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpRotateAxes**

gpRunMacroFile

PURPOSE Runs a macro file in **GAUSSplot**.

LIBRARY `gaussplot`

FORMAT **gpRunMacroFile(***macrofile***,***wait***);**

INPUT

macrofile string, the name of a macro file created with **GAUSSplot**.

wait scalar, one of the following:

- **0** do not wait; return to **GAUSS** immediately.
- **1** wait until the **GAUSSplot** window has been closed to return to **GAUSS**.

REMARKS When you call **gpPlot** or any of the other plotting commands (see Section 2.1.16) to plot a graph based on the information in a plot control structure, **GAUSSplot** writes the information out as a macro file and then runs the macro file to produce the graph. Call this command to run a macro file that was previously created by a **GAUSSplot** plotting command.

The plotting commands generate a temporary file name for the macro file if a name is not specified with **gpSetMacroFile**.

EXAMPLE
```
gpRunMacroFile("macro.mcr");
```

SOURCE `gp.src`

SEE ALSO **gpAddMacroFile**, **gpSetMacroFile**, **gpOpenLayoutFile**

gpSet3DViewDistance

PURPOSE Sets the distance between the viewer and the center point of a plot.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSet3DViewDistance**(*pgp*, *framenum*, *distance*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
distance	scalar, distance between the viewer and the center point of a plot in 3D grid coordinates.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS If you call **gpSet3DViewDistance** to specify the distance between the viewer and the center point of a plot, **GAUSSplot** will choose an (X,Y,Z) coordinate position for the viewer that satisfies *distance* and maintains the current perspective as much as possible.

If 3D perspective is enabled (see **gpUse3DPerspective**), then increasing *distance* will decrease the size of the plot in the frame, and decreasing *distance* will increase the size of the plot in the frame. By default, 3D perspective is not used.

To specify an exact location for the viewer in 3D grid coordinates, call **gpSet3DViewerPosition**.

If you call **gpSet3DViewerPosition** or **gpSetPlotFit** after **gpSet3DViewDistance**, then *distance* will be overridden.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
```

```
distance = 35;
 ret = gpSet3DViewDistance(&gp,framenum,distance);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSet3DViewerPosition**, **gpUse3DPerspective**

gpSet3DViewerPosition

PURPOSE Sets the position of the viewer in 3D grid coordinates.

LIBRARY gaussplot

FORMAT *ret* = **gpSet3DViewerPosition**(*pgp*, *framenum*, *xpos*, *ypos*, *zpos*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
xpos	scalar, X-coordinate position of the viewer.
ypos	scalar, Y-coordinate position of the viewer.
zpos	scalar, Z-coordinate position of the viewer.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS To specify a distance between the viewer and the center point of a plot and allow **GAUSSplot** to calculate an appropriate (X,Y,Z) coordinate position for the viewer, call **gpSet3DViewDistance**. If you call **gpSet3DViewDistance** after **gpSet3DViewerPosition**, then the former command will reset the distance between the viewer and the center point of the plot, while maintaining, as much as possible, the perspective set by **gpSet3DViewerPosition**.

If you call **gpSet3DViewerPosition** and one or more of the rotation commands (**gpRotateXAxis**, **gpRotateYAxis**, **gpRotateZAxis** or

gpRotateAxes), then the operations will be performed in the order in which you call them. For example, if you call **gpSet3DViewerPosition** and then call **gpRotateXAxis**, then the specified viewer position will be set before the rotation and will be changed by the rotation. However, if you call **gpRotateXAxis** and then **gpSet3DViewerPosition**, then the specified viewer position will be set after the rotation.

If you call **gpSet3DViewerPosition** and one or more of the rotation animation commands (**gpAnimateXAxisRotation**, **gpAnimateYAxisRotation**, **gpAnimateZAxisRotation**, or **gpAnimate3DRotation**), the viewer position will be set before the animation, regardless of the order in which the commands are called.

Calling **gpSetPlotFit** after **gpSet3DViewerPosition** will override the specified viewer position.

Be careful not to specify a position for the viewer that will place the entire plot outside of the viewer's field of view.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
xpos = 15;
ypos = 15;
zpos = 18;
ret = gpSet3DViewerPosition(&gp,framenum,xpos,ypos,zpos);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSet3DViewDistance**, **gpUse3DPerspective**

gpSetActiveFieldZones

PURPOSE Specifies which field zones to display in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetActiveFieldZones**(*pgp*, *framenum*, *fieldzonenum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, number(s) of the field zone(s) to be displayed.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command does not apply to Bar, XY Line or Polar Line plots. For all other plot types, the default is to display all field zones in a frame unless a call to **gpAnimateFieldZones** is encountered. If **gpAnimateFieldZones** is called in the program, then only the first field zone is displayed by default.

EXAMPLE

```
library gaussplot;
#include gp.sdf

struct gpPlotControl gp;
gp = gpPlotCreate;

ret = gpSetDataFile(&gp,0,"mydata.plt");
gpAdd3DCartesianFrame(&gp);
gpAdd2DContourFrame(&gp);

ret = gpSetActiveFieldZones(&gp,1,1);
ret = gpSetActiveFieldZones(&gp,2,2|3);

ret = gpPlot(&gp);
```

This example activates the first field zone in frame one and the second and third field zones in frame two.

SOURCE gpfieldzone.src

SEE ALSO **gpSetAxisMap**

gpSetArrowColor

PURPOSE Sets the color of an arrow.

LIBRARY gaussplot

FORMAT *ret* = **gpSetArrowColor**(*pgp*, *framenum*, *name*, *color*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of an arrow object created by **gpAddArrow**.
color	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
ret = gpAddArrow(&gp,framenum,"arrow1",xstart,ystart,xend,
                 yend,coord);
ret = gpSetArrowColor(&gp,framenum,"arrow1","green");
```

This example adds an arrow to frame one, drawing the arrow from the origin to point (1.5, 1). It then sets the color of the arrow to green.

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowHeadStyle**, **gpSetArrowLineThickness**

gpSetArrowHeadAngle

PURPOSE Sets the angle that one side of an arrow head makes with the arrow line segment.

LIBRARY gaussplot

FORMAT *ret* = **gpSetArrowHeadAngle**(*pgp*, *framenum*, *name*, *angle*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of an arrow object created by **gpAddArrow**.
angle	scalar, angle that one side of the arrow head makes with the line segment in degrees.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the arrow head angle is set to 12.

EXAMPLE

```
framenum = 1;
xstart = 0;
xend = 0.5;
ystart = 0;
yend = 1.25;
coord = 2;
aname = "arrow1";
angle = 20;

ret = gpAddArrow(&gp,framenum,aname,xstart,ystart,xend,yend,
```

```
                        coord);
ret = gpSetArrowHeadAngle(&gp,framenum,aname,angle);
```

This example adds an arrow to frame one and then sets the angle that one side of the arrow head makes with the arrow line segment to 20 degrees.

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowHeadStyle**

gpSetArrowHeadAttachment

PURPOSE Specifies to which end of an arrow the arrow head is attached.

LIBRARY gaussplot

FORMAT *ret* = **gpSetArrowHeadAttachment**(*pgp*, *framenum*, *name*, *attachment*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of an arrow object created by **gpAddArrow**.
attachment	scalar, one of the following:

1 arrow head attaches to the beginning of the line segment.
2 arrow head attaches to the end of the line segment.
3 arrow heads attach to both ends.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, arrows are drawn with an arrow head attached to the end of the line segment.

EXAMPLE

```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
aname = "arrow1";
attachment = 1;

ret = gpAddArrow(&gp,framenum,aname,xstart,ystart,xend,yend,
                 coord);
ret = gpSetArrowHeadAttachment(&gp,framenum,aname,attachment);
```

This example adds an arrow to frame one and then sets the arrow head attachment to the beginning of the line segment.

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowHeadStyle**

gpSetArrowHeadSize

PURPOSE Sets the size of an arrow head.

LIBRARY gaussplot

FORMAT *ret* = **gpSetArrowHeadSize**(*pgp*, *framenum name size*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of an arrow object created by **gpAddArrow**.
size	scalar, size of the arrow head as a percentage of the frame height.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, arrow heads have a size of 5.

EXAMPLE
```
framenum = 2;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
aname = "arrow1";
size = 7;

ret = gpAddArrow(&gp,framenum,aname,xstart,ystart,xend,yend,coord);
ret = gpSetArrowHeadSize(&gp,framenum,aname,size);
```

This example adds an arrow to frame two and then sets the size of the arrow head to 7% of the frame height.

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowHeadStyle**

gpSetArrowHeadStyle

PURPOSE Sets the style of an arrow head.

LIBRARY gaussplot

FORMAT *ret* =
gpSetArrowHeadStyle(*pgp*,*framenum*,*name*,*type*,*attachment*,*size*,*angle***);**

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	name	string, name of an arrow object created by **gpAddArrow**.
	type	scalar, arrow head type, one of the following: **1** plain arrow head. **2** filled arrow head. **3** hollow arrow head.
	attachment	scalar, arrow head attachment point, one of the following: **1** arrow head attaches to the beginning of the line segment. **2** arrow head attaches to the end of the line segment. **3** arrow heads attach to both ends.
	size	scalar, size of the arrow head as a percentage of the frame height.
	angle	scalar, angle in degrees that one side of the arrow head makes with the line segment.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is a convenience function, which allows you to set several arrow head style parameters with a single function call.

By default, arrows are drawn with plain arrow heads, which are attached to the end of the line segment. An arrow head has a default size of 5 and a default angle of 12.

EXAMPLE
```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
aname = "arrow1";

type = 1;
attachment = 1;
```

```
size = 7;
angle = 20;

ret = gpAddArrow(&gp,framenum,aname,xstart,ystart,xend,yend,
                 coord);
ret = gpSetArrowHeadStyle(&gp,framenum,aname,type,attachment,
                          size,angle);
```

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowHeadAngle**, **gpSetArrowHeadAttachment**, **gpSetArrowHeadSize**, **gpSetArrowHeadType**

gpSetArrowHeadType

PURPOSE Sets the type of an arrow head.

LIBRARY gaussplot

FORMAT *ret* = **gpSetArrowHeadType**(*pgp*, *framenum*, *name*, *type*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of an arrow object created by **gpAddArrow**.
type	scalar, arrow head type, one of the following:

1 plain arrow head.
2 filled arrow head.
3 hollow arrow head.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, arrows are drawn with plain arrow heads.

EXAMPLE

```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
aname = "arrow1";
type = 2;

ret = gpAddArrow(&gp,framenum,aname,xstart,ystart,xend,yend,
                 coord);
ret = gpSetArrowHeadType(&gp,framenum,aname,type);
```

This example adds an arrow to frame one and then sets the arrow head type to filled.

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowHeadStyle**

gpSetArrowLinePattern

PURPOSE Sets the pattern of the arrow line segment.

LIBRARY gaussplot

FORMAT *ret* = **gpSetArrowLinePattern**(*pgp*, *framenum*, *name*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of an arrow object created by **gpAddArrow**.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all arrows are drawn with solid line segments. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
aname = "arrow1";

ret = gpAddArrow(&gp,framenum,aname,xstart,ystart,xend,yend,
                 coord);
ret = gpSetArrowLinePattern(&gp,framenum,aname,"dashed");
```

This example adds an arrow to frame one and then sets the pattern of the arrow line segment to ‘dashed’.

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowLinePatternLength**, **gpSetArrowColor**

gpSetArrowLinePatternLength

PURPOSE Sets the length of the arrow line segment pattern.

LIBRARY gaussplot

FORMAT *ret* = **gpSetArrowLinePatternLength**(*pgp*, *framenum*, *name*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of an arrow object created by **gpAddArrow**.
length	scalar, length of one cycle of the line pattern as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is ignored if the line pattern is set to solid. For non-solid lines, the default pattern length is 2.

EXAMPLE

```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
aname = "arrow1";
length = 3;

ret = gpAddArrow(&gp,framenum,aname,xstart,ystart,xend,yend,coord);
ret = gpSetArrowLinePattern(&gp,framenum,aname,"dashed");
ret = gpSetArrowLinePatternLength(&gp,framenum,aname,length);
```

This example adds an arrow to frame one, sets the pattern of the arrow line segment to ‘dashed’, and then sets the length of one cycle of the pattern to 3% of the frame height.

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowLinePattern**, **gpSetArrowColor**

gpSetArrowLineThickness

PURPOSE Sets the thickness of the arrow line segment.

LIBRARY gaussplot

FORMAT *ret* = **gpSetArrowLineThickness**(*pgp*, *framenum*, *name*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of an arrow object created by **gpAddArrow**.
thickness	scalar, thickness of the arrow line segment as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, arrow line segments have a thickness of 0.1.

EXAMPLE

```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
aname = "arrow1";
thickness = 0.4;

ret = gpAddArrow(&gp,framenum,aname,xstart,ystart,xend,yend,
                 coord);
ret = gpSetArrowLineThickness(&gp,framenum,aname,thickness);
```

This example adds an arrow to frame one and then sets the thickness of the arrow line segment to 0.4.

SOURCE gparrow.src

SEE ALSO **gpAddArrow**, **gpSetArrowColor**

gpSetAxisMap

PURPOSE Sets the variable to be associated with each axis in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetAxisMap**(*pgp*, *framenum*, *vars*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
vars	K×1 vector, indices of the variables in the data file. – or – K×1 string array, names of the variables.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS If the specified frame contains a 2D Cartesian or 2D Contour plot, then *vars* must be a 2× 1 vector or string array, with the first element corresponding to the X-axis and the second element to the Y-axis. If the specified frame contains a 3D Cartesian or 3D Contour plot, then *vars* must be a 3×1 vector or string array, with the first element corresponding to the X-axis, the second element to the Y-axis, and the third element to the Z-axis.

By default, the first variable in the data file is associated with the X-axis, the second variable with the Y-axis, and (for 3D plots) the third variable with the Z-axis.

EXAMPLE

```
framenum = 1;
vars = { 1, 4, 5 };
ret = gpSetAxisMap(&gp,framenum,vars);
```

This example associates the first variable in the data file with the X-axis, the fourth variable with the Y-axis, and the fifth variable with the Z-axis.

SOURCE gp.src

SEE ALSO **gpSetActiveFieldZones**

gpSetBarColor

PURPOSE Sets the color of the bars in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetBarColor**(*pgp*, *framenum*, *linemapnum*, *linecolor*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
linecolor	string, name of bar outline color. – or – scalar, index of color in color list.
fillcolor	string, name of bar fill color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
linemapnum = 2;
ret = gpSetBarColor(&gp,framenum,linemapnum,"custom25",
                    "green");
```

SOURCE gpbarplot.src

SEE ALSO **gpBarPlotCreate**, **gpAddBarFrame**, **gpSetBarWidth**

gpSetBarLineThickness

PURPOSE Sets the line thickness of the bar outlines in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetBarLineThickness**(*pgp*, *framenum*, *linemapnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
thickness	scalar, line thickness of the bar outlines as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all bars outlines are set to a line thickness of 0.4.

EXAMPLE
```
framenum = 3;
linemapnum = 2;
thickness = 0.6;
ret = gpSetBarLineThickness(&gp,framenum,linemapnum,thickness);
```

SOURCE gpbarplot.src

SEE ALSO **gpSetBarColor**, **gpSetBarOrientation**, **gpSetBarWidth**

gpSetBarOrientation

PURPOSE Sets the orientation of the bars in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetBarOrientation**(*pgp*, *framenum*, *linemapnum*, *orientation*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
orientation	scalar, one of the following: **1** bars drawn vertically. **2** bars drawn horizontally.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, bars are drawn vertically.

EXAMPLE

```
framenum = 2;
linemapnum = 1;
orientation = 2;
ret = gpSetBarOrientation(&gp,framenum,linemapnum,orientation);
```

SOURCE gpbarplot.src

SEE ALSO **gpSetBarColor**, **gpSetBarLineThickness**, **gpSetBarWidth**

gpSetBarWidth

PURPOSE Sets the width of the bars in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetBarWidth**(*pgp*, *framenum*, *linemapnum*, *width*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
width	scalar, width of the bars as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all bars are set to a width of 2.5.

EXAMPLE
```
framenum = 3;
linemapnum = 2;
width = 1;
ret = gpSetBarWidth(&gp,framenum,linemapnum,width);
```

SOURCE gpbarplot.src

SEE ALSO **gpSetBarColor**, **gpSetBarLineThickness**, **gpSetBarOrientation**

gpSetBoundaryColor

PURPOSE Sets the color of the boundary line around the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetBoundaryColor**(*pgp*, *framenum*, *fieldzonenum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
linecolor	string, name of boundary line color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Field zone numbers correspond to the number of the associated zone in the data file. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
fieldzonenum = 2;
ret = gpSetBoundaryColor(&gp,framenum,fieldzonenum,"purple");
```

SOURCE `gp.src`

SEE ALSO **gpSetBoundaryThickness**, **gpShowBoundaryLayer**, **gpShowFieldZoneBoundaryLayer**

gpSetBoundaryThickness

PURPOSE Sets the line thickness of the boundary around the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetBoundaryThickness**(*pgp*, *framenum*, *fieldzonenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
thickness	scalar, thickness of the boundary line as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all boundary lines have a thickness of 0.4. Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE
```
framenum = 3;
fieldzonenum = 2;
thickness = 0.6;
ret = gpSetBoundaryThickness(&gp,framenum,fieldzonenum,
                                thickness);
```

SOURCE gp.src

SEE ALSO **gpSetBoundaryColor**, **gpShowBoundaryLayer**, **gpShowFieldZoneBoundaryLayer**

gpSetBoxColor

PURPOSE Sets the color of the percentile box symbol for the specified line map(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetBoxColor**(*pgp*, *framenum*, *linemapnum*, *boxcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
boxcolor	string, name of line color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

This command operates only on frames containing Box plots.

EXAMPLE
```
framenum = 1;
linemapnum = 3;
ret = gpSetBoxColor(&gp,framenum,linemapnum,"red");
```

SOURCE `gpboxplot.src`

SEE ALSO **gpBoxPlotCreate**, **gpAddBoxFrame**, **gpSetBoxWidth**

gpSetBoxMinMaxMethod

PURPOSE Sets the method by which the maximum and minimum values for the percentile box symbols in the specified frame are calculated.

LIBRARY `gausssplot`

FORMAT *ret* = **gpSetBoxMinMaxMethod**(*pgp*, *framenum*, *minmaxmethod*);

INPUT

pgp pointer to a plot control structure.

framenum scalar, frame number.

minmaxmethod scalar or 3×1 vector, where the first element is one of the following:

1 Minimum and maximum taken from the actual limits of the data.

2 Statistical standard with the minimum and maximum calculated according to interquartile range as follows:

$$\begin{aligned} intqrange &= 75^{th} - 25^{th} \\ min &= 25^{th} - 1.5 intqrange \\ max &= 75^{th} + 1.5 intqrange \end{aligned}$$

3 Minimum and maximum percentiles taken from the second and third elements of *minmaxmethod*, respectively.

and the second and third elements are optional minimum and maximum percentiles, used only if *minmaxmethod*[1] = 3.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command operates only on frames containing Box plots, and must be called BEFORE adding the frame with **gpAddBoxFrame** when creating a multiple-frame plot.

By default, minimum and maximum values are taken from the actual limits of the data.

EXAMPLE
```
framenum = 1;
minmaxmethod = { 3, 10, 90 };
ret = gpSetBoxMinMaxMethod(&gp,framenum,minmaxmethod);
```

SOURCE gpboxplot.src

SEE ALSO **gpBoxPlotCreate**, **gpAddBoxFrame**, **gpSetBoxColor**, **gpSetBoxWidth**

gpSetBoxWidth

PURPOSE Sets the width of the percentile box symbols for the line maps in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetBoxWidth**(*pgp*, *framenum*, *boxwidth*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
boxwidth	scalar, width of percentile box symbols in X-axis grid units.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command operates only on frames containing Box plots, and must be called BEFORE adding the frame with **gpAddBoxFrame** when creating a multiple-frame plot.

By default, box widths are set to one half of the average horizontal distance between percentile box symbols in the frame.

If *boxwidth* is set to 0, the box plot is drawn as two vertical lines representing the quartile ranges with a filled circle representing the 50^{th} percentile.

EXAMPLE
```
framenum = 1;
boxwidth = 0;
ret = gpSetBoxWidth(&gp,framenum,boxwidth);
```

SOURCE gpboxplot.src

SEE ALSO **gpBoxPlotCreate**, **gpAddBoxFrame**, **gpSetBoxColor**, **gpSetBoxMinMaxMethod**

gpSetContourColorDistribution

PURPOSE Sets the color map distribution of contour flooding for the specified contour group.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourColorDistribution(***pgp*, *framenum*, *contourgroupnum*, *distribution***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar or N×1 vector, contour group number(s).
distribution	scalar, one of the following:
	1 banded distribution.
	2 continuous distribution.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, contour groups use banded color distribution, which uses a solid color between contour levels. Changing the color distribution for a contour group to continuous causes the shade of a color used between contour levels to vary depending on distance from the contour level.

All contour options are set within a contour group. Each field zone that is displayed as a contour plot must be associated with a particular contour group, which controls the contour options. Contour plots can be displayed with contour lines and/or contour flooding. If both lines and flooding are used in a field zone, they may be associated with different contour groups. By default a contour plot is displayed with contour flooding only, using contour group one to control all options.

EXAMPLE

```
framenum = 1;
contourgroupnum = 1;
distribution = 2;

ret = gpSetContourColorDistribution(&gp,framenum,
                                    contourgroupnum,distribution);
```

This example sets the color map distribution of contour flooding for contour group one to continuous.

SOURCE gpcontourplot.src

SEE ALSO **gpSetContourFloodGroup**, **gpSetContourVariable**

gpSetContourFloodGroup

PURPOSE Sets the contour group to be used for contour flooding in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourFloodGroup**(*pgp*, *framenum*, *fieldzonenum*, *contourgroupnum*);

INPUT *pgp* pointer to a plot control structure.

framenum scalar, frame number.

fieldzonenum scalar or N×1 vector, field zone number(s).

contourgroupnum scalar, contour group number.

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

All contour options are set within a contour group. Each field zone that is displayed as a contour plot must be associated with a particular contour group, which controls the contour options. Contour plots can be displayed with contour lines and/or contour flooding. If both lines and flooding are used in a field zone, they may be associated with different contour groups.

By default, a contour plot is displayed with contour flooding only, using contour group one to control all options.

EXAMPLE

```
framenum = 1;
contourgroupnum = 2;
variableindex = 4;
fieldzonenum = 2;

ret = gpSetContourVariable(&gp,framenum,contourgroupnum,variableindex)
ret = gpSetContourFloodGroup(&gp,framenum,fieldzonenum,contourgroupnum
```

This example sets the contour variable in contour group two to the fourth variable in the data file. It then sets the contour group to be used for contour flooding in field zone two to contour group two.

SOURCE

gpcontourplot.src

SEE ALSO

gpSetContourLineGroup, **gpSetContourVariable**, **gpSetContourColorDistribution**

gpSetContourLegendBoxColor

PURPOSE Sets the outline and background colors of the box surrounding a contour legend.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetContourLegendBoxColor**(*pgp*, *framenum*, *contourgroupnum*, *linecolor*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar, contour group number.
linecolor	string, name of color to use for box outline. – or – scalar, index of color in color list.
fillcolor	string, name of color to use for box background. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendBoxColor** must be preceded by a call to **gpAddContourLegend**. If the contour legend box type is set to none, then these colors are not used. If the contour legend box type is set to hollow, then fillcolor is not used, since a hollow box has no background. By default, a contour legend has a hollow box, which is surrounded by a black line. To change the box type, call **gpSetContourLegendBoxType**.

See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
contourgroupnum = 1;
```

```
linecolor = "red";
fillcolor = "";

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendBoxColor(&gp,framenum,
                                    contourgroupnum,linecolor,fillcolor
```

In this example, *fillcolor* is set to a null string because the box type is hollow and thus has no background.

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendBoxStyle**

gpSetContourLegendBoxStyle

PURPOSE Sets the style of the box surrounding a contour legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLegendBoxStyle**(*pgp*, *framenum*, *contourgroupnum*, *boxtype*, *thickness*, *linecolor*, *fillcolor*);

INPUT

pgp pointer to a plot control structure.

framenum scalar, frame number.

contourgroupnum scalar, contour group number.

boxtype scalar, one of the following:

1 no box.

2 hollow box (no background).

3 filled box.

thickness scalar, thickness of the box outline as a percentage of the frame height.

	linecolor	string, name of color to use for box outline. – or – scalar, index of color in color list.
	fillcolor	string, name of color to use for box background. – or – scalar, index of color in color list.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

This command is a convenience function that allows you to set several parameters at one time. See **gpSetContourLegendBoxType**, **gpSetContourLegendBoxThickness**, and **gpSetContourLegendBoxColor** to set the box style parameters individually.

A call to **gpSetContourLegendBoxStyle** must be preceded by a call to **gpAddContourLegend**. By default, a contour legend has a hollow box (i.e., a box with no background), which is surrounded by a black line that is 0.1% of the frame height. If *boxtype* is specified as hollow, then *fillcolor* is ignored.

See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
contourgroupnum = 1;
boxtype = 3;
thickness = 0.4;
linecolor = "red";
fillcolor = "white";

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendBoxStyle(&gp,framenum,contourgroupnum,
                                 boxtype,thickness,linecolor,fillcol
```

SOURCE

gpcontourplot.src

SEE ALSO

gpAddContourLegend, **gpSetContourLegendBoxColor**, **gpSetContourLegendBoxThickness**, **gpSetContourLegendBoxType**

gpSetContourLegendBoxThickness

PURPOSE Sets the thickness of the outline of the box surrounding a contour legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLegendBoxThickness**(*pgp*, *framenum*, *contourgroupnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar, contour group number.
thickness	scalar, thickness of the box outline as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendBoxThickness** must be preceded by a call to **gpAddContourLegend**. If the type of the contour box is set to none, then *thickness* is not used. By default, a contour legend has a hollow box (i.e., a box with no background), with an outline that is 0.1% of the frame height.

EXAMPLE

```
framenum = 1;
contourgroupnum = 1;
thickness = 0.4;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendBoxThickness(&gp,framenum,contourgroupnum,thic
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendBoxStyle**

gpSetContourLegendBoxType

PURPOSE Sets the type of the box surrounding a contour legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLegendBoxType**(*pgp*, *framenum*, *contourgroupnum*, *boxtype*);

INPUT

pgp	pointer to a plot control structure
framenum	scalar, frame number
contourgroupnum	scalar, contour group number
boxtype	scalar, one of the following:

1 no box.
2 hollow box (no background).
3 filled box.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendBoxType** must be preceded by a call to **gpAddContourLegend**. By default, a contour legend has a hollow box (i.e., a box with no background).

EXAMPLE

```
framenum = 1;
contourgroupnum = 1;
boxtype = 3;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendBoxType(&gp,framenum,contourgroupnum,boxtype);
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendBoxStyle**

gpSetContourLegendHeaderFont

PURPOSE Sets the font of the header text in a contour legend.

LIBRARY `gausssplot`

FORMAT *ret* = **gpSetContourLegendHeaderFont**(*pgp*, *framenum*, *contourgroupnum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar, contour group number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendHeaderFont** must be preceded by a call to **gpAddContourLegend**. By default, the header text in a contour legend is written in a helvetica font (*fonttype* = "helv"). The default font height is 2.5% of the frame height. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
contourgroupnum = 2;
height = 18;
heightunit = 2;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendHeaderFont(&gp,framenum,contourgroupnum,
                                        "timesitalic",height,heightunit);
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendLabelFont**, **gpSetContourLegendLineSpacing**, **gpSetContourLegendMargin**

gpSetContourLegendLabelFont

PURPOSE Sets the font of the labels in a contour legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLegendLabelFont**(*pgp*, *framenum*, *contourgroupnum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar, contour group number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendLabelFont** must be preceded by a call to **gpAddContourLegend**. By default, the labels in a contour legend are written in a helvetica font (*fonttype* = "helv"). The default font height is 2.5% of the frame height. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
```

```
contourgroupnum = 2;
height = 18;
heightunit = 2;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendLabelFont(&gp,framenum,contourgroupnum,
                                        "timesitalic",height,heightunit);
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendHeaderFont**, **gpSetContourLegendLineSpacing**, **gpSetContourLegendMargin**

gpSetContourLegendLineSpacing

PURPOSE Sets the spacing between the lines of text in a contour legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLegendLineSpacing**(*pgp*, *framenum*, *contourgroupnum*, *linespacing*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar, contour group number.
linespacing	scalar, the spacing between lines of text in contour legend.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendLineSpacing** must be preceded by a call to **gpAddContourLegend**. By default, line spacing is set to 1.

EXAMPLE
```
framenum = 1;
contourgroupnum = 1;
linespacing = 1.25;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendLineSpacing(&gp,framenum,contourgroupnum,lines
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendLabelFont**, **gpSetContourLegendMargin**, **gpSetContourLegendSkip**

gpSetContourLegendMargin

PURPOSE Sets the size of the margin between the contour legend text and the surrounding box.

LIBRARY gausssplot

FORMAT *ret* = **gpSetContourLegendMargin**(*pgp*, *framenum*, *contourgroupnum*, *margin*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar, contour group number.
margin	scalar, the size of the margin between the contour legend text and the surrounding box as a percentage of the legend box width.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendMargin** must be preceded by a call to **gpAddContourLegend**. By default, the margin is set to 10.

EXAMPLE
```
framenum = 1;
contourgroupnum = 1;
margin = 20;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendMargin(&gp,framenum,contourgroupnum,margin);
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendLineSpacing**, **gpSetContourLegendSkip**

gpSetContourLegendOrientation

PURPOSE Sets the orientation of a contour legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLegendOrientation(***pgp*, *framenum*, *contourgroupnum*, *orientation***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar, contour group number.
orientation	scalar, one of the following:
	1 legend oriented vertically.
	2 legend oriented horizontally.

Output

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendOrientation** must be preceded by a call to **gpAddContourLegend**. By default, orientation is vertical.

EXAMPLE
```
framenum = 1;
contourgroupnum = 1;
orientation = 2;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendOrientation(&gp,framenum,contourgroupnum,orien
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpMoveContourLegend**, **gpSetContourLegendView**

gpSetContourLegendSkip

PURPOSE Sets an option to show only one in every *skipnum* contour levels in a contour legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLegendSkip(***pgp*, *framenum*, *contourgroupnum*, *skipnum***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar, contour group number.
skipnum	scalar, show only one in every *skipnum* contour levels in contour legend.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendSkip** must be preceded by a call to **gpAddContourLegend**. By default, all contour levels are shown (*skipnum* = 1).

EXAMPLE
```
framenum = 1;
contourgroupnum = 1;
skipnum = 2;

ret = gpAddContourLegend(\verb+&+gp,framenum,contourgroupnum);
ret = gpSetContourLegendSkip(\verb+&+gp,framenum,contourgroupnum,skipnu
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendLineSpacing**, **gpSetContourLegendMargin**

gpSetContourLegendView

PURPOSE Sets view options for a contour legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLegendView**(*pgp*, *framenum*, *contourgroupnum*, *showheader*, *showlines*);

INPUT

pgp pointer to a plot control structure

framenum scalar, frame number

contourgroupnum scalar, contour group number

showheader scalar, one of the following:

- **0** do not show contour legend header.
- **1** show contour legend header containing the name of the contour variable.

showlines scalar, one of the following:

0 do not show black lines separating the contour color bands.
1 show black lines separating the contour color bands.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetContourLegendView** must be preceded by a call to **gpAddContourLegend**. By default, both the contour legend header and lines separating the contour color bands are shown.

EXAMPLE
```
framenum = 1;
contourgroupnum = 1;
showheader = 1;
showlines = 0;

ret = gpAddContourLegend(&gp,framenum,contourgroupnum);
ret = gpSetContourLegendView(&gp,framenum,
                                    contourgroupnum,showheader,showlines);
```

SOURCE gpcontourplot.src

SEE ALSO **gpAddContourLegend**, **gpSetContourLegendOrientation**, **gpSetContourLegendBoxStyle**

gpSetContourLineColor

PURPOSE Sets the color of the contour lines in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetContourLineColor**(*pgp*, *framenum*, *fieldzonenum*, *linecolor*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	fieldzonenum	scalar or N×1 vector, field zone number(s).
	linecolor	string, name of contour line color. – or – scalar, index of color in color list.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Field zone numbers correspond to the number of the associated zone in the data file. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
fieldzonenum = 2;
floodon = 1;
lineson = 1;

ret = gpShowFieldZoneContourLayer(&gp,framenum,fieldzonenum,floodon,li
ret = gpSetContourLineColor(&gp,framenum,fieldzonenum,"purple");
```

SOURCE `gpcontourplot.src`

SEE ALSO **gpSetContourLineGroup**, **gpSetContourVariable**

gpSetContourLineGroup

PURPOSE Sets the contour group to be used for contour lines in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* **= gpSetContourLineGroup(***pgp***,** *framenum***,** *fieldzonenum***,** *contourgroupnum***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
contourgroupnum	scalar, contour group number.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

All contour options are set within a contour group. Each field zone that is displayed as a contour plot must be associated with a particular contour group, which controls the contour options. Contour plots can be displayed with contour lines and/or contour flooding. If both lines and flooding are used in a field zone, they may be associated with different contour groups.

By default, a contour plot is displayed with contour flooding only, using contour group one to control all options. Use **gpShowFieldZoneContourLayer** to activate or deactivate contour lines and flooding. If contour lines are activated for a field, contour group one will be used to control the contour options by default. Calling **gpSetContourLineGroup** overrides that default.

EXAMPLE

```
framenum = 1;
contourgroupnum = 2;
variableindex = 4;
fieldzonenum = 1;
floodon = 1;
lineson = 1;

ret = gpShowFieldZoneContourLayer(&gp,framenum,fieldzonenum,floodon,li
ret = gpSetContourVariable(&gp,framenum,contourgroupnum,variableindex)
ret = gpSetContourLineGroup(&gp,framenum,fieldzonenum,contourgroupnum)
```

This example first activates contour lines for field zone one in frame one. It then sets the contour variable in contour group two to the fourth variable in the data file and sets the contour group to be used for contour lines to contour group two.

SOURCE

gpcontourplot.src

SEE ALSO **gpSetContourFloodGroup**, **gpSetContourVariable**, **gpSetContourLineColor**

gpSetContourVariable

PURPOSE Sets the contour variable for the specified contour group(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetContourVariable**(*pgp*, *framenum*, *contourgroupnum*, *variable*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
contourgroupnum	scalar or N×1 vector, contour group number(s).
variable	scalar, index of variable in the data file. – or – string, name of variable.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS All contour options are set within a contour group. Each field zone that is displayed as a contour plot must be associated with a particular contour group, which controls the contour options. Contour plots can be displayed with contour lines and/or contour flooding. If both lines and flooding are used in a field zone, they may be associated with different contour groups.

By default, a contour plot is displayed with contour flooding only, using contour group one to control all options. Each contour group uses the last variable in the data file as the default contour variable.

EXAMPLE
```
framenum = 1;
contourgroupnum = 1;
```

```
variableindex = 2;

ret = gpSetContourVariable(&gp,framenum,contourgroupnum,variableindex);
```

This example sets the contour variable in contour group one to the second variable in the data file.

SOURCE gpcontourplot.src

SEE ALSO **gpSetContourFloodGroup**, **gpSetContourLineGroup**

gpSetCurveFitDataFile

PURPOSE Specifies files to which curve fit details and data are written.

LIBRARY gaussplot

FORMAT *ret* = **gpSetCurveFitDataFile**(*pgp*, *framenum*, *linemapnum*, *datafile*, *detailfile*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar, line map number.
datafile	string or null string, name of curve fit data file.
detailfile	string or null string, name of curve fit detail file.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command allows you to specify files to which the details for a curve fit function and the curve data are written. Both the curve fit data and details are written in ASCII format.

If *datafile* is a null string, the curve data will not be written to a file. Likewise, if *detailfile* is a null string, the details for the curve fit function will not be written to a file.

The *linemapnum* argument must refer to a line map to which a curve fit has been applied with **gpSetCurveFitType**.

EXAMPLE
```
framenum = 1;
linemapnum = 1;
ret = gpSetCurveFitDataFile(&gp,framenum,linemapnum,
                                    "curvedat.txt","curvedetails.txt");
```

SOURCE gp.src

SEE ALSO **gpSetCurveFitType**

gpSetCurveFitDependentVariable

PURPOSE Specifies whether the variable associated with the X-axis or the variable associated with the Y-axis should be the dependent variable in computing a curve fit.

LIBRARY gaussplot

FORMAT *ret* = **gpSetCurveFitDependentVariable**(*pgp*, *framenum*, *linemapnum*, *axis*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
axis	scalar, one of the following:

1 sets the variable associated with the X-axis as the dependent variable, computing a curve fit function $x = f(y)$.

2 sets the variable associated with the Y-axis as the dependent variable, computing a curve fit function $y = f(x)$.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is used only for line maps to which a curve fit has been applied with **gpSetCurveFitType**. By default, the variable associated with the Y-axis is used as the dependent variable.

EXAMPLE
```
framenum = 1;
linemapnum = 3;
type = 3;
axis = 1;
ret = gpSetCurveFitType(&gp,framenum,linemapnum,type);
ret = gpSetCurveFitDependentVariable(&gp,framenum,linemapnum,axis);
```

SOURCE gplinemap.src

SEE ALSO **gpSetCurveFitType**, **gpSetCurveFitPoints**, **gpSetCurveFitRange**

gpSetCurveFitPoints

PURPOSE Sets the number of points on a curve.

LIBRARY gaussplot

FORMAT *ret* = **gpSetCurveFitPoints**(*pgp*, *framenum*, *linemapnum*, *points*);

INPUT *pgp* pointer to a plot control structure.

framenum scalar, frame number.

	linemapnum	scalar or N×1 vector, line map number(s).
	points	scalar, number of points.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is used only for line maps to which a curve fit has been applied with **gpSetCurveFitType**. By default, there are 200 data points on a curve.

EXAMPLE
```
framenum = 1;
linemapnum = 3;
type = 3;
points = 400;
ret = gpSetCurveFitType(&gp,framenum,linemapnum,type);
ret = gpSetCurveFitPoints(&gp,framenum,linemapnum,points);
```

SOURCE gplinemap.src

SEE ALSO **gpSetCurveFitType**, **gpSetCurveFitRange**, **gpSetCurveFitWeightVariable**

gpSetCurveFitPolynomialOrder

PURPOSE Sets the order of the polynomial function to be used for a curve fit.

LIBRARY gaussplot

FORMAT *ret* = **gpSetCurveFitPolynomialOrder(***pgp*, *framenum*, *linemapnum*, *order***);**

INPUT	*pgp*	pointer to a plot control structure.
	framenum	scalar, frame number.
	linemapnum	scalar or N×1 vector, line map number(s).

	order	scalar, 1 ≤ order ≤ 10, order of the polynomial function.

OUTPUT

	ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is used only for line maps to which a Polynomial curve fit has been applied with **gpSetCurveFitType**. By default, a third-order polynomial function is used. If *order* = 1, a Linear fit is applied.

EXAMPLE
```
framenum = 1;
linemapnum = 3;
type = 3;
order = 4;
ret = gpSetCurveFitType(&gp,framenum,linemapnum,type);
ret = gpSetCurveFitPolynomialOrder(&gp,framenum,linemapnum,order);
```

SOURCE gplinemap.src

SEE ALSO **gpSetCurveFitType**, **gpSetCurveFitPoints**, **gpSetCurveFitRange**, **gpSetCurveFitWeightVariable**

gpSetCurveFitRange

PURPOSE Sets the range of the independent variable for a curve.

LIBRARY gaussplot

FORMAT *ret* = **gpSetCurveFitRange**(*pgp*, *framenum*, *linemapnum*, *min*, *max*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
min	scalar, minimum value of the independent variable.

max scalar, maximum value of the independent variable.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is used only for line maps to which one of the following curve fits has been applied: Linear fit, Polynomial fit, Exponential fit, Power fit, or the Stineman interpolation. Call **gpSetCurveFitType** to apply a curve fit to a line map. By default, there is no finite range set on the independent variable.

EXAMPLE
```
framenum = 1;
linemapnum = 3;
type = 2;
min = 0;
max = 100;
ret = gpSetCurveFitType(&gp,framenum,linemapnum,type);
ret = gpSetCurveFitRange(&gp,framenum,linemapnum,min,max);
```

SOURCE gplinemap.src

SEE ALSO **gpSetCurveFitType**, **gpSetCurveFitPoints**, **gpSetCurveFitWeightVariable**

gpSetCurveFitSplineClamp

PURPOSE Sets the derivatives at the beginning and ending points of a spline fit curve.

LIBRARY gaussplot

FORMAT *ret* = **gpSetCurveFitSplineClamp**(*pgp*, *framenum*, *linemapnum*, *bderiv*, *ederiv*);

INPUT *pgp* pointer to a plot control structure.

framenum scalar, frame number.

	linemapnum	scalar or N×1 vector, line map number(s).
	bderiv	scalar, derivative at the beginning point of the spline fit curve.
	ederiv	scalar, derivative at the ending point of the spline fit curve.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is used only for line maps to which a Spline fit has been applied with **gpSetCurveFitType**. By default, a Spline fit curve is not clamped, that is, the derivatives at the beginning and ending points are not pre-set. A spline that is not clamped must be twice differentiable at the beginning and ending points and have a second derivative of zero at both points. However, a clamped spine need not be twice differentiable at its endpoints.

EXAMPLE
```
framenum = 1;
linemapnum = 3;
type = 6;
bderiv = 0;
ederiv = 5;
ret = gpSetCurveFitType(&gp,framenum,linemapnum,type);
ret = gpSetCurveFitSplineClamp(&gp,framenum,linemapnum,bderiv,ederiv);
```

SOURCE gplinemap.src

SEE ALSO **gpSetCurveFitType**, **gpSetCurveFitPoints**

gpSetCurveFitType

PURPOSE Sets the curve fit type of the line(s) in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetCurveFitType**(*pgp*, *framenum*, *linemapnum*, *type*);

INPUT

pgp pointer to a plot control structure.

framenum scalar, frame number.

linemapnum scalar or N×1 vector, line map number(s).

type scalar, one of the following:

1 No curve fit, linear segments are drawn between data points.
2 Linear fit.
3 Polynomial fit.
4 Exponential fit.
5 Power fit.
6 Spline fit.
7 Parametric spline fit.
8 Akima spline fit.
9 General curve fit.
10 Stineman interpolation.

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

By default, lines are drawn as line segments between data points.

Following is a description of each of the curve fit types:

Linear fit Fits a linear function to the data.

Polynomial fit Fits a polynomial function of the N^{th} order to the data, where $1 \leq N \leq 10$. If N = 1, a linear function is used. By default, N = 3. Call **gpSetCurveFitPolynomialOrder** to change the value of N.

Exponential fit Fits an exponential function of the form $y = a * e^{b*x}$ to the data. An Exponential fit can be used only when the *y*-values are all <0 or all >0.

Power fit Fits a power function of the form $y = a * x^b$ to the data. A Power fit can be used only when the *y*-values are all < 0 or all > 0.

Spline fit Draws a smooth curve that passes through every data point. Before the curve is drawn, the data points are sorted on the independent variable,

from lowest to highest. You may clamp the spline by specifying the derivative of the fit function at the beginning and ending points of the curve with **gpSetCurveFitSplineClamp**.

Parametric spline fit Draws a smooth curve that passes through every data point, using $x = f(i)$, and $y = g(i)$, where i is the index of the data point, and f and g are smooth functions. You may clamp the spline by specifying the derivative of the fit function at the beginning and ending points of the curve with **gpSetCurveFitSplineClamp**.

Akima spline fit A Spline fit that uses a nonlinear average of the line segment slopes to find the slopes at the ending points of each segment.

General curve fit Fits an equation of the form $y = c_1 * f_1(x) + c_2 * f_2(x) + \ldots + c_n * f_n(x)$, where $f_1, f_2, \ldots, f_n$ are user-specified sub-functions, $c_1, c_2, \ldots, c_n$ are constants and $1 \leq n \leq 8$. By default, the function $y = c_1 + c_2 * x + c_3 * x^2$ is used. To use a different function or to specify whether the fit should use normalized x and/or y variables, call **gpSetGeneralCurveFitOptions**.

Stineman interpolation Fits a curve that passes through every data point, reflecting the computed slope at each point. The curve will not have more inflection points than the data require.

A least squares algorithm is used to compute Linear, Polynomial, Exponential, and Power fits.

By default, a curve fit function is computed using x as the independent variable and y as the dependent variable. You may change this by calling **gpSetCurveFitDependentVariable**.

The number of points on a curve is 200 by default. To specify a different number of points, call **gpSetCurveFitPoints**.

For Linear, Polynomial, Exponential, and Power fits as well as the Stineman interpolation, you may specify a range for the independent variable with **gpSetCurveFitRange**. You may also use a weight variable for Linear, Polynomial, Exponential, and Power fits. Call **gpSetCurveFitWeightVariable** to specify a weight variable.

EXAMPLE

```
framenum = 1;
```

```
linemapnum = 3;
type = 3;
order = 2;
ret = gpSetCurveFitType(&gp,framenum,linemapnum,type);
ret = gpSetCurveFitPolynomialOrder(&gp,framenum,linemapnum,order);
```

SOURCE gplinemap.src

SEE ALSO **gpSetCurveFitPoints**, **gpSetCurveFitDependentVariable**, **gpSetCurveFitRange**, **gpSetCurveFitWeightVariable**, **gpSetCurveFitDataFile**

gpSetCurveFitWeightVariable

PURPOSE Specifies a weight variable for a curve.

LIBRARY gaussplot

FORMAT *ret* = **gpSetCurveFitWeightVariable**(*pgp*, *framenum*, *linemapnum*, *variable*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
variable	scalar, index of variable in the data file. – or – string, name of variable.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is used only for line maps to which one of the following curve fits has been applied: Linear fit, Polynomial fit, Exponential fit, or Power fit. Call **gpSetCurveFitType** to apply a curve fit to a line map.

By default, no weight variable is used.

EXAMPLE

```
framenum = 1;
linemapnum = 3;
type = 2;
var = 5;
ret = gpSetCurveFitType(&gp,framenum,linemapnum,type);
ret = gpSetCurveFitWeightVariable(&gp,framenum,linemapnum,var);
```

SOURCE gplinemap.src

SEE ALSO **gpSetCurveFitType**, **gpSetCurveFitPoints**, **gpSetCurveFitRange**

gpSetDataFile

PURPOSE Sets the data file to be used for the specified frame(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetDataFile**(*pgp*, *framenum*, *datafile*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
datafile	string, name of data file.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS If *framenum* = 0, then *datafile* will be set as the data file for all frames in the graph. Otherwise, the data file will be set for only the specified frames. If you specify frame number(s), then the specified frames must already exist when you call **gpSetDataFile**. You must specify a data file for all of the frames in a graph before calling **gpPlot** to plot the graph.

EXAMPLE
```
framenum = 0;
ret = gpSetDataFile(&gp,framenum,"mydata.plt");
```

This example sets `mydata.plt` as the data file to be used for all of the frames in the graph.

SOURCE `gp.src`

SEE ALSO **gpSetMacroFile**

gpSetDataSetTitle

PURPOSE Sets the title of a data set.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetDataSetTitle**(*pgdat*, *title*);

INPUT

pgdat	pointer to a **GAUSSplot** data structure.
title	string, title of data set.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The data set title is written at the top of the data file and is displayed in the headers of any frames that are associated with the data file by default. Call **gpSetDataSetTitle** before **gpWritePlotData** to specify a data set title.

If a data set title is not specified, then the name of the data file, without the `.plt` extension, will be used as the title.

EXAMPLE
```
library gaussplot;
#include gp.sdf
```

```
x = { 0, 0.4, 0.5, 0.67, 0.8, 1.05, 1.5, 1.6, 1.79, 1.9 };
y = { 14, 30, 18, 22, 29, 37, 42, 43, 32, 38 };

struct gpData gdat;
string vnames = { "X", "Y" };
gdat = gpSetPlotData(x~y,vnames);
ret = gpSetDataSetTitle(&gdat,"First Observation");
ret = gpWritePlotData(&gdat,"mydata.plt");
```

This example creates the data file `mydata.plt` and sets "First Observation" as the title of the data set.

SOURCE `gpwritedata.src`

SEE ALSO **gpWritePlotData**, **gpSetFrameHeaderText**

gpSetDefaultTextFont

PURPOSE Sets the default font to be used for text objects in a plot.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetDefaultTextFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following:

1 font height as a percentage of the height of the frame.
2 font height in points.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all text objects are written in a bold helvetica font (*fonttype* = "helvbold"). The default font height is 14 points. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
fontheight = 5;
heightunit = 1;
ret = gpSetDefaultTextFont(&gp,framenum,"timesbold",fontheight,heightu
```

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectFont**

gpSetErrorBarColor

PURPOSE Sets the color of the error bars in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetErrorBarColor**(*pgp*, *framenum*, *linemapnum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
linecolor	string, name of error bar color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the error bar layer is turned off. To activate the error bar layer in a frame, call **gpShowErrorBarLayer**.

See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
seton = 1;
linemapnum = 2;
ret = gpShowErrorBarLayer(&gp,framenum,seton);
ret = gpSetErrorBarColor(&gp,framenum,linemapnum,"purple");
```

SOURCE gplinemap.src

SEE ALSO **gpShowErrorBarLayer**, **gpShowLineMapErrorBarLayer**, **gpSetErrorBarVariable**, **gpSetErrorBarSize**

gpSetErrorBarLineThickness

PURPOSE Sets the line thickness of the error bars in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetErrorBarLineThickness**(*pgp*, *framenum*, *linemapnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
thickness	scalar, line thickness of the error bars as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the error bar layer is turned off. To activate the error bar layer in a frame, call **gpShowErrorBarLayer**. When the layer is activated, all error bars are set to a line thickness of 0.1 by default.

EXAMPLE

```
framenum = 3;
seton = 1;
linemapnum = 2;
thickness = 0.4;
ret = gpShowErrorBarLayer(&gp,framenum,seton);
ret = gpSetErrorBarLineThickness(&gp,framenum,linemapnum,thickness);
```

SOURCE gplinemap.src

SEE ALSO **gpSetErrorBarColor**, **gpSetErrorBarSize**, **gpSetErrorBarSkip**, **gpSetErrorBarType**

gpSetErrorBarSize

PURPOSE Sets the length of the crossbar for the error bars in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetErrorBarSize**(*pgp*, *framenum*, *linemapnum*, *size*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
size	scalar, length of the crossbar for the error bars as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the error bar layer is turned off. To activate the error bar layer in a frame, call **gpShowErrorBarLayer**. When the layer is activated, the crossbar for an error bar is set to a length of 2.5 by default.

EXAMPLE
```
framenum = 3;
seton = 1;
linemapnum = 2;
size = 4;
ret = gpShowErrorBarLayer(&gp,framenum,seton);
ret = gpSetErrorBarSize(&gp,framenum,linemapnum,size);
```

SOURCE gplinemap.src

SEE ALSO **gpSetErrorBarColor**, **gpSetErrorBarLineThickness**, **gpSetErrorBarSkip**, **gpSetErrorBarType**

gpSetErrorBarSkip

PURPOSE Sets an option to show error bars at only a subset of the data points in the specified line map(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetErrorBarSkip**(*pgp*, *framenum*, *linemapnum*, *skipnum*, *skipmode*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
skipnum	scalar, skip.
skipmode	scalar, one of the following:

1 show error bars at every data point.

2 show error bars at only one in every *skipnum* data points.

3 show error bars at the first data point and at each successive data point that is at least *skipnum* percent of the frame height from the previous error bar.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the error bar layer is turned off. To activate the error bar layer in a frame, call **gpShowErrorBarLayer**. When the layer is activated, error bars are shown at every data point by default. If *skipmode* is set to 1, then *skipnum* is ignored.

EXAMPLE

```
framenum = 3;
seton = 1;
linemapnum = 2;
skipnum = 3;
```

```
skipmode = 2;
ret = gpShowErrorBarLayer(&gp,framenum,seton);
ret = gpSetErrorBarSkip(&gp,framenum,linemapnum,skipnum,skipmode);
```

SOURCE gplinemap.src

SEE ALSO **gpSetErrorBarColor**, **gpSetErrorBarLineThickness**, **gpSetErrorBarSize**, **gpSetErrorBarType**

gpSetErrorBarType

PURPOSE Sets the type of the error bars in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetErrorBarType**(*pgp*, *framenum*, *linemapnum*, *type*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
type	scalar, one of the following:

1 error bars are drawn up from the data point where the error bar variable is positive and down from the data point where the error bar variable is negative.

2 error bars are drawn down from the data point where the error bar variable is positive and down from the data point where the error bar variable is negative.

3 error bars are drawn to the left of the data point where the error bar variable is positive and to the right of the data point where the error bar variable is negative.

4 error bars are drawn to the right of the data point where the error bar variable is positive and to the left of the data point where the error bar variable is negative.

5 error bars are drawn both to the left and the right of the data points.

6 error bars are drawn both up and down from the data points.

7 error bars are drawn up and down from the data points as well as to the left and to the right of the data points.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the error bar layer is turned off. To activate the error bar layer in a frame, call **gpShowErrorBarLayer**. When the layer is activated, error bars are drawn both up and down from the data points by default.

EXAMPLE
```
framenum = 3;
seton = 1;
linemapnum = 2;
type = 4;
ret = gpShowErrorBarLayer(&gp,framenum,seton);
ret = gpSetErrorBarType(&gp,framenum,linemapnum,type);
```

SOURCE gplinemap.src

SEE ALSO **gpSetErrorBarColor**, **gpSetErrorBarSize**, **gpSetErrorBarVariable**

gpSetErrorBarVariable

PURPOSE Sets the variable to be used for the error bars in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetErrorBarVariable**(*pgp*, *framenum*, *linemapnum*, *variable*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	linemapnum	scalar or N×1 vector, line map number(s).
	variable	scalar, index of variable in the data file. – or – string, name of variable.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the error bar layer is turned off. To activate the error bar layer in a frame, call **gpShowErrorBarLayer**. When the error bar layer is activated, the second variable in the data file will be used as the error bar variable by default.

The error bar variable controls the magnitude of the error bars from the data points. By default, error bars are drawn both up and down from the data points (call **gpSetErrorBarType** to change this behavior). Therefore, if the error bar variable is defined as an N×1 vector containing 0.5 in each row, then at each data point, error bars will be drawn from the data point to 0.5 units above the data point as well as from the data point to 0.5 units below.

EXAMPLE

```
framenum = 1;
seton = 1;
linemapnum = 2;
variableindex = 5;
ret = gpShowErrorBarLayer(&gp,framenum,seton);
ret = gpSetErrorBarVariable(&gp,framenum,linemapnum,variableindex);
```

This example sets the error bar variable for line map two in frame one to the fifth variable in the data file.

SOURCE gplinemap.src

SEE ALSO **gpShowErrorBarLayer**, **gpShowLineMapErrorBarLayer**, **gpSetErrorBarColor**, **gpShowErrorBarType**

gpSetErrorLevel

PURPOSE Sets the level of error handling in your program.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetErrorLevel**(*pgp*, *level*);

INPUT

pgp	pointer to a plot control structure.
level	scalar, one of the following:

- **1** return error codes if errors are encountered.
- **2** log error messages and return error codes if errors are encountered.
- **3** log error message and stop program run if an error is encountered.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, **GAUSSplot** commands simply return error codes if errors are encountered. You may pass the value of the error code into **gpGetErrorMessage** to get the associated message.

To set the level of error handling for your entire program, call **gpSetErrorLevel** immediately after initializing the plot control structure with one of the **gp***PlotType***PlotCreate** commands.

The data write commands **gpAddZone** and **gpWritePlotData** do not use the error handling level specified by a call to **gpSetErrorLevel**. You are responsible for checking the returns from these commands yourself.

EXAMPLE

```
library gaussplot;
#include gp.sdf
```

```
p = pi;
x = seqa(0,p/4,15);
y = sin(x);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpBarPlotCreate;

ret = gpSetErrorLevel(&gp,3);
call gpMakeBarPlot(&gp,x,y,vnames);
```

SOURCE gp.src

SEE ALSO **gpGetErrorMessage**

gpSetExportImageWidth

PURPOSE Specifies the width of the exported image in pixels.

LIBRARY gaussplot

FORMAT **gpSetExportImageWidth(***pgp*, *width***);**

INPUT

pgp	pointer to a plot control structure.
width	scalar, width of the exported image in pixels.

REMARKS By default, the width of the exported image is set to 671 pixels.

Calling **gpSetExportImageWidth** sets the image width for all subsequent exports to file formats for which the image width option is used until another call to **gpSetExportImageWidth** is encountered. The image width option is used for exports to the following file formats:

File Format	Export Command
BMP	**gpExportBMP**
JPEG	**gpExportJPEG**
PNG	**gpExportPNG**
Sun Raster	**gpExportRAS**
TIFF	**gpExportTIFF**
X-Windows	**gpExportXWD**

EXAMPLE

```
area = -2;
gpSetExportImageWidth(&gp,500);
ret = gpExportBMP(&gp,area,"graphic.bmp");
```

This example sets the width of the exported image to 500, and then exports the entire workspace to the BMP graphics file `graphic.bmp`.

SOURCE `gp.src`

SEE ALSO **gpSetExportJPEGQuality**, **gpSetPrintPalette**

gpSetExportJPEGQuality

PURPOSE Specifies the quality of an output JPEG graphics file.

LIBRARY `gaussplot`

FORMAT **gpSetExportJPEGQuality(***pgp*, *quality***);**

INPUT

pgp	pointer to a plot control structure.
quality	scalar, $1 \leq quality \leq 100$, quality of the output JPEG file.

REMARKS The higher *quality* is set, the larger and higher quality the JPEG output file. By default, *quality* is set to 75.

Calling **gpSetExportJPEGQuality** sets the quality for all subsequent calls to **gpExportJPEG** until another call to **gpSetExportJPEGQuality** is encountered.

EXAMPLE

```
quality = 95;
area = -2;
gpSetExportJPEGQuality(&gp,quality);
ret = gpExportJPEG(&gp,area,"graphic.jpg");
```

This example sets the quality of the JPEG output file to 95, and then exports the entire workspace to the JPEG graphics file `graphic.jpg`.

SOURCE `gp.src`

SEE ALSO **gpExportJPEG**, **gpSetExportImageWidth**

gpSetFieldZoneLegendBoxColor

PURPOSE Sets the outline and background colors of the box surrounding a field zone legend.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetFieldZoneLegendBoxColor**(*pgp*, *framenum*, *linecolor*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
linecolor	string, name of color to use for box outline. – or – scalar, index of color in color list.
fillcolor	string, name of color to use for box background. – or – scalar, index of color in color list.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendBoxColor** must be preceded by a call to **gpAddFieldZoneLegendFrame**. If the field zone legend box type is set to none, then these colors are not used. If the line legend box type is set to hollow, then *fillcolor* is not used, since a hollow box has no background. By default, a field zone legend has a filled box with a black outline and a white background. To change the box type, call **gpSetFieldZoneLegendBoxType**.

See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
boxtype = 2;
linecolor = "red";
fillcolor = "";

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendBoxType(&gp,framenum,boxtype);
ret = gpSetFieldZoneLegendBoxColor(&gp,framenum,linecolor,fillcolor);
```

In this example, *fillcolor* is set to a null string because the box type is hollow and thus has no background.

SOURCE `gpfieldzone.src`

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendBoxStyle**

gpSetFieldZoneLegendBoxStyle

PURPOSE Sets the style of the box surrounding a field zone legend.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetFieldZoneLegendBoxStyle**(*pgp*, *framenum*, *boxtype*, *thickness*, *linecolor*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
boxtype	scalar, one of the following: **1** no box. **2** hollow box (no background). **3** filled box.
thickness	scalar, thickness of the box outline as a percentage of the height of the parent frame.
linecolor	string, name of color to use for box outline. – or – scalar, index of color in color list.
fillcolor	string, name of color to use for box background. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

This command is a convenience function that allows you to set several parameters at one time. See **gpSetFieldZoneLegendBoxType**, **gpSetFieldZoneLegendBoxThickness**, and **gpSetFieldZoneLegendBoxColor** to set the box style parameters individually.

A call to **gpSetFieldZoneLegendBoxStyle** must be preceded by a call to **gpAddFieldZoneLegendFrame**. If *boxtype* is specified as hollow, then *fillcolor* is not used, since a hollow box has no background. If it is specified as none, then *linecolor*, *fillcolor* and *thickness* are not used. By default, a field zone legend has a filled box with a black outline that is 0.1% of the frame height and a white background.

See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
boxtype = 2;
thickness = 0.4;
linecolor = "red";
fillcolor = "";

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendBoxStyle(&gp,framenum,boxtype,
                                   thickness,linecolor,fillcolor);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendBoxColor**, **gpSetFieldZoneLegendBoxThickness**, **gpSetFieldZoneLegendBoxType**

gpSetFieldZoneLegendBoxThickness

PURPOSE Sets the thickness of the outline of the box surrounding a field zone legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetFieldZoneLegendBoxThickness**(*pgp*, *framenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
thickness	scalar, thickness of the box outline as a percentage of the height of the parent frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendBoxThickness** must be preceded by a call to **gpAddFieldZoneLegendFrame**. If the type of the field zone legend box is set to none, then *thickness* is not used. By default, a field zone legend has a filled box with an outline that is 0.1% of the parent frame height.

EXAMPLE
```
framenum = 1;
thickness = 0.4;

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendBoxThickness(&gp,framenum,thickness);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendBoxStyle**

gpSetFieldZoneLegendBoxType

PURPOSE Sets the type of the box surrounding a field zone legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetFieldZoneLegendBoxType**(*pgp*, *framenum*, *boxtype*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
boxtype	scalar, one of the following: **1** no box. **2** hollow box (no background). **3** filled box.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendBoxType** must be preceded by a call to **gpAddFieldZoneLegendFrame**. By default, a field zone legend has a filled box.

EXAMPLE
```
framenum = 1;
boxtype = 1;

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendBoxType(&gp,framenum,boxtype);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendBoxStyle**

gpSetFieldZoneLegendFont

PURPOSE Sets the font of the text in a field zone legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetFieldZoneLegendFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following:

1 font height as a percentage of the height of the parent frame.

2 font height in points.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendFont** must be preceded by a call to **gpAddFieldZoneLegendFrame**. By default, the text in a field zone legend is written in a helvetica bold font (*fonttype* = "helvbold"). The default font height is 3% of the parent frame height. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
height = 13;
heightunit = 2;

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendFont(&gp,framenum,"timesitalic",height,heigh
```

SOURCE gpfieldzone.src

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendLineSpacing**,

gpSetFieldZoneLegendTitle

gpSetFieldZoneLegendLineSpacing

PURPOSE Sets the spacing between the lines of text in a field zone legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetFieldZoneLegendLineSpacing**(*pgp*, *framenum*, *linespacing*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
linespacing	scalar, the spacing between lines of text in field zone legend.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendLineSpacing** must be preceded by a call to **gpAddFieldZoneLegendFrame**. By default, the line spacing is set to 1.

EXAMPLE

```
framenum = 1;
linespacing = 1.25;

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendLineSpacing(&gp,framenum,linespacing);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendFont**, **gpSetFieldZoneLegendUpperMargin**, **gpSetFieldZoneLegendLowerMargin**

gpSetFieldZoneLegendLowerMargin

PURPOSE Sets the size of the margin between the bottom of the field zone legend text and the bottom of the surrounding box.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetFieldZoneLegendLowerMargin**(*pgp*, *framenum*, *margin*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
margin	scalar, the size of the margin between the bottom of the field zone legend text and the bottom of the surrounding box in relation to the height of the legend font.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendLowerMargin** must be preceded by a call to **gpAddFieldZoneLegendFrame**.

If the margin is set to 1, then the size of the margin equals the height of the font used for the legend text. By default, the margin is set to 0.5, which produces a margin that is half the size of the font used for the legend text.

EXAMPLE
```
framenum = 1;
margin = 1;

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendLowerMargin(&gp,framenum,margin);
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendUpperMargin**, **gpSetFieldZoneLegendLineSpacing**

gpSetFieldZoneLegendTitle

PURPOSE Sets a title for a field zone legend.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetFieldZoneLegendTitle**(*pgp*, *framenum*, *title*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
title	string, title for the legend.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendTitle** must be preceded by a call to **gpAddFieldZoneLegendFrame**. By default, a field zone legend has no title.

If **gpSetFieldZoneLegendTitle** is called, then *title* is added to the top of the field zone legend, centered horizontally. The spaces between the top of the legend text and the bottom of the title and between the top of the title and the top of the surrounding box are both controlled by **gpSetFieldZoneLegendUpperMargin**.

EXAMPLE
```
framenum = 1;
title = "Field Zone Legend";
ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendTitle(&gp,framenum,title);
```

SOURCE gpfieldzone.src

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendTitleFont**

gpSetFieldZoneLegendTitleFont

PURPOSE Sets the font of the title in a field zone legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetFieldZoneLegendTitleFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the parent frame. **2** font height in points.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendTitleFont** must be preceded by calls to **gpAddFieldZoneLegendFrame** and **gpSetFieldZoneLegendTitle**. By default, if a title exists, it is written in the same font as the legend text. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
height = 13;
heightunit = 2;

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendTitleFont(&gp,framenum,"timesitalic",height,
```

SOURCE gpfieldzone.src

SEE ALSO **gpAddFieldZoneLegendFrame**, **gpSetFieldZoneLegendTitle**, **gpSetFieldZoneLegendFont**

gpSetFieldZoneLegendUpperMargin

PURPOSE Sets the size of the margin between the top of the field zone legend text and the top of the surrounding box.

LIBRARY gaussplot

FORMAT *ret* = **gpSetFieldZoneLegendUpperMargin**(*pgp*, *framenum*, *margin*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number of the parent frame.
margin	scalar, the size of the margin between the top of the field zone legend text and the top of the surrounding box in relation to the height of the legend font.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

The parent frame is the frame that the field zone legend describes.

A call to **gpSetFieldZoneLegendUpperMargin** must be preceded by a call to **gpAddFieldZoneLegendFrame**.

If the margin is set to 1, then the size of the margin equals the height of the font used for the legend text. By default, the margin is set to 0.5, which produces a margin that is half the size of the font used for the legend text.

If a title is added to the legend with **gpSetFieldZoneLegendTitle**, then *margin* controls the space between the top of the legend text and the bottom of the legend title as well as the space between the top of the legend title and the top of the surrounding box.

EXAMPLE

```
framenum = 1;
margin = 1;

ret = gpAddFieldZoneLegendFrame(&gp,framenum);
ret = gpSetFieldZoneLegendUpperMargin(&gp,framenum,margin);
```

SOURCE

gpfieldzone.src

SEE ALSO

gpAddFieldZoneLegendFrame, **gpSetFieldZoneLegendLowerMargin**, **gpSetFieldZoneLegendLineSpacing**

gpSetFieldZoneNames

PURPOSE Sets names for all of the field zones in a frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetFieldZoneNames**(*pgp*, *framenum*, *names*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
names	N×1 string array, names of the field zones.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, a field zone is given the same name as the associated zone in the data file. If you write the data file yourself, you may specify zone names with **gpAddZone**. Otherwise, default names ("Zone 1", "Zone 2",. . ., "Zone N") are used.

These names are used to identify the field zones in a field zone legend. Call **gpSetFieldZoneNames** to rename the field zones in the frame before adding a legend.

N must equal the number of field zones in the specified frame.

EXAMPLE
```
framenum = 1;
string names = { "Observed data", "Estimation" };
ret = gpSetFieldZoneNames(&gp,framenum,names);
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpAddFieldZoneLegendFrame**

gpSetFrameBackgroundColor

PURPOSE Sets the background color for the specified frame.

LIBRARY `gausssplot`

FORMAT *ret* = **gpSetFrameBackgroundColor**(*pgp*, *framenum*, *backgroundcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
backgroundcolor	string, name of background color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

Call **gpSetFrameStyle** if you do not want to display the frame background.

EXAMPLE
```
framenum = 1;
ret = gpSetFrameBackgroundColor(&gp,framenum,"custom2");
```

SOURCE `gp.src`

SEE ALSO **gpSetFrameHeaderColor**, **gpSetFrameStyle**, **gpSetFrameName**

gpSetFrameHeaderColor

PURPOSE Sets the header color for the specified frame.

LIBRARY `gausssplot`

FORMAT *ret* = **gpSetFrameHeaderColor**(*pgp*, *framenum*, *headercolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
headercolor	string, name of header color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

Call **gpSetFrameStyle** if you do not want to display the frame header.

EXAMPLE
```
framenum = 1;
ret = gpSetFrameHeaderColor(&gp,framenum,"blue");
```

SOURCE `gp.src`

SEE ALSO **gpSetFrameBackgroundColor**, **gpSetFrameStyle**, **gpSetFrameName**

gpSetFrameHeaderText

PURPOSE Sets the text to be displayed in all frame headers in a graph.

LIBRARY `gaussplot`

FORMAT **`gpSetFrameHeaderText(`***pgp*`, `*text*`);`**

INPUT

pgp	pointer to a plot control structure.
name	string, frame header text.

REMARKS The text string may include any normal text characters (although some may be produced only with special input, see below), as well as any or all of the following dynamic text options:

Text Input	Output in Frame Header
&(DATASETTITLE)	Title of data set in associated data file
&(DATE)	Current date
&(FRAMENAME)	Name of frame
&(PLOTTYPE)	Basic plot type of frame, numeric, one of the following: 1 XY Line plot 2 2D Cartesian plot 3 3D Cartesian plot 4 Polar Line plot
&(TIME)	Current time

If you include '&(DATE)' in your frame header text string, the date will be displayed as DD Mon YYYY, where Mon is the abbreviated month name. For example, the date 12/1/2004 would be displayed as '1 Dec 2004'. If you include '&(TIME)' in your frame header text string, the time will be displayed as HH:MI:SS.

The following characters may be produced only with special input:

Character	Text Input
&	\&
\	\\\\
”	\\\”

By default, the following frame header text string is used:

```
"&(FRAMENAME) <math>o</math> &(DATE) <math>o</math> &(DATASETTITLE)"
```

where the substring

```
<math>o</math>
```

produces a long vertical bar. See Appendix A for a list of all of the available characters in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
text = "&(FRAMENAME), &(DATE), &(TIME)";
gpSetFrameHeaderText(&gp,text);
```

SOURCE gp.src

SEE ALSO **gpSetFrameName**, **gpSetFrameStyle**, **gpMoveFrame**

gpSetFrameName

PURPOSE Sets the name for the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetFrameName**(*pgp*, *framenum*, *name*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, frame name.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, a frame is given the name "Frame *framenum*", where *framenum* is padded on the left with zeros to three digits. The name is displayed in the frame header by default.

EXAMPLE
```
framenum = 1;
name = "US Map";
ret = gpSetFrameName(&gp,framenum,name);
```

SOURCE gp.src

SEE ALSO **gpSetFrameHeaderText**, **gpSetFrameStyle**, **gpMoveFrame**

gpSetFrameStyle

PURPOSE Sets style options for the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetFrameStyle**(*pgp*, *framenum*, *showheader*, *showborder*, *showbackground*);

INPUT

pgp	pointer to a plot control structure.

	framenum	scalar, frame number.
	showheader	scalar, one of the following:
		0 do not display frame header.
		1 display frame header.
	showborder	scalar, one of the following:
		0 do not display frame border.
		1 display frame border.
	showbackground	scalar, one of the following:
		0 do not display frame background.
		1 display frame background.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the frame header, border, and background are all displayed.

EXAMPLE
```
framenum = 1;
showheader = 0;
showborder = 0;
showbackground = 1;
ret = gpSetFrameStyle(&gp,framenum,showheader,showborder,showbackgroun
```

SOURCE gp.src

SEE ALSO **gpMoveFrame**, **gpSetFrameName**, **gpSetWorkspaceView**

gpSetGeneralCurveFitOptions

PURPOSE Sets the sub-functions $f_1, f_2, \ldots, f_n$ to be used in a curve fit function of the form $y = c_1 * f_1(x) + c_2 * f_2(x) + \ldots + c_n * f_n(x)$, where $1 \leq n \leq 8$, and specifies whether to use normalized x and/or y variables.

LIBRARY gaussplot

FORMAT *ret* = **gpSetGeneralCurveFitOptions**(*pgp*, *framenum*, *linemapnum*, *terms*, *normx*, *normy*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
terms	M×1 string array, $1 \leq M \leq 8$, terms for the fit function.
normx	scalar, one of the following:
	0 do not use a normalized *x* variable.
	1 use a normalized *x* variable.
normy	scalar, one of the following:
	0 do not use a normalized *y* variable.
	1 use a normalized *y* variable.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command applies only to line maps to which a General curve fit has been applied with **gpSetCurveFitType**. By default, the function $y = c_1 + c_2 * x + c_3 * x^2$ is used.

The syntax for *terms* allows most common C operators and functions. Use 'x' as the independent variable in each of the terms, even if you have set the variable that is associated with the X-axis as the dependent variable by a call to **gpSetCurveFitDependentVariable**. See **GAUSSplot**'s online help for more information by clicking on the 'Help' button on the 'Curves' tab of the 'Mapping Style' dialog box.

EXAMPLE

```
framenum = 1;
linemapnum = 3;
type = 9;
string terms = { "1", "sin(x)", "cos(x)" };
normx = 0;
normy = 0;
ret = gpSetCurveFitType(&gp,framenum,linemapnum,type);
ret = gpSetGeneralCurveFitOptions(&gp,framenum,linemapnum,terms,normx,
```

This example sets the curve fit type for line map three in frame one to a General curve fit and then sets the curve fit sub-functions to produce a function of the form $y = c_1 + c_2 * sin(x) + c_3 * cos(x)$.

SOURCE gplinemap.src

SEE ALSO **gpSetCurveFitType**, **gpSetCurveFitPoints**, **gpSetCurveFitDependentVariable**

gpSetGeometryColor

PURPOSE Sets the color of a geometry object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetGeometryColor**(*pgp*, *framenum*, *name*, *color*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a geometry object.
color	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

You may call **gpSetGeometryColor** to set the color of a geometry created with any of the following commands:

gpAddCircle

gpAddEllipse

gpAddRectangle

gpAddSquare

EXAMPLE

```
framenum = 1;
xstart = 0;
ystart = 0;
xradius = 1.5;
coord = 2;
ret = gpAddCircle(&gp,framenum,"circ1",xstart,ystart,xradius,
                  coord);
ret = gpSetGeometryColor(&gp,framenum,"circ1","green");
```

This example adds a circle of radius 1.5 to frame one, centered on the point (0,0). It then sets the color of the circle to green.

SOURCE gpgeometry.src

SEE ALSO **gpSetGeometryLineThickness**, **gpSetGeometryLinePattern**, **gpFillGeometry**

gpSetGeometryFillColor

PURPOSE Sets the fill color of a geometry object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetGeometryFillColor**(*pgp*, *framenum*, *name*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a geometry object.

	fillcolor	string, name of color. – or – scalar, index of color in color list.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

By default, geometries are not filled. If you call **gpFillGeometry** to specify that a geometry is to be filled, the default fill color is white.

You may call **gpSetGeometryFillColor** to set the fill color of a geometry created with any of the following commands:

gpAddCircle
gpAddEllipse
gpAddRectangle
gpAddSquare

EXAMPLE

```
framenum = 1;
xstart = 0;
ystart = 0;
xradius = 1.5;
coord = 2;
fill = 1;
ret = gpAddCircle(&gp,framenum,"circ1",xstart,ystart,xradius,
                  coord);
ret = gpFillGeometry(&gp,framenum,"circ1",fill);
ret = gpSetGeometryFillColor(&gp,framenum,"circ1","red");
```

This example adds a circle of radius 1.5 to frame one, centered on the point (0,0). It then specifies that the circle is to be filled and sets the fill color to red.

SOURCE

gpgeometry.src

SEE ALSO

gpFillGeometry, **gpSetGeometryColor**

gpSetGeometryLinePattern

PURPOSE Sets the line pattern of a geometry object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetGeometryLinePattern**(*pgp*, *framenum*, *name*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a geometry object.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, geometry objects are drawn using solid lines. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

You may call **gpSetGeometryLinePattern** to set the line pattern of a geometry created with any of the following commands:

gpAddCircle
gpAddEllipse

gpAddRectangle

gpAddSquare

EXAMPLE

```
framenum = 1;
xstart = 0;
ystart = 0;
xradius = 1.5;
coord = 2;
ret = gpAddCircle(&gp,framenum,"circ1",xstart,ystart,xradius,
                  coord);
ret = gpSetGeometryLinePattern(&gp,framenum,"circ1","dashed");
```

This example adds a circle of radius 1.5 to frame one, centered on the point (0,0). It then sets the line pattern of the circle to 'dashed'.

SOURCE gpgeometry.src

SEE ALSO **gpSetGeometryLinePatternLength**, **gpSetGeometryColor**, **gpSetGeometryLineThickness**

gpSetGeometryLinePatternLength

PURPOSE Sets the line pattern length of a geometry object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetGeometryLinePatternLength**(*pgp*, *framenum*, *name*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a geometry object.

	length	scalar, length of one cycle of the line pattern as a percentage of the frame height.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is ignored if the line pattern is set to solid. For non-solid lines, the default pattern length is 2.

You may call **gpSetGeometryLinePatternLength** to set the line pattern length of a geometry created with any of the following commands:

gpAddCircle
gpAddEllipse
gpAddRectangle
gpAddSquare

EXAMPLE
```
framenum = 1;
xstart = 0;
ystart = 0;
xradius = 1.5;
coord = 2;
plength = 5;
ret = gpAddCircle(&gp,framenum,"circ1",xstart,ystart,xradius,
                  coord);
ret = gpSetGeometryLinePattern(&gp,framenum,"circ1","dashed");
ret = gpSetGeometryLinePatternLength(&gp,framenum,"circ1",
                  plength);
```

This example adds a circle of radius 1.5 to frame one, centered on the point (0,0). It then sets the line pattern of the circle to 'dashed', and the line pattern length to 5.

SOURCE `gpgeometry.src`

SEE ALSO **gpSetGeometryLinePattern**, **gpSetGeometryLineColor**, **gpSetGeometryLineThickness**

gpSetGeometryLineThickness

PURPOSE Sets the line thickness of a geometry object.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetGeometryLineThickness**(*pgp*, *framenum*, *name*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a geometry object.
thickness	scalar, line thickness of the geometry object as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, geometry objects have a line thickness of 0.1.

You may call **gpSetGeometryLineThickness** to set the line thickness of a geometry created with any of the following commands:

gpAddCircle

gpAddEllipse

gpAddRectangle

gpAddSquare

EXAMPLE

```
framenum = 1;
xstart = 0;
ystart = 0;
xradius = 1.5;
coord = 2;
```

```
thickness = 0.4;
ret = gpAddCircle(&gp,framenum,"circ1",xstart,ystart,xradius,
                  coord);
ret = gpSetGeometryLineThickness(&gp,framenum,"circ1",
                                 thickness);
```

This example adds a circle of radius 1.5 to frame one, centered on the point (0,0). It then sets the line thickness of the circle to 0.4.

SOURCE gpgeometry.src

SEE ALSO **gpSetGeometryColor**, **gpSetGeometryLinePattern**, **gpFillGeometry**

gpSetGeometryNumSides

PURPOSE Sets the number of sides approximating a circular or elliptical geometry object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetGeometryNumSides**(*pgp*, *framenum*, *name*, *nsides*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a geometry object.
nsides	scalar, number of sides approximating the circular or elliptical geometry object.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, circular and elliptical geometry objects are approximated by 72 sides.

To create regular polygons with *nsides* sides, call this function after **gpAddCircle**. To create non-regular polygons with *nsides* sides, call this function after **gpAddEllipse**.

EXAMPLE

```
framenum = 1;
xstart = 0;
ystart = 0;
xradius = 1.5;
coord = 2;
nsides = 8;
ret = gpAddCircle(&gp,framenum,"oct1",xstart,ystart,xradius,
                  coord);
ret = gpSetGeometryNumSides(&gp,framenum,"oct1",nsides);
```

This example adds an octogon of radius 1.5 to frame one, centered on the point (0,0).

SOURCE gpgeometry.src

SEE ALSO **gpAddCircle**, **gpAddEllipse**, **gpSetGeometryColor**

gpSetGridAreaFillColor

PURPOSE Sets the fill color for the grid area in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetGridAreaFillColor**(*pgp*, *framenum*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fillcolor	string, name of fill color. – or – scalar, index of color in color list.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the grid area is filled for 3D Cartesian and 3D Contour plots and unfilled for all other plot types. The default fill color is white. Because grid area light source shading is used by default in 3D Cartesian and 3D Contour plots, the grid area appears as gray. To deactivate grid area light source shading, call **gpUseGridAreaLighting**.

EXAMPLE
```
framenum = 1;
fill = 1;
ret = gpFillGridArea(&gp,framenum,fill);
ret = gpSetGridAreaFillColor(&gp,framenum,"custom2");
```

SOURCE gp.src

SEE ALSO **gpFillGridArea**, **gpUseGridAreaLighting**, **gpPositionGrid**

gpSetIPlaneRange

PURPOSE Sets the range of I-planes to display in I, IJ, IK or IJK plane surface plotting.

LIBRARY gaussplot

FORMAT *ret* = **gpSetIPlaneRange**(*pgp*, *framenum*, *fieldzonenum*, *min*, *max*, *skipnum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
min	scalar, first I-plane to display.
max	scalar, last I-plane to display.
skipnum	scalar, show only one in every *skipnum* I-planes.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command applies only to 3D Cartesian and 3D Contour volume plots for which I, IJ, IK or IJK plane surface plotting has been selected with **gpSetSurfacesToPlot**. See that command for more information on these surface plotting options.

Assuming that the X, Y and Z-axis variables used in the graph are L × N × P, then there are P I-planes in the graph. Thus, *max* must be ≤ P. By default, *min* is set to 1, *max* is set to P and *skip* is set to 1 (i.e., all I-planes between 1 and P are displayed).

EXAMPLE
```
framenum = 1;
fieldzonenum = 2;
iplanes = 3;
min = 2;
max = 10;
skip = 2;
ret = gpSetSurfacesToPlot(&gp,framenum,fieldzonenum,iplanes);
ret = gpSetIPlaneRange(&gp,framenum,fieldzonenum,min,max,skip);
```

This example specifies that all of the even I-planes beginning with 2 and ending with 10 be displayed in the frame.

SOURCE `gpfieldzone.src`

SEE ALSO **gpSetSurfacesToPlot**, **gpSetJPlaneRange**, **gpSetKPlaneRange**

gpSetJPlaneRange

PURPOSE Sets the range of J-planes to display in J, JK, IJ or IJK plane surface plotting.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetJPlaneRange**(*pgp*, *framenum*, *fieldzonenum*, *min*, *max*, *skipnum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
min	scalar, first J-plane to display.
max	scalar, last J-plane to display.
skipnum	scalar, show only one in every *skipnum* J-planes.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command applies only to 3D Cartesian and 3D Contour volume plots for which J, JK, IJ or IJK plane surface plotting has been selected with **gpSetSurfacesToPlot**. See that command for more information on these surface plotting options.

Assuming that the X, Y and Z-axis variables used in the graph are L×N×P, then there are N J-planes in the graph. Thus, *max* must be ≤ *N*. By default, *min*, *max* and *skip* are all set to 1 (i.e., only the first J-plane is displayed).

EXAMPLE

```
framenum = 1;
fieldzonenum = 2;
jplanes = 4;
min = 3;
max = 7;
skip = 1;
ret = gpSetSurfacesToPlot(&gp,framenum,fieldzonenum,jplanes);
ret = gpSetJPlaneRange(&gp,framenum,fieldzonenum,min,max,skip);
```

This example specifies that all of the J-planes beginning with 3 and ending with 7 be displayed in the frame.

SOURCE gpfieldzone.src

SEE ALSO **gpSetSurfacesToPlot**, **gpSetIPlaneRange**, **gpSetKPlaneRange**

gpSetKPlaneRange

PURPOSE Sets the range of K-planes to display in K, IK, JK or IJK plane surface plotting.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetKPlaneRange**(*pgp*, *framenum*, *fieldzonenum*, *min*, *max*, *skipnum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
min	scalar, first K-plane to display.
max	scalar, last K-plane to display.
skipnum	scalar, show only one in every *skipnum* K-planes.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command applies only to 3D Cartesian and 3D Contour volume plots for which K, IK, JK or IJK plane surface plotting has been selected with **gpSetSurfacesToPlot**. See that command for more information on these surface plotting options.

Assuming that the X, Y and Z-axis variables used in the graph are L×N×P, then there are L K-planes in the graph. Thus, *max* must be ≤ *L*. By default, *min*, *max* and *skip* are all set to 1 (i.e., only the first K-plane is displayed).

EXAMPLE

```
framenum = 1;
fieldzonenum = 2;
kplanes = 5;
min = 2;
max = 8;
skip = 1;
```

```
ret = gpSetSurfacesToPlot(&gp,framenum,fieldzonenum,kplanes);
ret = gpSetKPlaneRange(&gp,framenum,fieldzonenum,min,max,skip);
```

This example specifies that all of the K-planes beginning with 2 and ending with 8 be displayed in the frame.

SOURCE gpfieldzone.src

SEE ALSO **gpSetSurfacesToPlot**, **gpSetIPlaneRange**, **gpSetJPlaneRange**

gpSetLightingEffect

PURPOSE Sets the lighting effect to use in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetLightingEffect**(*pgp*, *framenum*, *fieldzonenum*, *effect*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
effect	scalar, one of the following: **1** paneled lighting effect. **2** gouraud lighting effect.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, a paneled lighting effect is used. This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
fieldzonenum = 2;
```

```
effect = 2;
ret = gpSetLightingEffect(&gp,framenum,fieldzonenum,effect);
```

SOURCE gpfieldzone.src

SEE ALSO **gpUseLightingEffect**, **gpUseFieldZoneContourLighting**, **gpUseFieldZoneShadeLighting**

gpSetLineColor

PURPOSE Sets the color of the line(s) in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineColor**(*pgp*, *framenum*, *linemapnum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
linecolor	string, name of line color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
linemapnum = 3;
ret = gpSetLineColor(&gp,framenum,linemapnum,"red");
```

SOURCE gplinemap.src

SEE ALSO **gpXYLinePlotCreate**, **gpAddXYLineFrame**, **gpSetLineThickness**

gpSetLineLegendBoxColor

PURPOSE Sets the outline and background colors of the box surrounding a line legend.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetLineLegendBoxColor**(*pgp*, *framenum*, *linecolor*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linecolor	string, name of color to use for box outline. – or – scalar, index of color in color list.
fillcolor	string, name of color to use for box background. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetLineLegendBoxColor** must be preceded by a call to **gpAddLineLegend**. If the line legend box type is set to none, then these colors are not used. If the line legend box type is set to hollow, then *fillcolor* is not used, since a hollow box has no background. By default, a line legend has a hollow box, which is surrounded by a black line. To change the box type, call **gpSetLineLegendBoxType**.

See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
linecolor = "red";
```

```
fillcolor = "";

ret = gpAddLineLegend(&gp,framenum);
ret = gpSetLineLegendBoxColor(&gp,framenum,linecolor,fillcolor);
```

In this example, *fillcolor* is set to a null string because the box type is hollow and thus has no background.

SOURCE gplinemap.src

SEE ALSO **gpAddLineLegend**, **gpSetLineLegendBoxStyle**

gpSetLineLegendBoxStyle

PURPOSE Sets the style of the box surrounding a line legend.

LIBRARY gausssplot

FORMAT *ret* = **gpSetLineLegendBoxStyle**(*pgp*, *framenum*, *boxtype*, *thickness*, *linecolor*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
boxtype	scalar, one of the following: **1** no box. **2** hollow box (no background). **3** filled box.
thickness	scalar, thickness of the box outline as a percentage of the frame height.
linecolor	string, name of color to use for box outline. – or – scalar, index of color in color list.

	fillcolor	string, name of color to use for box background. – or – scalar, index of color in color list.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is a convenience function that allows you to set several parameters at one time. See **gpSetLineLegendBoxType**, **gpSetLineLegendBoxThickness**, and **gpSetLineLegendBoxColor** to set the box style parameters individually.

A call to **gpSetLineLegendBoxStyle** must be preceded by a call to **gpAddLineLegend**. If *boxtype* is specified as hollow, then *fillcolor* is not used, since a hollow box has no background. If it is specified as none, then *linecolor*, *fillcolor* and *thickness* are not used. By default, a line legend has a hollow box, which is surrounded by a black line that is 0.1% of the frame height.

See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
boxtype = 3;
thickness = 0.4;
linecolor = "red";
fillcolor = "white";

ret = gpAddLineLegend(&gp,framenum);
ret = gpSetLineLegendBoxStyle(&gp,framenum,boxtype,
                                    thickness,linecolor,fillcolor);
```

SOURCE gplinemap.src

SEE ALSO **gpAddLineLegend**, **gpSetLineLegendBoxColor**, **gpSetLineLegendBoxThickness**, **gpSetLineLegendBoxType**

gpSetLineLegendBoxThickness

PURPOSE Sets the thickness of the outline of the box surrounding a line legend.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetLineLegendBoxThickness**(*pgp*, *framenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
thickness	scalar, thickness of the box outline as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetLineLegendBoxThickness** must be preceded by a call to **gpAddLineLegend**. If the type of the line legend box is set to none, then *thickness* is not used. By default, a line legend has a hollow box (i.e., a box with no background), with an outline that is 0.1% of the frame height.

EXAMPLE

```
framenum = 1;
thickness = 0.4;

ret = gpAddLineLegend(&gp,framenum);
ret = gpSetLineLegendBoxThickness(&gp,framenum,thickness);
```

SOURCE `gplinemap.src`

SEE ALSO **gpAddLineLegend**, **gpSetLineLegendBoxStyle**

gpSetLineLegendBoxType

PURPOSE Sets the type of the box surrounding a line legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineLegendBoxType**(*pgp*, *framenum*, *boxtype*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
boxtype	scalar, one of the following:

1 no box.
2 hollow box (no background).
3 filled box.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetLineLegendBoxType** must be preceded by a call to **gpAddLineLegend**. By default, a line legend has a hollow box (i.e., a box with no background).

EXAMPLE

```
framenum = 1;
boxtype = 3;

ret = gpAddLineLegend(&gp,framenum);
ret = gpSetLineLegendBoxType(&gp,framenum,boxtype);
```

SOURCE gplinemap.src

SEE ALSO **gpAddLineLegend**, **gpSetLineLegendBoxStyle**

gpSetLineLegendFont

PURPOSE Sets the font of the text in a line legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineLegendFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetLineLegendFont** must be preceded by a call to **gpAddLineLegend**. By default, the text in a line legend is written in a helvetica bold font (*fonttype* = "helvbold"). The default font height is 3% of the frame height. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
height = 18;
heightunit = 2;

ret = gpAddLineLegend(&gp,framenum);
ret = gpSetLineLegendFont(&gp,framenum,"timesitalic",height,heightunit
```

SOURCE gplinemap.src

SEE ALSO **gpAddLineLegend**, **gpSetLineLegendLineSpacing**, **gpSetLineLegendMargin**

gpSetLineLegendLineSpacing

PURPOSE Sets the spacing between the lines of text in a line legend.

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineLegendLineSpacing**(*pgp*, *framenum*, *linespacing*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linespacing	scalar, the spacing between lines of text in line legend.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetLineLegendLineSpacing** must be preceded by a call to **gpAddLineLegend**. By default, the line spacing is set to 1.

EXAMPLE

```
framenum = 1;
linespacing = 1.25;

ret = gpAddLineLegend(&gp,framenum);
ret = gpSetLineLegendLineSpacing(&gp,framenum,linespacing);
```

SOURCE gplinemap.src

SEE ALSO **gpAddLineLegend**, **gpSetLineLegendFont**, **gpSetLineLegendMargin**

gpSetLineLegendMargin

PURPOSE Sets the size of the margin between the line legend text and the surrounding box.

LIBRARY gaussplot

FORMAT *ret* **= gpSetLineLegendMargin(***pgp***,** *framenum***,** *margin***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
margin	scalar, the size of the margin between the line legend text and the surrounding box as a percentage of the legend box width.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetLineLegendMargin** must be preceded by a call to **gpAddLineLegend**. By default, the margin is set to 10.

EXAMPLE
```
framenum = 1;
margin = 20;

ret = gpAddLineLegend(&gp,framenum);
ret = gpSetLineLegendMargin(&gp,framenum,margin);
```

SOURCE gplinemap.src

SEE ALSO **gpAddLineLegend**, **gpSetLineLegendLineSpacing**

gpSetLineMapAxes

PURPOSE Sets the X and Y axes to be used for the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineMapAxes(***pgp*, *framenum*, *linemapnum*, *xaxisnum*, *yaxisnum***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
xaxisnum	scalar, 1 ≤ xaxisnum ≤ 5, X-axis number.
yaxisnum	scalar, 1 ≤ yaxisnum ≤ 5, Y-axis number.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. By default, only one X-axis and one Y-axis are used. To display additional X and/or Y axes, call **gpSetLineMapAxes** to link one or more line maps to X and/or Y axes numbered greater than one. Axes that are added in this way are displayed by default. Call **gpShowXAxis** or **gpShowYAxis** to deactivate display of these axes.

Calling **gpSetLineMapAxes** will create the specified line map(s) if they do not already exist, associating the first variable in the first zone of the data file with the X-axis and the second variable in the first zone of the data file with the Y-axis. It will also create the specified frame if it does not yet exist. However, if the program calls **gpPlotCreate** to initialize the plot control structure, then there is no plot type associated with the graph. Thus **gpAddXYLineFrame**, and/or **gpAddBarFrame** should be called before **gpSetLineMapAxes** to ensure that the frame defaults are set correctly for the plot type.

Call **gpSetLineMaps** to specify multiple line maps in a frame or different variables for the X and Y axes. Call **gpSetLineMapZone** to specify a different zone for one or more line maps.

EXAMPLE
```
framenum = 3;
linemapnum = { 2, 3 };
xaxisnum = 1;
yaxisnum = 2;
ret = gpSetLineMapAxes(&gp,framenum,linemapnum,xaxisnum,yaxisnum);
```

This example specifies that line maps two and three should be plotted along the first X-axis and a second Y-axis.

SOURCE gplinemap.src

SEE ALSO **gpSetLineMaps**, **gpSetLineMapZone**

gpSetLineMapNames

PURPOSE Sets names for all of the line maps in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineMapNames**(*pgp*, *framenum*, *names*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
names	N×1 string array, line map names.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, line maps are given either the same name as the variable associated with the Y (or R) axis in the line map or "V*num*", where *num* is the index of the variable in the data file, if the variable names cannot be accessed from the structure file when the line maps are created. See Section 1.8 for information about structure files.

These names are used to identify the line maps in a line legend. Use **gpSetLineMapNames** to rename the line maps in a frame before adding a line legend.

N must equal the number of line maps in the specified frame.

EXAMPLE
```
framenum = 3;
string names = { "sin(x)", "cos(x)", "log(x)" };
ret = gpSetLineMapNames(&gp,framenum,names);
```

SOURCE gplinemap.src

SEE ALSO **gpSetLineMaps**, **gpAddLineLegend**

gpSetLineMaps

PURPOSE Sets all of the line maps for a specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetLineMaps**(*pgp*, *framenum*, *linemaps*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemaps	N×2 matrix, in which the first column contains the indices of the variables to be associated with the X-axis (or Theta-axis) for each line map, and the second column contains the indices of the variables to be associated with the Y-axis or (R-axis). - or - N×2 string array, in which the first column contains the names of the variables to be associated with the X-axis (or Theta-axis) for each line map, and the second column contains the names of the variables to be associated with the Y-axis or (R-axis).

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, XY Line and Bar plots contain one line map, in which the first variable in the data file is associated with the X-axis and the second variable with the Y-axis. Polar Line plots also contain one line map by default; for Polar plots, the first variable in the data file is associated with the Theta-axis and the second variable with the R-axis.

A call to **gpSetLineMaps** will create *N* line maps in the specified frame. If the specified frame does not yet exist, the frame will be created. However, if the program calls **gpPlotCreate** to initialize the plot control structure, then there is no plot type associated with the graph. Thus **gpAddXYLineFrame**, **gpAddBarFrame**, and/or **gpAddPolarLineFrame** should be called before **gpSetLineMaps** to ensure that the frame defaults are set correctly for the plot type.

If you input a string array containing variable names for *linemaps*, the variable names must match the names in the `.fsr` structure file associated with the data file (see Section 1.8 for more information). If the data file has not yet been created, or the associated `.fsr` structure file cannot be accessed, you must use variable indices instead of variable names.

If the specified frame has already been created and you want the frame to contain only one line map, in which the first variable in the data file is associated with the X or Theta-axis and the second variable with the Y or R-axis, then there is no need to call **gpSetLineMaps** for that frame.

If you call one of the **gpMake***PlotType***Plot** commands to plot the graph, then the line maps will be set automatically, and you do not need to call **gpSetLineMaps**. See one of the following commands for more information:

gpMakeBarPlot
gpMakePolarLinePlot
gpMakeXYLinePlot

EXAMPLE

```
framenum = 3;
linemaps = { 1 4, 1 5, 2 5, 3 6 };
ret = gpSetLineMaps(&gp,framenum,linemaps);
```

This example creates four line maps in the third frame as follows, assuming that the frame contains an XY Line or Bar plot:

Line Map	X-axis variable	Y-axis variable
1	1	4
2	1	5
3	2	5
4	3	6

SOURCE `gplinemap.src`

SEE ALSO **gpSetLineMapNames**, **gpSetLineMapZone**, **gpSetLineMapAxes**

gpSetLineMapZone

PURPOSE Sets the zone in the data file to be used for the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineMapZone**(*pgp*, *framenum*, *linemapnum*, *zonenum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
zonenum	scalar, zone number.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, XY Line and Bar plots contain one line map, in which the first variable in the first zone of the data file is associated with the X-axis and the second variable in the first zone with the Y-axis. Polar Line plots also contain one line map by default; for Polar Line plots, the first variable in the first zone of the data file is associated with the Theta-axis and the second variable in the first zone with the R-axis.

Call **gpSetLineMaps** to specify multiple line maps in a frame or different variables for the X and Y (or Theta and R) axes. Call **gpSetLineMapZone** to specify a different zone for one or more line maps.

A call to **gpSetLineMapZone** will create the specified line map(s) if they do not already exist, using the above mentioned defaults for the X and Y (or Theta and R) variables. It will also create the specified frame if it does not yet exist. However, if the program calls **gpPlotCreate** to initialize the plot control structure, then there is no plot type associated with the graph. Thus **gpAddXYLineFrame**, **gpAddBarFrame**, and/or **gpAddPolarLineFrame** should be called before **gpSetLineMapZone** to ensure that the frame defaults are set correctly for the plot type.

EXAMPLE
```
framenum = 3;
linemapnum = { 2, 3, 5 };
zonenum = 2;
ret = gpSetLineMapZone(&gp,framenum,linemapnum,zonenum);
```

This example specifies that the variables for line maps two, three, and five in frame three are to be taken from the second zone in the data file.

SOURCE gplinemap.src

SEE ALSO **gpSetLineMaps**, **gpSetLineMapAxes**

gpSetLinePattern

PURPOSE Sets the pattern of the line(s) in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetLinePattern**(*pgp*, *framenum*, *linemapnum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all lines are drawn as solid lines. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
linemapnum = 2;
ret = gpSetLinePattern(&gp,framenum,linemapnum,"dashed");
```

SOURCE gplinemap.src

SEE ALSO **gpSetLinePatternLength**, **gpSetLineColor**, **gpSetLineThickness**

gpSetLinePatternLength

PURPOSE Sets the pattern length of the line(s) in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetLinePatternLength**(*pgp*, *framenum*, *linemapnum*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
length	scalar, length of one cycle of the line pattern as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is ignored if the line pattern is set to solid. For non-solid lines, the default pattern length is 2.

EXAMPLE
```
framenum = 1;
linemapnum = 2;
length = 4;
ret = gpSetLinePattern(&gp,framenum,linemapnum,"dashdot");
ret = gpSetLinePatternLength(&gp,framenum,linemapnum,length);
```

SOURCE gplinemap.src

SEE ALSO **gpSetLinePattern**, **gpSetLineColor**, **gpSetLineThickness**

gpSetLineSegmentColor

PURPOSE Sets the color of a line segment.

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineSegmentColor**(*pgp*, *framenum*, *name*, *color*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a line segment object created by **gpAddLineSegment**.
color	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
xstart = 0;
xend = 1.25;
ystart = 0;
yend = -1.25;
coord = 2;
ret = gpAddLineSegment(&gp,framenum,"line1",xstart,ystart,xend,yend,co
ret = gpSetLineSegmentColor(&gp,framenum,"line1","green");
```

This example adds a line segment to frame one, drawing the line segment from the origin to the point (1.25, −1.25). It then sets the color of the line segment to green.

SOURCE gplinesegment.src

SEE ALSO **gpAddLineSegment**, **gpSetLineSegmentThickness**

gpSetLineSegmentPattern

PURPOSE Sets the pattern of the line segment.

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineSegmentPattern**(*pgp*, *framenum*, *name*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a line segment object created by **gpAddLineSegment**.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all line segments are drawn as solid lines. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
aname = "lineseg1";

ret = gpAddLineSegment(&gp,framenum,aname,xstart,ystart,xend,yend,coor
ret = gpSetLineSegmentPattern(&gp,framenum,aname,"dashdotdot");
```

This example adds a line segment to frame one and then sets the pattern of the line segment to 'dashdotdot'.

SOURCE gplinesegment.src

SEE ALSO **gpAddLineSegment**, **gpSetLineSegmentPatternLength**, **gpSetLineSegmentColor**

gpSetLineSegmentPatternLength

gpSetLineSegmentPatternLength

PURPOSE Sets the length of the line segment pattern.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetLineSegmentPatternLength**(*pgp*, *framenum*, *name*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a line segment object created by **gpAddLineSegment**.
length	scalar, length of one cycle of the line pattern as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is ignored if the line pattern is set to solid. For non-solid lines, the default pattern length is 2.

EXAMPLE

```
framenum = 1;
xstart = 0;
xend = 1.5;
ystart = 0;
yend = 1;
coord = 2;
lsname = "lineseg1";
length = 3;

ret = gpAddLineSegment(&gp,framenum,lsname,xstart,ystart,xend,yend,coo
ret = gpSetLineSegmentPattern(&gp,framenum,lsname,"dashdotdot");
ret = gpSetLineSegmentPatternLength(&gp,framenum,lsname,length);
```

This example adds a line segment to frame one, sets the pattern of the line segment to ‘dashdotdot’, and then sets the length of one cycle of the pattern to 3% of the frame height.

SOURCE gplinesegment.src

SEE ALSO **gpAddLineSegment**, **gpSetLineSegmentPattern**, **gpSetLineSegmentColor**

gpSetLineSegmentThickness

PURPOSE Sets the thickness of a line segment.

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineSegmentThickness**(*pgp*, *framenum*, *name*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a line segment object created by **gpAddLineSegment**.
thickness	scalar, thickness of the line segment as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, line segments have a thickness of 0.1.

EXAMPLE

```
framenum = 1;
xstart = 0;
xend = 1;
ystart = 0.25;
yend = 1.75;
coord = 2;
lname = "line1";
thickness = 0.4;
```

```
ret = gpAddLineSegment(&gp,framenum,lname,xstart,ystart,xend,yend,2);
ret = gpSetLineSegmentThickness(&gp,framenum,lname,thickness);
```

This example adds a line segment to frame one and then sets the thickness of the line segment to 0.4.

SOURCE gplinesegment.src

SEE ALSO **gpAddLineSegment**, **gpSetLineSegmentColor**

gpSetLineThickness

PURPOSE Sets the thickness of the line(s) in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetLineThickness**(*pgp*, *framenum*, *linemapnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
thickness	scalar, thickness of the line(s) as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all lines have a thickness of 0.1.

EXAMPLE

```
framenum = 3;
linemapnum = 2;
thickness = 0.4;
ret = gpSetLineThickness(&gp,framenum,linemapnum,thickness);
```

SOURCE gplinemap.src

SEE ALSO **gpSetLineColor**, **gpSetLinePattern**

gpSetMacroFile

PURPOSE Sets the name of the macro file to be created.

LIBRARY gaussplot

FORMAT **gpSetMacroFile**(*pgp*, *macrofile*);

INPUT

pgp	pointer to a plot control structure.
macrofile	string, name of macro file.

REMARKS The **GAUSSplot** plotting commands (see Section 2.1.16) take the information in a plot control structure and write it out to a **GAUSSplot** macro file, which is then run in the **GAUSSplot** GUI to create the graph.

If **gpSetMacroFile** is not called, a temporary name will be generated, and the file will be placed on the **GAUSS** temporary file path. Edit gauss.cfg to change this path. To delete the temporary file after running your program, call **gpCleanUp**.

EXAMPLE
```
gpSetMacroFile(&gp,"plot.mcr");
```

SOURCE gp.src

SEE ALSO **gpSetDataFile**

gpSetMagnification

PURPOSE Resizes a plot.

LIBRARY gaussplot

FORMAT *ret* = **gpSetMagnification**(*pgp*, *framenum*, *magnification*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
magnification	scalar, plot magnification.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS If *magnification* is set to 1, then the plot data will fill the whole frame. Setting *magnification* > 1 enlarges the plot, while setting *magnification* < 1 reduces the size of the plot.

By default, Polar Line plots are sized to fit all data points, axes, text, arrows and other geometric objects to the frame (see **gpSetPlotFit**), and thus *magnification* is less than 1. All other plot types have a magnification of 1 by default.

If you call both **gpSetMagnification** and **gpSetPlotFit**, the last one called will override.

EXAMPLE
```
framenum = 1;
magnification = 1.25;
ret = gpSetMagnification(&gp,framenum,magnification);
```

SOURCE gp.src

SEE ALSO **gpSetPlotFit**, **gpSetTranslation**, **gpSetFrameStyle**

gpSetMeshColor

PURPOSE Sets the color of the mesh layer in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetMeshColor**(*pgp*, *framenum*, *fieldzonenum*, *meshcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
meshcolor	string, name of mesh color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Field zone numbers correspond to the number of the associated zone in the data file. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
fieldzonenum = 2;
ret = gpSetMeshColor(&gp,framenum,fieldzonenum,"purple");
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpShowMeshLayer**, **gpShowFieldZoneMeshLayer**, **gpSetMeshLinePattern**

gpSetMeshLinePattern

PURPOSE Sets the line pattern of the mesh in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetMeshLinePattern**(*pgp*, *framenum*, *fieldzonenum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, mesh is drawn with solid lines. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
linemapnum = 2;
ret = gpSetMeshLinePattern(&gp,framenum,linemapnum,"dashdotdot");
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpSetMeshLinePatternLength**, **gpSetMeshColor**, **gpSetMeshLineThickness**, **gpSetMeshType**

gpSetMeshLinePatternLength

PURPOSE Sets the length of the mesh line pattern in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetMeshLinePatternLength**(*pgp*, *framenum*, *fieldzonenum*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
length	scalar, length of one cycle of the line pattern as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS This command is ignored if the line pattern is set to solid. For non-solid lines, the default pattern length is 2.

EXAMPLE
```
framenum = 1;
fieldzonenum = 2;
length = 4;
ret = gpSetMeshLinePattern(&gp,framenum,fieldzonenum,"dashdot");
ret = gpSetMeshLinePatternLength(&gp,framenum,fieldzonenum,length);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetMeshLinePattern**, **gpSetMeshColor**, **gpSetMeshLineThickness**, **gpSetMeshType**

gpSetMeshLineThickness

PURPOSE Sets the line thickness of the mesh layer in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetMeshLineThickness**(*pgp*, *framenum*, *fieldzonenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
thickness	scalar, line thickness of the mesh layer as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the mesh layer in each field zone has a line thickness of 0.1. Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE

```
framenum = 3;
fieldzonenum = 2;
thickness = 0.4;
ret = gpSetMeshLineThickness(&gp,framenum,fieldzonenum,thickness);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetMeshColor**, **gpSetMeshLinePattern**, **gpSetMeshType**

gpSetMeshType

PURPOSE Sets the type of the mesh in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetMeshType**(*pgp*, *framenum*, *fieldzonenum*, *type*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
type	scalar, one of the following:

1 Wire frame - mesh is drawn under all other field layers except scatter and vector and thus cannot be seen if the contour or shade layers are active.

2 Overlay - mesh is drawn over other field zone layers.

3 Hidden line - mesh is drawn over other field zone layers, and lines that are hidden behind other surfaces are removed.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, overlay mesh is used. Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE

```
framenum = 2;
fieldzonenum = 2;
meshtype = 3;
ret = gpSetMeshType(&gp,framenum,fieldzonenum,meshtype);
```

This example sets the mesh type for the second field zone in the second frame to hidden line.

SOURCE gpfieldzone.src

SEE ALSO **gpSetMeshColor**, **gpSetMeshLinePattern**, **gpSetMeshLineThickness**

gpSetPaperOrientation

PURPOSE Sets the orientation of the page on which the graph is displayed.

LIBRARY gaussplot

FORMAT *ret* = **gpSetPaperOrientation**(*pgp*, *orientation*);

INPUT

pgp	pointer to a plot control structure.
orientation	scalar, one of the following:
	1 portrait.
	2 landscape.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the page is landscape.

EXAMPLE
```
orientation = 1;
ret = gpSetPaperOrientation(&gp,orientation);
```

SOURCE gp.src

SEE ALSO **gpSetPaperSize**, **gpPrint**

gpSetPaperSize

PURPOSE Sets the size of the page on which the graph is displayed.

LIBRARY gaussplot

FORMAT *ret* = **gpSetPaperSize**(*pgp*, *orientation*);

INPUT

pgp	pointer to a plot control structure.
size	scalar, one of the following:

1 letter (11 × 8.5).
2 double (17 × 11).
3 A4 (11.6929 × 8.26772).
4 A3 (16.5354 × 11.6929).
5 Custom1 (14 × 8.5).
6 Custom2 (10 × 8).

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the page size is letter. All dimensions are given in inches and are *width* × *height*. The dimensions assume that the page is landscape. If the page orientation is changed by a call to **gpSetPaperOrientation**, then the dimensions will be reversed.

EXAMPLE
```
size = 3;
ret = gpSetPaperSize(&gp,size);
```

SOURCE gp.src

SEE ALSO **gpSetPaperOrientation**, **gpPrint**

gpSetPlotData

PURPOSE Creates a **GAUSSplot** data structure for a data file with only one zone containing K variables that are N×1 and writes the data to the structure.

LIBRARY gaussplot

FORMAT *gdat* = **gpSetPlotData**(*x*, *vnames*);

INPUT

x	N×K matrix containing K variables that are each N×1 and are horizontally concatenated.
vnames	K×1 string array, variable names.

OUTPUT

gdat	a **GAUSSplot** data structure containing *x* and *vnames*.

REMARKS Use **gpSetPlotData** to write data to a **gpData** structure for a data file with only one zone containing K variables that are N×1.

EXAMPLE

```
library gaussplot;
#include gp.sdf

p = pi;
x = seqa(0,p/4,15);
y = sin(x);

struct gpData gdat;
string vnames = { "X", "Y" };
gdat = gpSetPlotData(x~y,vnames);
ret = gpWritePlotData(&gdat,"mydata.plt");
```

SOURCE gpwritedata.src

SEE ALSO **gpWritePlotData**, **gpInitPlotData**, **gpAddZone**

gpSetPlotFit

PURPOSE Controls how a plot is sized to the frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetPlotFit**(*pgp*, *framenum*, *fit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
fit	scalar, one of the following:

1 resize to fit all data points to the frame.

2 resize to fit all data points, axes, text, arrows and other geometric objects to the frame.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, Polar Line plots are sized to fit all data points, axes, text, arrows and other geometric objects to the frame, while all other plot types are sized to fit only the data points to the frame.

Calling **gpSetPlotFit** after **gpSet3DViewDistance** or **gpSet3DViewerPosition** will override the 3D view specifications.

If you call both **gpSetMagnification** and **gpSetPlotFit**, the last one called will override.

EXAMPLE
```
framenum = 1;
fit = 2;
ret = gpSetPlotFit(&gp,framenum,fit);
```

SOURCE gp.src

SEE ALSO **gpSetMagnification**, **gpSetFrameStyle**, **gpSetWorkspaceView**

gpSetPolarAxesTitles

PURPOSE Sets the titles for the polar axes in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetPolarAxesTitles**(*pgp*, *framenum*, *thetatitle*, *rtitle*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
thetatitle	string, title for the Theta-axis.
rtitle	string, title for the R-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, each axis is titled with the name(s) of the variable(s) that are associated with that axis (the variables are separated by a comma and a space if there are more than one).

This command is a convenience function that allows you to set the titles for both axes at one time. Use **gpSetThetaAxisTitle** or **gpSetRAxisTitle** to set the title for only one axis.

EXAMPLE
```
framenum = 1;
ret = gpSetPolarAxesTitles(&gp,framenum,"Theta","R");
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisTitle**, **gpSetRAxisTitle**, **gpShowPolarAxes**

gpSetPrintFormat

PURPOSE Sets the print output format for printing on UNIX or Linux.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetPrintFormat**(*pgp*, *printfmt*);

INPUT

pgp	pointer to a plot control structure.
printfmt	scalar, print output format, one of the following: **1** PostScript. **2** HP-GL. **3** HP-GL/2.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The default format for print output on UNIX and Linux is PostScript. We recommend that you always use PostScript output when printing on these platforms.

Calling **gpSetPrintFormat** specifies the print output format used for all subsequent print jobs (specified by the **gpPrint** command) until another call to **gpSetPrintFormat** is encountered.

By default, graphs printed using the PostScript and HP-GL/2 formats are printed in color. Call **gpSetPrintPalette** to print the graph as a monochrome image. Graphs printed using the HP-GL format are always printed as monochrome images; **gpSetPrintPalette** is ignored for HP-GL format print output.

EXAMPLE
```
printfmt = 3;
ret = gpSetSetPrintFormat(&gp,printfmt);
ret = gpPrint(&gp);
```

SOURCE `gp.src`

SEE ALSO **gpPrint**, **gpPrintToFile**, **gpSetPrintPalette**

gpSetPrintPalette

PURPOSE Sets the print/export palette to monochrome or color.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetPrintPalette**(*pgp*, *palette*);

INPUT

pgp	pointer to a plot control structure.
palette	scalar, one of the following:
	1 print/export as monochrome image.
	2 print/export as color image.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all images are printed and exported as color images.

Calling **gpSetPrintPalette** specifies the print/export palette used for all subsequent printing and exporting to file formats for which the palette option is used until another call to **gpSetPrintPalette** is encountered. The palette option is used for exports to the following file formats:

File Format	Export Command
EPS	**gpExportEPS**
HPGL/2	**gpExportHPGL2**
PostScript	**gpExportPS**
TIFF	**gpExportTIFF**
WMF	**gpExportWMF**

EXAMPLE

```
palette = 1;
```

```
area = -2;
ret = gpSetPrintPalette(&gp,palette);
ret = gpExportPS(&gp,area,"graphic.ps");
```

This example sets the print/export palette to monochrome, and then exports the entire workspace to the PostScript graphics file `graphic.ps`.

SOURCE `gp.src`

SEE ALSO **gpSetExportImageWidth**, **gpSetPrintFormat**, **gpPrint**

gpSetRAxisColor

PURPOSE Sets the color of the R-axis line in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetRAxisColor**(*pgp*, *framenum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis lines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
ret = gpSetRAxisColor(&gp,framenum,"green");
```

SOURCE gppolarlineplot.src

SEE ALSO **gpShowPolarAxes**, **gpSetRAxisThickness**

gpSetRAxisLabelColor

PURPOSE Sets the color of the R-axis tick labels in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetRAxisLabelColor**(*pgp*, *framenum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 2;
ret = gpSetRAxisLabelColor(&gp,framenum,"purple");
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetRAxisLabelFont**, **gpSetRAxisLabelOffset**, **gpSetRAxisLabelSkip**

gpSetRAxisLabelFont

PURPOSE Sets the type and size of the font used for the R-axis tick labels in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetRAxisLabelFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points. **3** font height as a percentage of the length of the axis line.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all polar axes tick labels are written in a helvetica font (*fonttype* = "helv") with a font height equal to 3% of the frame height. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
fontheight = 10;
heightunit = 2;
ret = gpSetRAxisLabelFont(&gp,framenum,"times",fontheight,heightunit);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetRAxisLabelColor**, **gpSetRAxisLabelOffset**, **gpSetRAxisLabelSkip**

gpSetRAxisLabelOffset

PURPOSE Sets the offset of the R-axis tick labels from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetRAxisLabelOffset**(*pgp*, *framenum*, *offset*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	offset	scalar, offset of tick labels from axis line as a percentage of the frame height.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels have an offset of 1. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
offset = 2;
ret = gpSetRAxisLabelOffset(&gp,framenum,offset);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetRAxisLabelColor**, **gpSetRAxisLabelFont**, **gpSetRAxisLabelSkip**

gpSetRAxisLabelSkip

PURPOSE Sets an option to show only one in every *skipnum* R-axis tick labels in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetRAxisLabelSkip**(*pgp*, *framenum*, *skipnum*);

INPUT	*pgp*	pointer to a plot control structure.
	framenum	scalar, frame number.
	skipnum	scalar, show only one in every *skipnum* R-axis tick labels.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels are shown (*skipnum* = 1). This command applies ONLY to frames that contain Polar Line Plots.

EXAMPLE
```
framenum = 3;
skipnum = 2;
ret = gpSetRAxisLabelSkip(&gp,framenum,skipnum);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetRAxisLabelColor**, **gpSetRAxisLabelFont**, **gpSetRAxisLabelOffset**

gpSetRAxisRange

PURPOSE Sets the range of the R-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetRAxisRange**(*pgp*, *framenum*, *min*, *max*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
min	scalar, minimum value for the R-axis.
max	scalar, maximum value for the R-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the range of the R-axis is calculated based on the values of the data points. Since radii are plotted along the R-axis, all *r* values should be magnitudes, and therefore, positive. This command applies ONLY to frames that contain Polar Line plots.

To change the number of major and minor tick marks or the distance between them, call **gpSetRAxisTicks**.

EXAMPLE
```
framenum = 1;
min = 5;
max = 15;
ret = gpSetRAxisRange(&gp,framenum,min,max);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetRAxisTicks**

gpSetRAxisThickness

PURPOSE Sets the thickness of the R-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetRAxisThickness**(*pgp*, *framenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
thickness	scalar, thickness of the axis line as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis lines are set to a thickness of 0.4. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
thickness = 0.6;
ret = gpSetRAxisThickness(&gp,framenum,thickness);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetRAxisColor**

gpSetRAxisTicks

PURPOSE Sets the distance between major tick marks and the number of minor tick marks on the R-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetRAxisTicks**(*pgp*, *framenum*, *majorstep*, *minornum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
majorstep	scalar, step between major tick marks in R-axis units.
minornum	scalar, number of minor tick marks between each set of major tick marks.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the step between major tick marks and the number of minor tick marks on the R-axis is calculated based on the range of the axis. To modify the range of the R-axis, call **gpSetRAxisRange**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
majorstep = 2;
minornum = 3;
ret = gpSetRAxisTicks(&gp,framenum,majorstep,minornum);
```

This example sets the distance between major tick marks on the R-axis to 2

units, and the number of minor tick marks between each set of major tick marks to 3. Thus there are 0.5 units from one tick mark to the next.

SOURCE gp.src

SEE ALSO **gpSetRAxisRange**

gpSetRAxisTitle

PURPOSE Sets the title for the R-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetRAxisTitle**(*pgp*, *framenum*, *title*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
title	string, title.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, each axis is titled with the name(s) of the variable(s) that are associated with that axis (the variables are separated by a comma and a space if there are more than one). Use **gpSetPolarAxesTitles** to set the titles for both the Theta and R axes in a frame.

This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
ret = gpSetRAxisTitle(&gp,framenum,"R");
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetPolarAxesTitles**, **gpSetRAxisTitleColor**, **gpSetRAxisTitleFont**, **gpSetRAxisTitleOffset**

gpSetRAxisTitleColor

PURPOSE Sets the color of the R-axis title in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetRAxisTitleColor**(*pgp*, *framenum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis titles are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
ret = gpSetRAxisTitleColor(&gp,framenum,"green");
```

SOURCE `gppolarlineplot.src`

SEE ALSO **gpSetRAxisTitle**, **gpSetRAxisTitleFont**, **gpSetRAxisTitleOffset**

gpSetRAxisTitleFont

PURPOSE Sets the type and size of the font used for the R-axis title in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetRAxisTitleFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points. **3** font height as a percentage of the length of the axis line.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all polar axis tick labels are written in a bold helvetica font (*fonttype* = "helvbold") with a font height equal to 3.6% of the frame height. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
fontheight = 5;
heightunit = 1;
ret = gpSetRAxisTitleFont(&gp,framenum,"timesbold",fontheight,heightun
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetRAxisTitle**, **gpSetRAxisTitleColor**, **gpSetRAxisTitleOffset**

gpSetRAxisTitleOffset

PURPOSE Sets the offset of the R-axis title from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetRAxisTitleOffset**(*pgp*, *framenum*, *offset*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	offset	scalar, offset of title from axis line as a percentage of the height of the frame.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the offset is set to 6. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
offset = 8;
ret = gpSetRAxisTitleOffset(&gp,framenum,offset);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetRAxisTitle**, **gpSetRAxisTitleColor**, **gpSetRAxisTitleFont**

gpSetScatterColor

PURPOSE Sets the color of the scatter symbols in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetScatterColor**(*pgp*, *framenum*, *fieldzonenum*, *scattercolor*);

INPUT	*pgp*	pointer to a plot control structure.
	framenum	scalar, frame number.
	fieldzonenum	scalar or N×1 vector, field zone number(s).
	scattercolor	string, name of scatter color. – or – scalar, index of color in color list.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the scatter layer is turned off. To activate the scatter layer in a frame, call **gpShowScatterLayer**.

Field zone numbers correspond to the number of the associated zone in the data file. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
seton = 1;
fieldzonenum = 3;
ret = gpShowScatterLayer(&gp,framenum,seton);
ret = gpSetScatterColor(&gp,framenum,fieldzonenum,"purple");
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowScatterLayer**, **gpShowFieldZoneScatterLayer**, **gpSetScatterFillMode**, **gpSetScatterShape**

gpSetScatterFillMode

PURPOSE Sets the fill mode of the scatter symbols in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetScatterFillMode**(*pgp*, *framenum*, *fieldzonenum*, *fillmode*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).

fillmode scalar, one of the following:

1 no fill.

2 use line color.

3 use background color.

4 use *fillcolor*.

fillcolor string, name of color.
– or –
scalar, index of color in color list.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the scatter layer is turned off. To activate the scatter layer in a frame, call **gpShowScatterLayer**. When the scatter layer is activated, scatter symbols are not filled by default.

If *fillmode* is not set to 4, then *fillcolor* is ignored.

Frames are numbered in the order that they are created. Field zone numbers correspond to the number of the associated zone in the data file. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
seton = 1;
fieldzonenum = 3;
fillmode = 2;
ret = gpShowScatterLayer(&gp,framenum,seton);
ret = gpSetScatterFillMode(&gp,framenum,fieldzonenum,fillmode,"");
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetScatterColor**, **gpSetScatterLineThickness**, **gpSetScatterShape**

gpSetScatterLineThickness

PURPOSE Sets the line thickness of the scatter symbols in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetScatterLineThickness**(*pgp*, *framenum*, *fieldzonenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
thickness	scalar, line thickness of the scatter symbols as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the scatter layer is turned off. To activate the scatter layer in a frame, call **gpShowScatterLayer**. When the scatter layer is activated, all scatter symbols have a line thickness of 0.1 by default.

Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE

```
framenum = 3;
seton = 1;
fieldzonenum = 2;
thickness = 0.4;
ret = gpShowScatterLayer(&gp,framenum,seton);
ret = gpSetScatterLineThickness(&gp,framenum,fieldzonenum,thickness);
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpSetScatterColor**, **gpSetScatterShape**, **gpSetScatterSize**

gpSetScatterShape

PURPOSE Sets the shape of the scatter symbols in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetScatterShape**(*pgp*, *framenum*, *fieldzonenum*, *shape*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
shape	string, name of symbol shape. – or – scalar, index of symbol shape in symbol shape list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the scatter layer is turned off. To activate the scatter layer in a frame, call **gpShowScatterLayer**. When the scatter layer is activated, the default symbol shape is a square.

The list of available shapes is as follows:

1 Square
2 Del (delta)
3 Grad (gradient)
4 RTri (right triangle)
5 LTri (left triangle)
6 Diamond
7 Circle
8 Point
9 Cube
10 Sphere
11 Octahedron

Scatter shape names are case insensitive in **GAUSSplot**. Use **gpSetScatterShapeASCII** to use an ASCII character for the scatter symbols in a field zone.

EXAMPLE
```
framenum = 1;
seton = 1;
fieldzonenum = 3;
ret = gpShowScatterLayer(&gp,framenum,seton);
ret = gpSetScatterShape(&gp,framenum,fieldzonenum,"diamond");
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetScatterShapeASCII**, **gpSetScatterColor**, **gpSetScatterSize**

gpSetScatterShapeASCII

PURPOSE Sets the shape of the scatter symbols in the specified field zone(s) to an ASCII character.

LIBRARY gaussplot

FORMAT *ret* = **gpSetScatterShapeASCII**(*pgp*, *framenum*, *fieldzonenum*,

fonttype, *character*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
fonttype	scalar, one of the following: **1** Base font, normal ASCII characters. **2** Greek font. **3** Math font.
character	string, single character to use as symbol.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

By default, the scatter layer is turned off. To activate the scatter layer in a frame, call **gpShowScatterLayer**. When the scatter layer is activated, the default symbol shape is a square.

Use **gpSetScatterShape** to use a preset symbol shape.

See Appendix A for a list of all of the available characters in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
seton = 1;
fieldzonenum = 3;
fonttype = 2;
ret = gpShowScatterLayer(&gp,framenum,seton);
ret = gpSetScatterShapeASCII(&gp,framenum,fieldzonenum,fonttype,"p");
```

This example sets the symbol shape for the third field zone in the first frame to the Greek letter 'π'.

SOURCE

gpfieldzone.src

SEE ALSO

gpSetScatterShape, **gpSetScatterColor**, **gpSetScatterSize**

gpSetScatterSize

PURPOSE Sets the size of the scatter symbols in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* **= gpSetScatterSize(***pgp***,** *framenum***,** *fieldzonenum***,** *size***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
size	scalar, size of the scatter symbols as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the scatter layer is turned off. To activate the scatter layer in a frame, call **gpShowScatterLayer**. When the scatter layer is activated, all scatter symbols have a size of 2.5 by default.

Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE
```
framenum = 3;
seton = 1;
fieldzonenum = 2;
size = 1.5;
ret = gpShowScatterLayer(&gp,framenum,seton);
ret = gpSetScatterSize(&gp,framenum,fieldzonenum,size);
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpSetScatterColor**, **gpSetScatterShape**

gpSetShadeColor

PURPOSE Sets the color of the shading in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetShadeColor**(*pgp*, *framenum*, *fieldzonenum*, *shadecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
shadecolor	string, name of shade color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the shade layer is turned off. To activate the shade layer in a frame, call **gpShowShadeLayer**.

Field zone numbers correspond to the number of the associated zone in the data file. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
seton = 1;
fieldzonenum = 1;
ret = gpShowShadeLayer(&gp,framenum,seton);
ret = gpSetShadeColor(&gp,framenum,fieldzonenum,"blue");
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpShowShadeLayer**, **gpShowFieldZoneShadeLayer**, **gpUseFieldZoneShadeLighting**, **gpUseTranslucency**

gpSetSurfacesToPlot

PURPOSE Specifies which surfaces are to be displayed for a 3D Cartesian or 3D Contour volume plot.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetSurfacesToPlot**(*pgp*, *framenum*, *fieldzonenum*, *surfaces*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
surfaces	scalar, one of the following:

1 boundary cell faces
2 exposed cell faces
3 I-Planes
4 J-Planes
5 K-Planes
6 IJ-Planes
7 JK-Planes
8 IK-Planes
9 IJK-Planes
10 every surface

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS The following table explains each of the surface plotting options, assuming that the X, Y and Z-axis variables used in the graph are L×N×P:

Option		Explanation
1	boundary cell faces	Display all surfaces that are on the outside of the data zone
2	exposed cell faces	Identical to boundary cell faces unless data blanking is used. Then the surfaces between blanked and unblanked data values are plotted as well as the outer surfaces
3	I-planes	Display the L×N surfaces in the specified range (1 - P)
4	J-planes	Display the L×P surfaces in the specified range (1 - N)
5	K-planes	Display the N×P surfaces in the specified range (1 - L)
6	IJ-planes	Display the L×N surfaces in the specified range (1 - P) and the L×P surfaces in the specified range (1 - N)
7	JK-planes	Display the L×P surfaces in the specified range (1 - N) and the N×P surfaces in the specified range (1 - L)
8	IK-planes	Display the L×N surfaces in the specified range (1 - P) and the N×P surfaces in the specified range (1 - L)
9	IJK-planes	Display the L×N surfaces in the specified range (1 - P), the L×P surfaces in the specified range (1 - N) and the the N×P surfaces in the specified range (1 - L)
10	Every surface	Display all surfaces

By default, exposed cell face plotting is used for 3D Cartesian and 3D Contour volume plots.

See the TECPLOT® USER'S MANUAL in the file `tpum.pdf` for information on data blanking for the exposed cell face option.

To set the specified ranges for options 3 - 9, call **gpSetIPlaneRange**, **gpSetJPlaneRange** and/or **gpSetKPlaneRange**.

EXAMPLE

```
framenum = 1;
fieldzonenum = 2;
ret = gpSetSurfacesToPlot(&gp,framenum,fieldzonenum,10);
```

SOURCE `gpfieldzone.src`

SEE ALSO **gpSetIPlaneRange**, **gpSetJPlaneRange**, **gpSetKPlaneRange**

gpSetSymbolColor

PURPOSE Sets the color of the symbols in the specified line map(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetSymbolColor**(*pgp*, *framenum*, *linemapnum*, *symbolcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
symbolcolor	string, name of symbol color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the symbol layer is turned off. To activate the symbol layer in a frame, call **gpShowSymbolLayer**.

See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
seton = 1;
linemapnum = 3;
ret = gpShowSymbolLayer(&gp,framenum,seton);
ret = gpSetSymbolColor(&gp,framenum,linemapnum,"purple");
```

SOURCE `gplinemap.src`

SEE ALSO **gpShowSymbolLayer**, **gpShowLineMapSymbols**, **gpSetSymbolShape**

gpSetSymbolFillMode

PURPOSE Sets the fill mode of the symbols in the specified line map(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetSymbolFillMode**(*pgp*, *framenum*, *linemapnum*, *fillmode*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
fillmode	scalar, one of the following: **1** no fill. **2** use line color. **3** use background color. **4** use *fillcolor*.
fillcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the symbol layer is turned off. To activate the symbol layer in a frame, call **gpShowSymbolLayer**. When the symbol layer is activated, symbols are not filled by default.

If *fillmode* is not set to 4, then *fillcolor* is ignored.

See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
```

```
seton = 1;
linemapnum = 3;
fillmode = 3;
ret = gpShowSymbolLayer(&gp,framenum,seton);
ret = gpSetSymbolFillMode(&gp,framenum,linemapnum,fillmode,"");
```

SOURCE gplinemap.src

SEE ALSO **gpSetSymbolColor**, **gpSetSymbolLineThickness**, **gpSetSymbolShape**

gpSetSymbolLineThickness

PURPOSE Sets the line thickness of the symbols in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetSymbolLineThickness(***pgp*, *framenum*, *linemapnum*, *thickness***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
thickness	scalar, line thickness of the symbols as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the symbol layer is turned off. To activate the symbol layer in a frame, call **gpShowSymbolLayer**. When the symbol layer is activated, all symbols have a line thickness of 0.1 by default.

EXAMPLE

```
framenum = 3;
```

```
seton = 1;
linemapnum = 2;
thickness = 0.4;
ret = gpShowSymbolLayer(&gp,framenum,seton);
ret = gpSetSymbolLineThickness(&gp,framenum,linemapnum,thickness);
```

SOURCE gplinemap.src

SEE ALSO **gpSetSymbolColor**, **gpSetSymbolShape**, **gpSetSymbolSize**

gpSetSymbolShape

PURPOSE Sets the shape of the symbols in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetSymbolShape**(*pgp*, *framenum*, *linemapnum*, *shape*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
shape	string, name of symbol shape. – or – scalar, index of symbol shape in symbol shape list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the symbol layer is turned off. To activate the symbol layer in a frame, call **gpShowSymbolLayer**. When the symbol layer is activated, the default symbol shape is a square. The list of available shapes is as follows:

1 Square
2 Del (delta)
3 Grad (gradient)
4 RTri (right triangle)
5 LTri (left triangle)
6 Diamond
7 Circle

Symbol shape names are case insensitive in **GAUSSplot**. Use **gpSetSymbolShapeASCII** to use an ASCII character for the symbols in a line map.

EXAMPLE
```
framenum = 1;
seton = 1;
linemapnum = 3;
ret = gpShowSymbolLayer(&gp,framenum,seton);
ret = gpSetSymbolShape(&gp,framenum,linemapnum,"diamond");
```

SOURCE gplinemap.src

SEE ALSO **gpSetSymbolShapeASCII**, **gpSetSymbolColor**, **gpSetSymbolSize**, **gpSetSymbolSkip**

gpSetSymbolShapeASCII

PURPOSE Sets the shape of the symbols in the specified line map(s) to an ASCII character.

LIBRARY gaussplot

FORMAT *ret* = **gpSetSymbolShapeASCII**(*pgp*, *framenum*, *linemapnum*, *fonttype*, *character*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	linemapnum	scalar or N×1 vector, line map number(s).
	fonttype	scalar, one of the following: **1** Base font, normal ASCII characters. **2** Greek font. **3** Math font.
	character	string, single character to use as symbol.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the symbol layer is turned off. To activate the symbol layer in a frame, call **gpShowSymbolLayer**. When the symbol layer is activated, the default symbol shape is a square.

Use **gpSetSymbolShape** to use a preset symbol shape.

See Appendix A for a list of all of the available characters in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
seton = 1;
linemapnum = 3;
fonttype = 2;
ret = gpShowSymbolLayer(&gp,framenum,seton);
ret = gpSetSymbolShapeASCII(&gp,framenum,linemapnum,fonttype,"p");
```

This example sets the symbol shape for the third line map in the first frame to the Greek letter 'π'.

SOURCE gplinemap.src

SEE ALSO **gpSetSymbolShape**, **gpSetSymbolColor**, **gpSetSymbolSize**, **gpSetSymbolSkip**

gpSetSymbolSize

PURPOSE Sets the size of the symbols in the specified line map(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetSymbolSize**(*pgp*, *framenum*, *linemapnum*, *size*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
size	scalar, size of the symbols as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the symbol layer is turned off. To activate the symbol layer in a frame, call **gpShowSymbolLayer**. When the symbol layer is activated, all symbols have a size of 2.5 by default.

EXAMPLE

```
framenum = 3;
seton = 1;
linemapnum = 2;
size = 1.5;
ret = gpShowSymbolLayer(&gp,framenum,seton);
ret = gpSetSymbolSize(&gp,framenum,linemapnum,size);
```

SOURCE `gplinemap.src`

SEE ALSO **gpSetSymbolColor**, **gpSetSymbolShape**, **gpSetSymbolSkip**

gpSetSymbolSkip

PURPOSE Sets an option to show symbols at only a subset of the data points in the specified line map(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetSymbolSkip**(*pgp*, *framenum*, *linemapnum*, *skipnum*, *skipmode*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
skipnum	scalar, skip.
skipmode	scalar, one of the following:

1 show symbols at every data point.

2 show symbols at only one in every *skipnum* data points.

3 show symbols at the first data point and at each successive data point that is at least *skipnum* percent of the frame height from the previous symbol.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the symbol layer is turned off. To activate the symbol layer in a frame, call **gpShowSymbolLayer**. When the symbol layer is activated, symbols are shown at every data point by default. If *skipmode* is set to 1, then *skipnum* is ignored.

EXAMPLE
```
framenum = 3;
seton = 1;
linemapnum = 2;
skipnum = 5;
```

```
skipmode = 2;
ret = gpShowSymbolLayer(&gp,framenum,seton);
ret = gpSetSymbolSkip(&gp,framenum,linemapnum,skipnum,skipmode);
```

SOURCE gplinemap.src

SEE ALSO **gpSetSymbolColor**, **gpSetSymbolShape**, **gpSetSymbolSize**

gpSetTextObjectAnchorLocation

PURPOSE Sets the anchor location of a text object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTextObjectAnchorLocation**(*pgp*, *framenum*, *name*, *horizontal*, *vertical*);

INPUT

pgp pointer to a plot control structure.

framenum scalar, frame number.

name string, name of a text object created by **gpAddTextObject**.

horizontal scalar, one of the following:

1 anchor the left hand side of the text object to the specified X position.

2 anchor the horizontal midpoint of the text object to the specified X position.

3 anchor the right hand side of the text object to the specified X position.

vertical scalar, one of the following:

1 anchor the bottom edge of the text object to the specified Y position.

2 anchor the vertical midpoint of the text object to the specified Y position.

3 anchor the top edge of the text object to the specified Y position.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetTextObjectAnchorLocation** must be preceded by a call to **gpAddTextObject**.

EXAMPLE
```
framenum = 1;
xpos = 41;
ypos = 91;
coord = 1;
horizontal = 2;
vertical = 2;

ret = gpAddTextObject(&gp,framenum,"text1",xpos,
                          ypos,coord,"3D Contour Plot");
ret = gpSetTextObjectAnchorLocation(&gp,framenum,"text1",
                                        horizontal,vertical);
```

This example adds a text object, moves it to the position (41,91) in frame units (0 - 100), and then centers the text box both vertically and horizontally on that position.

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpRotateTextObject**

gpSetTextObjectBoxColor

PURPOSE Sets the outline and background colors of the box surrounding a text object.

LIBRARY gaussplot

gpSetTextObjectBoxColor

FORMAT *ret* = **gpSetTextObjectBoxColor**(*pgp*, *framenum*, *name*, *linecolor*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
linecolor	string, name of color to use for box outline.. – or – scalar, index of color in color list
fillcolor	string, name of color to use for box background.. – or – scalar, index of color in color list

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetTextObjectBoxColor** must be preceded by a call to **gpAddTextObject**. If the text object box type is set to none, then these colors are not used. If the text object box type is set to hollow, then *fillcolor* is not used, since a hollow box has no background. By default, the box type for a text object is set to none. To change the box type, call **gpSetTextObjectBoxType**.

See Section 2.2.1 for the list of available colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
objectname = "text1";
xpos = 41;
ypos = 91;
coord = 1;
filledbox = 3;
linecolor = "red";
fillcolor = "white";

ret = gpAddTextObject(&gp,framenum,objectname,xpos,
                          ypos,coord,"f(x) = sin(x)");
ret = gpSetTextObjectBoxType(&gp,framenum,objectname,filledbox);
```

```
ret = gpSetTextObjectBoxColor(&gp,framenum,objectname,linecolor,fillco
```

This example adds a text object to frame one, sets the box type of the text object to filled, and sets the box outline color to red and the box fill color to white.

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectBoxStyle**

gpSetTextObjectBoxStyle

PURPOSE Sets the style of the box surrounding a text object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTextObjectBoxStyle**(*pgp*, *framenum*, *name*, *boxtype*, *thickness*, *linecolor*, *fillcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
boxtype	scalar, one of the following: **1** no box. **2** hollow box (no background). **3** filled box.
thickness	scalar, thickness of the box outline as a percentage of the frame height.
linecolor	string, name of color to use for box outline.. – or – scalar, index of color in color list
fillcolor	string, name of color to use for box background.. – or – scalar, index of color in color list

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

This command is a convenience function that allows you to set several parameters at one time. See **gpSetTextObjectBoxType**, **gpSetTextObjectBoxThickness**, and **gpSetTextObjectBoxColor** to set the box style parameters individually.

A call to **gpSetTextObjectBoxStyle** must be preceded by a call to **gpAddTextObject**. If *boxtype* is set to none, then *thickness*, *linecolor*, and *fillcolor* are not used. If *boxtype* is set to hollow, then *fillcolor* is not used, since a hollow box has no background. By default, the box type for a text object is set to none.

See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
objectname = "text1";
xpos = 41;
ypos = 91;
coord = 1;
filledbox = 3;
thickness = 0.4;
linecolor = "red";
fillcolor = "white";

ret = gpAddTextObject(&gp,framenum,objectname,xpos,
                          ypos,coord,"f(x) = sin(x)");
ret = gpSetTextObjectBoxStyle(&gp,framenum,objectname,filledbox,
                                  thickness,linecolor,fillcolor);
```

SOURCE

gptext.src

SEE ALSO

gpAddTextObject, **gpSetTextObjectBoxColor**, **gpSetTextObjectBoxThickness**, **gpSetTextObjectBoxType**

gpSetTextObjectBoxThickness

PURPOSE Sets the thickness of the outline of the box surrounding a text object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTextObjectBoxThickness**(*pgp*, *framenum*, *name*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
thickness	scalar, thickness of the box outline as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetTextObjectBoxThickness** must be preceded by a call to **gpAddTextObject**. If the text object box type is set to none, then *thickness* is not used. By default, the box type for a text object is set to none. If the box type is changed to filled or hollow, then the default thickness of the box outline is 0.1. Call **gpSetTextObjectBoxType** to change the box type.

EXAMPLE

```
framenum = 1;
objectname = "text1";
xpos = 41;
ypos = 91;
coord = 1;
filledbox = 3;
thickness = 0.4;

ret = gpAddTextObject(&gp,framenum,objectname,xpos,ypos,coord,"f(x) =
ret = gpSetTextObjectBoxType(&gp,framenum,objectname,filledbox);
```

```
ret = gpSetTextObjectBoxThickness(&gp,framenum,objectname,thickness);
```

This example adds a text object to frame one, sets the box type of the text object to filled, and sets the thickness of the box outline to 0.4.

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectBoxStyle**

gpSetTextObjectBoxType

PURPOSE Sets the type of the box surrounding a text object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTextObjectBoxType**(*pgp*, *framenum*, *name*, *boxtype*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
boxtype	scalar, one of the following:

1 no box.
2 hollow box (no background).
3 filled box.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetTextObjectBoxType** must be preceded by a call to **gpAddTextObject**. By default, the box type for a text object is set to none.

EXAMPLE

```
framenum = 1;
```

```
objectname = "text1";
xpos = 41;
ypos = 91;
coord = 1;
filledbox = 3;

ret = gpAddTextObject(&gp,framenum,objectname,xpos,ypos,coord,"f(x) =
ret = gpSetTextObjectBoxType(&gp,framenum,objectname,filledbox);
```

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectBoxStyle**

gpSetTextObjectColor

PURPOSE Sets the color of the text in a text object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTextObjectColor**(*pgp*, *framenum*, *name*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetTextObjectColor** must be preceded by a call to **gpAddTextObject**. By default, the text in a text object is black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
objectname = "text1";
xpos = 41;
ypos = 91;
coord = 1;
ret = gpAddTextObject(&gp,framenum,objectname,xpos,
                          ypos,coord,"f(x) = sin(x)");
ret = gpSetTextObjectColor(&gp,framenum,objectname,"blue");
```

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectFont**, **gpSetTextObjectLineSpacing**, **gpSetTextObjectMargin**

gpSetTextObjectFont

PURPOSE Sets the font of the text in a text object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTextObjectFont**(*pgp*, *framenum*, *name*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame.

2 font height in points.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetTextObjectFont** must be preceded by a call to **gpAddTextObject**. By default, the text in a text object is written in a helvetica bold font (*fonttype* = "helvbold"). The default font height is 14 points. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
objectname = "text1";
xpos = 41;
ypos = 91;
coord = 1;
height = 18;
heightunit = 2;

ret = gpAddTextObject(&gp,framenum,objectname,xpos,
                          ypos,coord,"f(x) = sin(x)");
ret = gpSetTextObjectFont(&gp,framenum,objectname,
                              "timesitalic",height,heightunit);
```

SOURCE gptext.src

SEE ALSO **gpSetDefaultTextFont**, **gpAddTextObject**, **gpSetTextObjectColor**

gpSetTextObjectLineSpacing

PURPOSE Sets the spacing between the lines of text in a text object.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTextObjectLineSpacing**(*pgp*, *framenum*, *name*, *linespacing*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
linespacing	scalar, the spacing between the lines of text.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetTextObjectLineSpacing** must be preceded by a call to **gpAddTextObject**. By default, line spacing is set to 1.

EXAMPLE

```
framenum = 1;
objectname = "text1";
xpos = 41;
ypos = 91;
coord = 1;
linespacing = 1.5;

ret = gpAddTextObject(&gp,framenum,objectname,xpos,ypos,coord,"f(x) =
ret = gpSetTextObjectLineSpacing(&gp,framenum,objectname,linespacing);
```

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectColor**, **gpSetTextObjectFont**, **gpSetTextObjectMargin**

gpSetTextObjectMargin

PURPOSE Sets the size of the margin between the text in a text object and the surrounding box.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetTextObjectMargin**(*pgp*, *framenum*, *name*, *margin*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
name	string, name of a text object created by **gpAddTextObject**.
margin	scalar, the size of the margin between the text and the surrounding box as a percentage of the text height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS A call to **gpSetTextObjectMargin** must be preceded by a call to **gpAddTextObject**. By default, the box type is set to none, so there is no margin. However, if the box type is changed to hollow or filled with **gpSetTextObjectBoxType**, then the margin is set by default to 20.

EXAMPLE

```
framenum = 1;
objectname = "text1";
xpos = 41;
ypos = 91;
coord = 1;
filledbox = 3;
margin = 60;
```

```
ret = gpAddTextObject(&gp,framenum,objectname,xpos,
                          ypos,coord,"f(x) = sin(x)");
ret = gpSetTextObjectBoxType(&gp,framenum,objectname,filledbox);
ret = gpSetTextObjectMargin(&gp,framenum,objectname,margin);
```

SOURCE gptext.src

SEE ALSO **gpAddTextObject**, **gpSetTextObjectColor**, **gpSetTextObjectFont**, **gpSetTextObjectLineSpacing**

gpSetThetaAxisColor

PURPOSE Sets the color of the Theta-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisColor**(*pgp*, *framenum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis lines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
ret = gpSetThetaAxisColor(&gp,framenum,"green");
```

SOURCE gppolarlineplot.src

SEE ALSO **gpShowPolarAxes**, **gpSetThetaAxisThickness**

gpSetThetaAxisLabelColor

PURPOSE Sets the color of the Theta-axis tick labels in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisLabelColor**(*pgp*, *framenum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 2;
ret = gpSetThetaAxisLabelColor(&gp,framenum,"purple");
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisLabelFont**, **gpSetThetaAxisLabelOffset**, **gpSetThetaAxisLabelSkip**

gpSetThetaAxisLabelFont

PURPOSE Sets the type and size of the font used for the Theta-axis tick labels in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetThetaAxisLabelFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points. **3** font height as a percentage of the length of the axis line.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all polar axes tick labels are written in a helvetica font (*fonttype* = "helv") with a font height equal to 3% of the frame height. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
fontheight = 10;
heightunit = 2;
ret = gpSetThetaAxisLabelFont(&gp,framenum,"times",fontheight,heightun
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisLabelColor**, **gpSetThetaAxisLabelOffset**, **gpSetThetaAxisLabelSkip**

gpSetThetaAxisLabelOffset

PURPOSE Sets the offset of the Theta-axis tick labels from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisLabelOffset**(*pgp*, *framenum*, *offset*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
offset	scalar, offset of tick labels from axis line as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels have an offset of 1. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
offset = 2;
ret = gpSetThetaAxisLabelOffset(&gp,framenum,offset);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisLabelColor**, **gpSetThetaAxisLabelFont**, **gpSetThetaAxisLabelSkip**

gpSetThetaAxisLabelSkip

PURPOSE Sets an option to show only one in every *skipnum* Theta-axis tick labels in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisLabelSkip**(*pgp*, *framenum*, *skipnum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
skipnum	scalar, show only one in every *skipnum* Theta-axis tick labels.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels are shown (*skipnum* = 1). This command applies ONLY to frames that contain Polar Line Plots.

EXAMPLE
```
framenum = 3;
skip = 2;
ret = gpSetThetaAxisLabelSkip(&gp,framenum,skip);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisLabelColor**, **gpSetThetaAxisLabelFont**, **gpSetThetaAxisLabelOffset**

gpSetThetaAxisRange

PURPOSE Sets the range of the Theta-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisRange**(*pgp*, *framenum*, *min*, *max*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
min	scalar, minimum value for the Theta-axis.
max	scalar, maximum value for the Theta-axis.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, *theta* is calculated in degrees, and the range of the Theta-axis is 0 - 360. If the *theta* mode is changed to radians with a call to **gpSetThetaMode**, then the default range of the Theta-axis becomes 0 - 2π.

If you call **gpSetThetaAxisPeriod** to reset the Theta-axis range, you should not set (*max* − *min*) > *period*, where *period* is the *theta* range of a complete circle. If the *theta* mode is set to degrees, then the default period is 360. If the *theta* mode is set to radians, then the default period is 2π. Call **gpSetThetaPeriod** to change the period of *theta*.

EXAMPLE

```
framenum = 1;
min = 0;
max = 180;
ret = gpSetThetaAxisRange(&gp,framenum,min,max);
```

This example sets the range of the Theta-axis to 0 - 180. Assuming that the period of *theta* is still set to the default of 360, this call to **gpSetThetaAxisRange** will cause only the top half of the Theta-axis circle to be displayed.

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaMode**, **gpSetThetaPeriod**, **gpSetThetaAxisTicks**

gpSetThetaAxisThickness

PURPOSE Sets the thickness of the Theta-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisThickness**(*pgp*, *framenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
thickness	scalar, thickness of the axis line as a percentage of the height of the frame.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis lines are set to a thickness of 0.4. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
thickness = 0.6;
ret = gpSetThetaAxisThickness(&gp,framenum,thickness);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisColor**

gpSetThetaAxisTicks

PURPOSE Sets the distance between major tick marks and the number of minor tick marks on the Theta-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisTicks**(*pgp*, *framenum*, *majorstep*, *minornum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
majorstep	scalar, step between major tick marks in Theta-axis units.
minornum	scalar, number of minor tick marks between each set of major tick marks.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the step between major tick marks and the number of minor tick marks on the Theta-axis is calculated based on the range of the axis. To modify

the range of the Theta-axis, call **gpSetThetaAxisRange**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
majorstep = 15;
minornum = 2;
ret = gpSetThetaAxisTicks(&gp,framenum,majorstep,minornum);
```

This example sets the distance between major tick marks on the Theta-axis to 15 units, and the number of minor tick marks between each set of major tick marks to 2. Thus there are 5 units from one tick mark to the next.

SOURCE gp.src

SEE ALSO **gpSetThetaAxisRange**

gpSetThetaAxisTitle

PURPOSE Sets the title for the Theta-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisTitle**(*pgp*, *framenum*, *title*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
title	string, title.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, each axis is titled with the name(s) of the variable(s) that are associated with that axis (the variables are separated by a comma and a space if

there are more than one). This command applies ONLY to frames that contain Polar Line plots.

Use **gpSetPolarAxesTitles** to set the titles for both the Theta and R axes in a frame.

EXAMPLE
```
framenum = 1;
ret = gpSetThetaAxisTitle(&gp,framenum,"Theta");
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetPolarAxesTitles**, **gpSetThetaAxisTitleColor**, **gpSetThetaAxisTitleFont**, **gpSetThetaAxisTitleOffset**

gpSetThetaAxisTitleColor

PURPOSE Sets the color of the Theta-axis title in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisTitleColor**(*pgp*, *framenum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis titles are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
ret = gpSetThetaAxisTitleColor(&gp,framenum,"green");
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisTitle**, **gpSetThetaAxisTitleFont**, **gpSetThetaAxisTitleOffset**

gpSetThetaAxisTitleFont

PURPOSE Sets the type and size of the font used for the Theta-axis title in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisTitleFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points. **3** font height as a percentage of the length of the axis line.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all polar axes titles are written in a bold helvetica font (*fonttype* = "helvbold") with a font height equal to 3.6% of the frame height. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
fontheight = 5;
heightunit = 1;
ret = gpSetThetaAxisTitleFont(&gp,framenum,"timesbold",fontheight,heig
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisTitle**, **gpSetThetaAxisTitleColor**, **gpSetThetaAxisTitleOffset**

gpSetThetaAxisTitleOffset

PURPOSE Sets the offset of the Theta-axis title from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaAxisTitleOffset**(*pgp*, *framenum*, *offset*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
offset	scalar, offset of title from axis line as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the offset is set to 6. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
offset = 8;
ret = gpSetThetaAxisTitleOffset(&gp,framenum,offset);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaAxisTitle**, **gpSetThetaAxisTitleColor**, **gpSetThetaAxisTitleFont**

gpSetThetaMode

PURPOSE Specifies whether *theta* is calculated in degrees or in radians.

LIBRARY gaussplot

FORMAT *ret* = **gpSetThetaMode**(*pgp*, *framenum*, *mode*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
mode	scalar, one of the following:

1 calculate *theta* in degrees.
2 calculate *theta* in radians.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, *theta* is calculated in degrees, and the range of the Theta-axis is 0 - 360. If the *theta* mode is changed to radians with **gpSetThetaMode**, then the default range of the Theta-axis becomes 0 - 2π. You can change the Theta-axis range by calling **gpSetThetaRange**. However, since **gpSetThetaMode** resets the Theta-axis range, you should not call **gpSetThetaRange** before **gpSetThetaMode**.

The default period of *theta* (i.e., the *theta* range of a complete circle) is 360. If the *theta* mode is set to radians with **gpSetThetaMode**, then the period becomes 2π. To change the period of *theta* to a value other than 360 or 2π, call **gpSetThetaPeriod**. Calling **gpSetThetaPeriod** also changes the *theta* mode to a user-specified mode, which is neither degrees nor radians. Therefore, if you want to change the period by calling **gpSetThetaPeriod**, you should not also call **gpSetThetaMode**.

EXAMPLE
```
framenum = 1;
mode = 2;
ret = gpSetThetaMode(&gp,framenum,mode);
```

This example sets the *theta* mode to radians, which causes the period to be set to 2π, and the range to 0 - 2π.

SOURCE gppolarlineplot.src

SEE ALSO **gpSetThetaRange**, **gpSetThetaPeriod**, **gpSetThetaAxisTicks**

gpSetThetaPeriod

PURPOSE Sets the *theta* range of a complete circle.

gpSetTileMargins

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetThetaPeriod**(*pgp*, *framenum*, *period*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
period	scalar, number of units on the Theta-axis that make a complete circle.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, *theta* is calculated in degrees, and the period of *theta* is 360. If the *theta* mode is changed to radians with **gpSetThetaPeriod**, then the default period of *theta* becomes 2π. To change the period of *theta* to a value other than 360 or 2π, call **gpSetThetaPeriod**. Calling **gpSetThetaPeriod** also changes the *theta* mode to a user-specified mode, which is neither degrees nor radians. Therefore, if you want to change the period by calling **gpSetThetaPeriod**, you should not also set the *theta* mode with **gpSetThetaMode**.

Calling **gpSetThetaPeriod** resets the Theta-axis range to 0 - *period*. Call **gpSetThetaRange** after **gpSetThetaPeriod** to change the range of the Theta-axis.

EXAMPLE

```
framenum = 1;
period = 100;
ret = gpSetThetaPeriod(&gp,framenum,period);
```

This example sets the *theta* period to 100 (meaning that 100 units on the Theta-axis make a complete circle), changes the *theta* mode to a user-specified mode (neither degrees nor radians), and resets the range of the Theta-axis to 0 - 100.

SOURCE `gppolarlineplot.src`

SEE ALSO **gpSetThetaMode**, **gpSetThetaRange**, **gpSetThetaAxisTicks**

gpSetTileMargins

PURPOSE Sets the margins between the frames and the edges of the paper for the **gpTile** command.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTileMargins**(*pgp*, *lmargin*, *rmargin*, *tmargin*, *bmargin*);

INPUT

pgp	pointer to a plot control structure.
lmargin	scalar, margin between frames and left edge of paper in inches.
rmargin	scalar, margin between frames and right edge of paper in inches.
tmargin	scalar, margin between frames and top edge of paper in inches.
bmargin	scalar, margin between frames and bottom edge of paper in inches.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, **gpTile** leaves 0.25 inch margins between the frames and the edges of the paper. To change the margins, you must call **gpSetTileMargins** before **gpTile**.

EXAMPLE

```
framesperrow = 3;
lrmargin = 0.5;
tmargin = 0.25;
bmargin = 0.4;
ret = gpSetTileMargins(&gp,lrmargin,lrmargin,tmargin,bmargin);
ret = gpTile(&gp,framesperrow);
```

This example sets the margins between the frames and the left and right edges of the paper to 0.5 inches, the margin between the frames and the top edge of the paper to 0.25 inches, and the margin between the frames and the bottom edge of the paper to 0.4 inches. It then generates three rows of frames in the graph,

placing three frames in the first and third rows and two frames in the second row. The example assumes that a total of 8 frames have been added to the graph.

SOURCE gp.src

SEE ALSO **gpTile**, **gpMoveFrame**

gpSetTranslation

PURPOSE Translates a plot horizontally and/or vertically.

LIBRARY gaussplot

FORMAT *ret* = **gpSetTranslation**(*pgp*, *framenum*, *xtranslate*, *ytranslate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
xtranslate	scalar, horizontal translation as a percentage of the frame width.
ytranslate	scalar, vertical translation as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the plot is centered. Inputting a positive value for *xtranslate* translates the plot right, while a negative value translates the plot left. Similarly, a positive value for *ytranslate* translates the plot up, while a negative value translates the plot down.

EXAMPLE

```
framenum = 1;
xtranslate = 25;
ytranslate = -15;
ret = gpSetTranslation(&gp,framenum,xtranslate,ytranslate);
```

This example translates the plot 25% of the frame width to the right and 15% of the frame height down.

SOURCE gp.src

SEE ALSO **gpSetMagnification**, **gpSetPlotFit**, **gpSetFrameStyle**

gpSetTranslucency

PURPOSE Sets the percentage of translucency of shade and contour layers in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetTranslucency(***pgp*, *framenum*, *fieldzonenum*, *translucency***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
translucency	scalar, percentage of translucency.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Field zone numbers correspond to the number of the associated zone in the data file. By default, translucency is set to 50%.

EXAMPLE

```
framenum = 1;
seton = 1;
fieldzonenum = 2;
translucency = 75;
ret = gpUseTranslucencyEffect(&gp,framenum,seton);
ret = gpSetTranslucency(&gp,framenum,fieldzonenum,translucency);
```

SOURCE gpfieldzone.src

SEE ALSO **gpUseTranslucencyEffect**, **gpUseFieldZoneTranslucency**, **gpShowShadeLayer**

gpSetVectorArrowHeadType

PURPOSE Sets the arrow head type for the vectors in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetVectorArrowHeadType**(*pgp*, *framenum*, *fieldzonenum*, *type*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
type	scalar, one of the following:

1 plain arrow head.
2 filled arrow head.
3 hollow arrow head.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the vector layer is turned off. To activate the vector layer in a frame, call **gpShowVectorLayer**. When the vector layer is activated, plain arrow heads are used by default.

Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE

```
framenum = 1;
```

```
seton = 1;
fieldzonenum = 2;
type = 3;
ret = gpShowVectorLayer(&gp,framenum,seton);
ret = gpSetVectorArrowHeadType(&gp,framenum,fieldzonenum,type);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetVectorColor**, **gpSetVectorType**, **gpSetVectorVariables**

gpSetVectorColor

PURPOSE Sets the color of the vectors in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetVectorColor**(*pgp*, *framenum*, *fieldzonenum*, *vectorcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
vectorcolor	string, name of vector color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the vector layer is turned off. To activate the vector layer in a frame, call **gpShowVectorLayer**.

Field zone numbers correspond to the number of the associated zone in the data file. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
seton = 1;
fieldzonenum = 2;
ret = gpShowVectorLayer(&gp,framenum,seton);
ret = gpSetVectorColor(&gp,framenum,fieldzonenum,"purple");
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowVectorLayer**, **gpShowFieldZoneVectorLayer**, **gpSetVectorLinePattern**, **gpSetVectorLineThickness**

gpSetVectorLinePattern

PURPOSE Sets the line pattern of the vectors in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetVectorLinePattern**(*pgp*, *framenum*, *fieldzonenum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the vector layer is turned off. To activate the vector layer in a frame, call **gpShowVectorLayer**. When the vector layer is activated, vectors are drawn with solid lines by default.

The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
seton = 1;
fieldzonenum = 2;
ret = gpShowVectorLayer(&gp,framenum,seton);
ret = gpSetVectorLinePattern(&gp,framenum,fieldzonenum,"dashdotdot");
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetVectorLinePatternLength**, **gpSetVectorColor**, **gpSetVectorType**

gpSetVectorLinePatternLength

PURPOSE Sets the length of the vector line pattern in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetVectorLinePatternLength**(*pgp*, *framenum*, *fieldzonenum*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).

	length	scalar, length of one cycle of the line pattern as a percentage of the frame height.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the vector layer is turned off. To activate the vector layer in a frame, call **gpShowVectorLayer**. When the vector layer is activated, vectors are drawn with solid lines by default, and the line pattern length is ignored. If a different line pattern is selected with **gpSetVectorLinePattern**, the default pattern length is 2.

EXAMPLE
```
framenum = 1;
seton = 1;
fieldzonenum = 2;
length = 4;
ret = gpShowVectorLayer(&gp,framenum,seton);
ret = gpSetVectorLinePattern(&gp,framenum,fieldzonenum,"dashdot");
ret = gpSetVectorLinePatternLength(&gp,framenum,fieldzonenum,length);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetVectorLinePattern**, **gpSetVectorColor**, **gpSetVectorLineThickness**

gpSetVectorLineThickness

PURPOSE Sets the line thickness of the vectors in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetVectorLineThickness**(*pgp*, *framenum*, *fieldzonenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
thickness	scalar, line thickness of the vectors as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the vector layer is turned off. To activate the vector layer in a frame, call **gpShowVectorLayer**. When the vector layer is activated, vectors have a line thickness of 0.1 by default.

Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE
```
framenum = 3;
seton = 1;
fieldzonenum = 2;
thickness = 0.4;
ret = gpShowVectorLayer(&gp,framenum,seton);
ret = gpSetVectorLineThickness(&gp,framenum,fieldzonenum,thickness);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSetVectorColor**, **gpSetVectorType**, **gpSetVectorVariables**

gpSetVectorType

PURPOSE Sets the type of vector used in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpSetVectorType**(*pgp*, *framenum*, *fieldzonenum*, *type*);

INPUT

pgp pointer to a plot control structure.

framenum scalar, frame number.

fieldzonenum scalar or N×1 vector, field zone number(s).

type scalar, one of the following:

1 vector tail at data point.

2 vector head at data point.

3 vector midpoint at data point.

4 display only vector head.

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

By default, the vector layer is turned off. To activate the vector layer in a frame, call **gpShowVectorLayer**. When the vector layer is activated, vectors are placed in the graph with the vector tails at data points by default.

Field zone numbers correspond to the number of the associated zone in the data file.

EXAMPLE

```
framenum = 1;
seton = 1;
fieldzonenum = 2;
type = 3;
ret = gpShowVectorLayer(&gp,framenum,seton);
ret = gpSetVectorType(&gp,framenum,fieldzonenum,type);
```

SOURCE

gpfieldzone.src

SEE ALSO

gpSetVectorColor, **gpSetVectorArrowHeadType**, **gpSetVectorVariables**

gpSetVectorVariables

PURPOSE Sets the variables to use as the vector components for the vectors in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetVectorVariables**(*pgp*, *framenum*, *variables*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
variables	N×1 string array, names of variables. – or – N×1 vector, indices of the variables in the data file.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the vector layer is turned off. To activate the vector layer in a frame, call **gpShowVectorLayer**.

If the specified frame contains a 2D Cartesian or 2D Contour plot, *variables* should be a 2×1 vector or string array. If the frame contains a 3D Cartesian or 3D Contour plot, *variables* should be a 3×1 vector or string array. This command does not apply to Bar, XY Line or Polar Line plots.

By default, the first and second variables in the data set will be used as the vector components for 2D plots, and the first, second and third variables will be used as the vector components for 3D plots.

EXAMPLE
```
framenum = 1;
showvectors = 1;
variables = { 1,4,5 };
ret = gpShowVectorLayer(&gp,framenum,showvectors);
ret = gpSetVectorVariables(&gp,framenum,variables);
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowVectorLayer**, **gpShowFieldZoneVectorLayer**, **gpSetVectorColor**, **gpSetVectorType**

gpSetWorkspaceView

PURPOSE Sets workspace view options.

LIBRARY gaussplot

FORMAT **gpSetWorkspaceView(***pgp*, *showruler*, *showpaper***);**

INPUT

pgp	pointer to a plot control structure.
showruler	scalar, one of the following:
	0 do not display the ruler.
	1 display the ruler.
showpaper	scalar, one of the following:
	0 do not display the paper.
	1 display the paper.

REMARKS By default, a ruler is shown on the bottom and the right-hand side of the workspace, and paper is shown behind the frames in the workspace.

EXAMPLE
```
showruler = 0;
showpaper = 0;
gpSetWorkspaceView(&gp,showruler,showpaper);
```

SOURCE gp.src

SEE ALSO **gpSetFrameStyle**

gpSetXAxisColor

PURPOSE Sets the color of an X-axis line in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetXAxisColor**(*pgp*, *framenum*, *axisnum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, all axis lines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
axisnum = 1;
ret = gpSetXAxisColor(&gp,framenum,axisnum,"green");
```

SOURCE `gp.src`

SEE ALSO **gpShowXAxis**, **gpPositionXAxis**, **gpSetXAxisThickness**

gpSetXAxisGridLineColor

PURPOSE Sets the color of the X-axis major gridlines.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetXAxisGridLineColor**(*pgp*, *framenum*, *axisnum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, 1 ≤ *axisnum leq* 5, X-axis number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, major gridlines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowXAxisGridLines(&gp,framenum,axisnum,show);
```

```
ret = gpSetXAxisGridLineColor(&gp,framenum,axisnum,"red");
```

SOURCE gp.src

SEE ALSO **gpShowXAxisGridLines**, **gpSetXAxisGridLineThickness**, **gpSetXAxisGridPattern**

gpSetXAxisGridLineThickness

PURPOSE Sets the thickness of the X-axis major gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisGridLineThickness**(*pgp*, *framenum*, *axisnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
thickness	scalar, thickness of the X-axis major gridlines as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, major gridlines have a line thickness of 0.1.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
thickness = 0.25;
ret = gpShowXAxisGridLines(&gp,framenum,axisnum,show);
ret = gpSetXAxisGridLineThickness(&gp,framenum,axisnum,thickness);
```

SOURCE gp.src

SEE ALSO **gpShowXAxisGridLines**, **gpSetXAxisGridLineColor**, **gpSetXAxisGridPattern**

gpSetXAxisGridPattern

PURPOSE Sets the pattern of the X-axis major gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisGridPattern**(*pgp*, *framenum*, *axisnum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display

additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, major gridlines are solid. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowXAxisGridLines(&gp,framenum,axisnum,show);
ret = gpSetXAxisGridPattern(&gp,framenum,axisnum,"dashed");
```

SOURCE gp.src

SEE ALSO **gpSetXAxisGridPatternLength**, **gpShowXAxisGridLines**, **gpSetXAxisGridLineColor**

gpSetXAxisGridPatternLength

PURPOSE Sets the length of the X-axis major grid line pattern.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisGridPatternLength**(*pgp*, *framenum*, *axisnum*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
length	scalar, length of one cycle of the line pattern as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

Major gridlines are drawn as solid lines by default, and the pattern length is ignored. If a different line pattern is selected with **gpSetXAxisGridLinePattern**, then the default pattern length is 2.

EXAMPLE

```
framenum = 1;
axisnum = 1;
show = 1;
length = 4;
ret = gpShowXAxisGridLines(&gp,framenum,axisnum,show);
ret = gpSetXAxisGridPattern(&gp,framenum,axisnum,"dashed");
ret = gpSetXAxisGridPatternLength(&gp,framenum,axisnum,length);
```

SOURCE

gp.src

SEE ALSO

gpSetXAxisGridPattern, **gpShowXAxisGridLines**, **gpSetXAxisGridLineColor**

gpSetXAxisLabelColor

PURPOSE Sets the color of X-axis tick labels in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetXAxisLabelColor**(*pgp*, *framenum*, *axisnum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, all axis tick labels are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 2;
axisnum = 1;
ret = gpSetXAxisLabelColor(&gp,framenum,axisnum,"purple");
```

SOURCE `gp.src`

SEE ALSO **gpSetXAxisLabelFont**, **gpSetXAxisLabelOffset**, **gpSetXAxisLabelSkip**

gpSetXAxisLabelFont

PURPOSE Sets the type and size of the font used for X-axis tick labels in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetXAxisLabelFont**(*pgp*, *framenum*, *axisnum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points. **3** font height as a percentage of the length of the axis line (valid only for 3D Cartesian and 3D Contour plots).

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an

X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, all axis tick labels are written in a helvetica font (*fonttype* = "helv"). The default font height is equal to 3% of the axis length for 3D Cartesian and 3D Contour plots and equal to 3% of the frame height for all other plot types. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
axisnum = 1;
fontheight = 10;
heightunit = 2;
ret = gpSetXAxisLabelFont(&gp,framenum,axisnum,"times",fontheight,heig
```

SOURCE gp.src

SEE ALSO **gpSetXAxisLabelColor**, **gpSetXAxisLabelOffset**, **gpSetXAxisLabelSkip**

gpSetXAxisLabelOffset

PURPOSE Sets the offset of X-axis tick labels from the axis line in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetXAxisLabelOffset**(*pgp*, *framenum*, *axisnum*, *offset*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
offset	scalar, offset of tick labels from axis line.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

The offset is defined as a percentage of the median axis length for 3D Cartesian and 3D Contour plots, and as a percentage of the frame height for all other plot types. By default, all axis tick labels have an offset of 1. This command does not apply to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
axisnum = 1;
offset = 2;
ret = gpSetXAxisLabelOffset(&gp,framenum,axisnum,offset);
```

SOURCE `gp.src`

SEE ALSO **gpSetXAxisLabelColor**, **gpSetXAxisLabelFont**, **gpSetXAxisLabelSkip**

gpSetXAxisLabelSkip

PURPOSE Sets an option to show only one in every *skipnum* X-axis tick labels in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetXAxisLabelSkip**(*pgp*, *framenum*, *axisnum*, *skipnum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
skipnum	scalar, show only one in every *skipnum* X-axis tick labels.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, all axis tick labels are shown (*skipnum* = 1). This command does not apply to frames that contain Polar Line Plots.

EXAMPLE
```
framenum = 3;
axisnum = 1;
skip = 2;
ret = gpSetXAxisLabelSkip(&gp,framenum,axisnum,skip);
```

SOURCE gp.src

SEE ALSO **gpSetXAxisLabelColor**, **gpSetXAxisLabelFont**, **gpSetXAxisLabelOffset**

gpSetXAxisMinorGridLineColor

PURPOSE Sets the color of the X-axis minor gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisMinorGridLineColor**(*pgp*, *framenum*, *axisnum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, minor gridlines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowXAxisMinorGridLines(&gp,framenum,axisnum,show);
ret = gpSetXAxisMinorGridLineColor(&gp,framenum,axisnum,"red");
```

SOURCE gp.src

SEE ALSO **gpShowXAxisMinorGridLines**, **gpSetXAxisMinorGridLineThickness**, **gpSetXAxisMinorGridPattern**

gpSetXAxisMinorGridLineThickness

PURPOSE Sets the thickness of the X-axis minor gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisMinorGridLineThickness**(*pgp*, *framenum*, *axisnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
thickness	scalar, thickness of the X-axis minor gridlines as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an

X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, minor gridlines have a line thickness of 0.1.

EXAMPLE

```
framenum = 1;
axisnum = 1;
show = 1;
thickness = 0.25;
ret = gpShowXAxisMinorGridLines(&gp,framenum,axisnum,show);
ret = gpSetXAxisMinorGridLineThickness(&gp,framenum,axisnum,thickness)
```

SOURCE gp.src

SEE ALSO **gpShowXAxisMinorGridLines**, **gpSetXAxisMinorGridLineColor**, **gpSetXAxisMinorGridPattern**

gpSetXAxisMinorGridPattern

PURPOSE Sets the pattern of the X-axis minor gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisMinorGridPattern**(*pgp*, *framenum*, *axisnum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, minor gridlines are dotted. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowXAxisMinorGridLines(&gp,framenum,axisnum,show);
ret = gpSetXAxisMinorGridPattern(&gp,framenum,axisnum,"dashed");
```

SOURCE gp.src

SEE ALSO **gpSetXAxisMinorGridPatternLength**, **gpShowXAxisMinorGridLines**, **gpSetXAxisMinorGridLineColor**

gpSetXAxisMinorGridPatternLength

PURPOSE Sets the length of the X-axis minor grid line pattern.

gpSetXAxisRange

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetXAxisMinorGridPatternLength(***pgp*, *framenum*, *axisnum*, *length***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
length	scalar, length of one cycle of the line pattern as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

Minor gridlines are drawn as dotted lines with a pattern length of 2 by default.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
length = 4;
ret = gpShowXAxisMinorGridLines(&gp,framenum,axisnum,show);
ret = gpSetXAxisMinorGridPattern(&gp,framenum,axisnum,"dashdotdot");
ret = gpSetXAxisMinorGridPatternLength(&gp,framenum,axisnum,length);
```

SOURCE `gp.src`

SEE ALSO **gpSetXAxisMinorGridPattern**, **gpShowXAxisMinorGridLines**, **gpSetXAxisMinorGridLineColor**

gpSetXAxisRange

PURPOSE Sets the range of an X-axis in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetXAxisRange**(*pgp*, *framenum*, *axisnum*, *min*, *max*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
min	scalar, minimum value for the X-axis.
max	scalar, maximum value for the X-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

The range of an X-axis is calculated based on the values of the data points by default. If the axes are dependent on one another, which is the default for 2D and 3D Cartesian and 2D and 3D Contour plots, then the range of the X-axis is determined by the X, Y and (for 3D plots), Z values of the data points. Otherwise, the range is determined by only the X values. To change the number of major and minor tick marks or the distance between them, call **gpSetXAxisTicks**.

EXAMPLE
```
framenum = 1;
axisnum = 1;
```

```
min = 5;
max = 15;
ret = gpSetXAxisRange(&gp,framenum,axisnum,min,max);
```

SOURCE gp.src

SEE ALSO **gpSetXAxisTicks**

gpSetXAxisThickness

PURPOSE Sets the thickness of an X-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisThickness**(*pgp*, *framenum*, *axisnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
thickness	scalar, thickness of the axis line as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

All axis lines are set to a thickness of 0.4 by default. This command does not apply to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
thickness = 0.6;
ret = gpSetXAxisThickness(&gp,framenum,axisnum,thickness);
```

SOURCE gp.src

SEE ALSO **gpSetXAxisColor**

gpSetXAxisTicks

PURPOSE Sets the distance between major tick marks and the number of minor tick marks on an X-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisTicks**(*pgp*, *framenum*, *axisnum*, *majorstep*, *minornum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
majorstep	scalar, step between major tick marks in X-axis units.
minornum	scalar, number of minor tick marks between each set of major tick marks.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an

X-axis numbered greater than one. X axes that are added in this way are displayed by default.

The step between major tick marks and the number of minor tick marks on an X-axis is calculated based on the range of the axis by default. To modify the range of an X-axis, call **gpSetXAxisRange**.

EXAMPLE

```
framenum = 1;
axisnum = 1;
majorstep = 2;
minornum = 3;
ret = gpSetXAxisTicks(&gp,framenum,axisnum,majorstep,minornum);
```

This example sets the distance between major tick marks on the first X-axis to 2 units, and the number of minor tick marks between each set of major tick marks to 3. Therefore, there will be 0.5 units from one tick mark to the next.

SOURCE gp.src

SEE ALSO **gpSetXAxisRange**

gpSetXAxisTitle

PURPOSE Sets the title for an X-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisTitle**(*pgp*, *framenum*, *axisnum*, *title*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
title	string, title.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

Each axis is titled with the name(s) of the variable(s) that are associated with that axis by default (the variables are separated by a comma and a space if there are more than one).

EXAMPLE
```
framenum = 1;
axisnum = 1;
ret = gpSetXAxisTitle(&gp,framenum,axisnum,"X");
```

SOURCE gp.src

SEE ALSO **gpSetXAxisTitleColor**, **gpSetXAxisTitleFont**, **gpSetXAxisTitleOffset**

gpSetXAxisTitleColor

PURPOSE Sets the color of an X-axis title in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisTitleColor**(*pgp*, *framenum*, *axisnum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.

	textcolor	string, name of color. – or – scalar, index of color in color list.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

All axis titles are black by default. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
ret = gpSetXAxisTitleColor(&gp,framenum,axisnum,"green");
```

SOURCE gp.src

SEE ALSO **gpSetXAxisTitle**, **gpSetXAxisTitleFont**, **gpSetXAxisTitleOffset**

gpSetXAxisTitleFont

PURPOSE Sets the type and size of the font used for an X-axis title in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisTitleFont**(*pgp*, *framenum*, *axisnum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp pointer to a plot control structure.

framenum scalar, frame number.

axisnum scalar, $1 \leq axisnum \leq 5$, X-axis number.

fonttype string, name of font.
– or –
scalar, index of font in font list.

height scalar, font height.

heightunit scalar, one of the following:

1 font height as a percentage of the height of the frame.

2 font height in points.

3 font height as a percentage of the length of the axis line (valid only for 3D Cartesian and 3D Contour plots)

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

All axis tick labels are written in a bold helvetica font (*fonttype* = "helvbold") by default. The default font height is equal to 3.6% of the axis length for 3D Cartesian and 3D Contour plots and equal to 3.6% of the frame height for all other plot types. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
fontheight = 5;
heightunit = 1;
ret = gpSetXAxisTitleFont(&gp,framenum,axisnum,"timesbold",fontheight,
```

SOURCE gp.src

SEE ALSO **gpSetXAxisTitle**, **gpSetXAxisTitleColor**, **gpSetXAxisTitleOffset**

gpSetXAxisTitleOffset

PURPOSE Sets the offset of an X-axis title from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetXAxisTitleOffset**(*pgp*, *framenum*, *axisnum*, *offset*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
	offset	scalar, offset of title from axis line.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

For 3D Cartesian and 3D Contour plots, the offset is defined as a percentage of the median axis length, and the default is 8. For all other plot types, it is defined as a percentage of the height of the frame, and the default is 6. This command does not apply to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
offset = 10;
ret = gpSetXAxisTitleOffset(&gp,framenum,axisnum,offset);
```

SOURCE gp.src

SEE ALSO **gpSetXAxisTitle**, **gpSetXAxisTitleColor**, **gpSetXAxisTitleFont**

gpSetYAxisColor

PURPOSE Sets the color of a Y-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisColor**(*pgp*, *framenum*, *axisnum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

All axis lines are black by default. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
axisnum = 1;
ret = gpSetYAxisColor(&gp,framenum,axisnum,"blue");
```

SOURCE gp.src

SEE ALSO **gpShowYAxis**, **gpSetYAxisThickness**

gpSetYAxisGridLineColor

PURPOSE Sets the color of the Y-axis major gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisGridLineColor**(*pgp*, *framenum*, *axisnum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, major gridlines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowYAxisGridLines(&gp,framenum,axisnum,show);
ret = gpSetYAxisGridLineColor(&gp,framenum,axisnum,"blue");
```

SOURCE gp.src

SEE ALSO **gpShowYAxisGridLines**, **gpSetYAxisGridLineThickness**, **gpSetYAxisGridPattern**

gpSetYAxisGridLineThickness

PURPOSE Sets the thickness of the Y-axis major gridlines.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetYAxisGridLineThickness**(*pgp*, *framenum*, *axisnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
thickness	scalar, thickness of the Y-axis major gridlines as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, major gridlines have a line thickness of 0.1.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
thickness = 0.25;
ret = gpShowYAxisGridLines(&gp,framenum,axisnum,show);
ret = gpSetYAxisGridLineThickness(&gp,framenum,axisnum,thickness);
```

SOURCE `gp.src`

SEE ALSO **gpShowYAxisGridLines**, **gpSetYAxisGridLineColor**, **gpSetYAxisGridPattern**

gpSetYAxisGridPattern

PURPOSE Sets the pattern of the Y-axis major gridlines.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetYAxisGridPattern**(*pgp*, *framenum*, *axisnum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, major gridlines are solid. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowYAxisGridLines(&gp,framenum,axisnum,show);
ret = gpSetYAxisGridPattern(&gp,framenum,axisnum,"dashdot");
```

SOURCE gp.src

SEE ALSO **gpSetYAxisGridPatternLength**, **gpShowYAxisGridLines**, **gpSetYAxisGridLineColor**

gpSetYAxisGridPatternLength

PURPOSE Sets the length of the Y-axis major grid line pattern.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisGridPatternLength**(*pgp*, *framenum*, *axisnum*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.

	length	scalar, length of one cycle of the line pattern as a percentage of the frame height.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

Major gridlines are drawn as solid lines by default. If a different line pattern is selected with **gpSetYAxisGridLinePattern**, then the default pattern length is 2.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
length = 4;
ret = gpShowYAxisGridLines(&gp,framenum,axisnum,show);
ret = gpSetYAxisGridPattern(&gp,framenum,axisnum,"dashdot");
ret = gpSetYAxisGridPatternLength(&gp,framenum,axisnum,length);
```

SOURCE gp.src

SEE ALSO **gpSetYAxisGridPattern**, **gpShowYAxisGridLines**, **gpSetYAxisGridLineColor**

gpSetYAxisLabelColor

PURPOSE Sets the color of Y-axis tick labels in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisLabelColor**(*pgp*, *framenum*, *axisnum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

All axis tick labels are black by default. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
axisnum = 2;
ret = gpSetYAxisLabelColor(&gp,framenum,axisnum,"red");
```

SOURCE gp.src

SEE ALSO **gpSetYAxisLabelFont**, **gpSetYAxisLabelOffset**, **gpSetYAxisLabelSkip**

gpSetYAxisLabelFont

PURPOSE Sets the type and size of the font used for Y-axis tick labels in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisLabelFont**(*pgp*, *framenum*, *axisnum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points. **3** font height as a percentage of the length of the axis line (valid only for 3D Cartesian and 3D Contour plots).

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

All axis tick labels are written in a helvetica font (*fonttype* = "helv") by default. The default font height is equal to 3% of the axis length for 3D Cartesian and 3D Contour plots and equal to 3% of the frame height for all other plot types. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
axisnum = 1;
height = 12;
heightunit = 2;
ret = gpSetYAxisLabelFont(&gp,framenum,axisnum,"timesitalic",height,he
```

SOURCE gp.src

SEE ALSO **gpSetYAxisLabelColor**, **gpSetYAxisLabelOffset**, **gpSetYAxisLabelSkip**

gpSetYAxisLabelOffset

PURPOSE Sets the offset of Y-axis tick labels from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisLabelOffset**(*pgp*, *framenum*, *axisnum*, *offset*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
offset	scalar, offset of tick labels from axis line.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

The offset is defined as a percentage of the median axis length for 3D Cartesian and 3D Contour plots, and as a percentage of the frame height for all other plot types. By default, all axis tick labels have an offset of 1. This command does not apply to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 2;
axisnum = 1;
offset = 2;
ret = gpSetYAxisLabelOffset(&gp,framenum,axisnum,offset);
```

SOURCE gp.src

SEE ALSO **gpSetYAxisLabelColor**, **gpSetYAxisLabelFont**, **gpSetYAxisLabelSkip**

gpSetYAxisLabelSkip

PURPOSE Sets an option to show only one in every *skipnum* Y-axis tick labels in the specified frame.

gpSetYAxisMinorGridLineColor

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetYAxisLabelSkip**(*pgp*, *framenum*, *axisnum*, *skipnum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
skipnum	scalar, show only one in every *skipnum* Y-axis tick labels.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

All axis tick labels are shown (*skipnum* = 1) by default. This command does not apply to frames that contain Polar Line Plots.

EXAMPLE

```
framenum = 1;
axisnum = 2;
skip = 2;
ret = gpSetYAxisLabelSkip(&gp,framenum,axisnum,skip);
```

SOURCE `gp.src`

SEE ALSO **gpSetYAxisLabelColor**, **gpSetYAxisLabelFont**, **gpSetYAxisLabelOffset**

gpSetYAxisMinorGridLineColor

PURPOSE Sets the color of the Y-axis minor gridlines.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetYAxisMinorGridLineColor**(*pgp*, *framenum*, *axisnum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, minor gridlines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowYAxisMinorGridLines(&gp,framenum,axisnum,show);
ret = gpSetYAxisMinorGridLineColor(&gp,framenum,axisnum,"blue");
```

SOURCE gp.src

SEE ALSO **gpShowYAxisMinorGridLines**, **gpSetYAxisMinorGridLineThickness**, **gpSetYAxisMinorGridPattern**

gpSetYAxisMinorGridLineThickness

PURPOSE Sets the thickness of the Y-axis minor gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisMinorGridLineThickness**(*pgp*, *framenum*, *axisnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
thickness	scalar, thickness of the Y-axis minor gridlines as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, minor gridlines have a line thickness of 0.1.

EXAMPLE

```
framenum = 1;
```

```
axisnum = 1;
show = 1;
thickness = 0.25;
ret = gpShowYAxisMinorGridLines(&gp,framenum,axisnum,show);
ret = gpSetYAxisMinorGridLineThickness(&gp,framenum,axisnum,thickness)
```

SOURCE gp.src

SEE ALSO **gpShowYAxisMinorGridLines**, **gpSetYAxisMinorGridLineColor**, **gpSetYAxisMinorGridPattern**

gpSetYAxisMinorGridPattern

PURPOSE Sets the pattern of the Y-axis minor gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisMinorGridPattern**(*pgp*, *framenum*, *axisnum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display

additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, minor gridlines are dotted. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**.

EXAMPLE

```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowYAxisMinorGridLines(&gp,framenum,axisnum,show);
ret = gpSetYAxisMinorGridPattern(&gp,framenum,axisnum,"dashdot");
```

SOURCE gp.src

SEE ALSO **gpSetYAxisMinorGridPatternLength**, **gpShowYAxisMinorGridLines**, **gpSetYAxisMinorGridLineColor**

gpSetYAxisMinorGridPatternLength

PURPOSE Sets the length of the Y-axis minor grid line pattern.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisMinorGridPatternLength**(*pgp*, *framenum*, *axisnum*, *length*);

INPUT	*pgp*	pointer to a plot control structure.
	framenum	scalar, frame number.
	axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
	length	scalar, length of one cycle of the line pattern as a percentage of the frame height.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

Minor gridlines are drawn as dotted lines with a pattern length of 2 by default.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
length = 4;
ret = gpShowYAxisMinorGridLines(&gp,framenum,axisnum,show);
ret = gpSetYAxisMinorGridPattern(&gp,framenum,axisnum,"dashdotdot");
ret = gpSetYAxisMinorGridPatternLength(&gp,framenum,axisnum,length);
```

SOURCE gp.src

SEE ALSO **gpSetYAxisMinorGridPattern**, **gpShowYAxisMinorGridLines**, **gpSetYAxisMinorGridLineColor**

gpSetYAxisRange

PURPOSE Sets the range of a Y-axis in the specified frame.

gpSetYAxisRange

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetYAxisRange**(*pgp*, *framenum*, *axisnum*, *min*, *max*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
min	scalar, minimum value for the Y-axis.
max	scalar, maximum value for the Y-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

The range of the Y-axis is calculated based on the values of the data points by default. If the axes are dependent on one another, which is the default for 2D and 3D Cartesian and 2D and 3D Contour plots, then the range of the Y-axis will be determined by the X, Y and (for 3D plots) Z values of the data points. Otherwise, the range will be determined by only the Y values. To change the number of major and minor tick marks or the distance between them, call **gpSetYAxisTicks**.

EXAMPLE

```
framenum = 1;
axisnum = 2;
min = -10;
max = 100;
ret = gpSetYAxisRange(&gp,framenum,axisnum,min,max);
```

SOURCE `gp.src`

SEE ALSO **gpSetYAxisTicks**

gpSetYAxisThickness

PURPOSE Sets the thickness of a Y-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisThickness**(*pgp*, *framenum*, *axisnum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
thickness	scalar, thickness of the axis line as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

All axis lines are set to a thickness of 0.4 by default. This command does not apply to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
thickness = 0.2;
ret = gpSetYAxisThickness(&gp,framenum,axisnum,thickness);
```

SOURCE gp.src

SEE ALSO **gpSetYAxisColor**

gpSetYAxisTicks

PURPOSE Sets the distance between major tick marks and the number of minor tick marks on a Y-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisTicks**(*pgp*, *framenum*, *axisnum*, *majorstep*, *minornum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
majorstep	scalar, step between major tick marks in Y-axis units.
minornum	scalar, number of minor tick marks between each set of major tick marks.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

The step between major tick marks and the number of minor tick marks on the Y-axis is calculated based on the range of the axis by default. To modify the range of the Y-axis, call **gpSetYAxisRange**.

EXAMPLE

```
framenum = 1;
axisnum = 2;
majorstep = 1;
minornum = 3;
ret = gpSetYAxisTicks(&gp,framenum,axisnum,majorstep,minornum);
```

This example sets the distance between major tick marks on the second Y-axis to 1 unit, and the number of minor tick marks between each set of major tick marks to 3. Therefore, there will be 0.25 units from one tick mark to the next.

SOURCE gp.src

SEE ALSO **gpSetYAxisRange**

gpSetYAxisTitle

PURPOSE Sets the title for a Y-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisTitle**(*pgp*, *framenum*, *axisnum*, *title*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
title	string, title.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display

additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

Each axis is titled with the name(s) of the variable(s) that are associated with that axis by default (the variables are separated by a comma and a space if there are more than one).

EXAMPLE

```
framenum = 1;
axisnum = 1;
ret = gpSetYAxisTitle(&gp,framenum,axisnum,"Y");
```

SOURCE gp.src

SEE ALSO **gpSetYAxisTitleColor**, **gpSetYAxisTitleFont**, **gpSetYAxisTitleOffset**

gpSetYAxisTitleColor

PURPOSE Sets the color of a Y-axis title in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisTitleColor**(*pgp*, *framenum*, *axisnum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

All axis titles are black by default. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
ret = gpSetYAxisTitleColor(&gp,framenum,axisnum,"purple");
```

SOURCE gp.src

SEE ALSO **gpSetYAxisTitle**, **gpSetYAxisTitleFont**, **gpSetYAxisTitleOffset**

gpSetYAxisTitleFont

PURPOSE Sets the type and size of the font used for a Y-axis title in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisTitleFont**(*pgp*, *framenum*, *axisnum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
fonttype	string, name of font. – or – scalar, index of font in font list.

height scalar, font height.

heightunit scalar, one of the following:

1 font height as a percentage of the height of the frame.

2 font height in points.

3 font height as a percentage of the length of the axis line (valid only for 3D Cartesian and 3D Contour plots).

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

All axis tick labels are written in a bold helvetica font (*fonttype* = "helvbold") by default. The default font height is equal to 3.6% of the axis length for 3D Cartesian and 3D Contour plots and equal to 3.6% of the frame height for all other plot types. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command does not apply to frames that contain Polar Line plots.

EXAMPLE

```
framenum = 1;
```

```
axisnum = 1;
height = 15;
heightunit = 2;
ret = gpSetYAxisTitleFont(&gp,framenum,axisnum,"courierbold",height,he
```

SOURCE gp.src

SEE ALSO **gpSetYAxisTitle**, **gpSetYAxisTitleColor**, **gpSetYAxisTitleOffset**

gpSetYAxisTitleOffset

PURPOSE Sets the offset of a Y-axis title from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetYAxisTitleOffset**(*pgp*, *framenum*, *axisnum*, *offset*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
offset	scalar, offset of title from axis line.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

For 3D Cartesian and 3D Contour plots, the offset is defined as a percentage of the median axis length, and the default is 8. For all other plot types, it is defined

as a percentage of the height of the frame, and the default is 6. This command does not apply to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 3;
axisnum = 1;
offset = 5;
ret = gpSetYAxisTitleOffset(&gp,framenum,axisnum,offset);
```

SOURCE gp.src

SEE ALSO **gpSetYAxisTitle**, **gpSetYAxisTitleColor**, **gpSetYAxisTitleFont**

gpSetZAxisColor

PURPOSE Sets the color of the Z-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisColor**(*pgp*, *framenum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis lines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies only to frames that contain 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 2;
```

```
ret = gpSetZAxisColor(&gp,framenum,"red");
```

SOURCE gp.src

SEE ALSO **gpShowZAxis**, **gpSetZAxisThickness**

gpSetZAxisGridLineColor

PURPOSE Sets the color of the Z-axis major gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisGridLineColor**(*pgp*, *framenum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, major gridlines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies only to frames that contain 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
ret = gpSetZAxisGridLineColor(&gp,framenum,"blue");
```

SOURCE gp.src

SEE ALSO **gpShowZAxisGridLines**, **gpSetZAxisGridLineThickness**, **gpSetZAxisGridPattern**

gpSetZAxisGridLineThickness

PURPOSE Sets the thickness of the Z-axis major gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisGridLineThickness**(*pgp*, *framenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
thickness	scalar, thickness of the Z-axis major gridlines as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, major gridlines have a line thickness of 0.1. This command applies only to frames that contain 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
thickness = 0.25;
ret = gpSetZAxisGridLineThickness(&gp,framenum,thickness);
```

SOURCE gp.src

SEE ALSO **gpShowZAxisGridLines**, **gpSetZAxisGridLineColor**, **gpSetZAxisGridPattern**

gpSetZAxisGridPattern

PURPOSE Sets the pattern of the Z-axis major gridlines.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisGridPattern**(*pgp*, *framenum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, major gridlines are solid. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**. This command applies only to frames that contain 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
ret = gpSetZAxisGridPattern(&gp,framenum,"dashdot");
```

SOURCE `gp.src`

SEE ALSO **gpSetZAxisGridPatternLength**, **gpShowZAxisGridLines**, **gpSetZAxisGridLineColor**

gpSetZAxisGridPatternLength

PURPOSE Sets the length of the Z-axis major grid line pattern.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisGridPatternLength**(*pgp*, *framenum*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
length	scalar, length of one cycle of the line pattern as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Major gridlines are drawn as solid lines by default. If a different line pattern is selected with **gpSetZAxisGridLinePattern**, then the default pattern length is 2. This command applies only to frames that contain 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
length = 4;
ret = gpSetZAxisGridPattern(&gp,framenum,"dashdot");
ret = gpSetZAxisGridPatternLength(&gp,framenum,length);
```

SOURCE `gp.src`

SEE ALSO **gpSetZAxisGridPattern**, **gpShowZAxisGridLines**, **gpSetZAxisGridLineColor**

gpSetZAxisLabelColor

PURPOSE Sets the color of the Z-axis tick labels in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisLabelColor**(*pgp*, *framenum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies only to frames that contain 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
ret = gpSetZAxisLabelColor(&gp,framenum,"blue");
```

SOURCE `gp.src`

SEE ALSO **gpSetZAxisLabelFont**, **gpSetZAxisLabelOffset**, **gpSetZAxisLabelSkip**

gpSetZAxisLabelFont

PURPOSE Sets the type and size of the font used for the Z-axis tick labels in the specified frame.

gpSetZAxisLabelFont

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisLabelFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame **2** font height in points. **3** font height as a percentage of the length of the axis line (valid only for 3D Cartesian and 3D Contour plots).

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels are written in a helvetica font (*fonttype* = "helv"). The default font height is equal to 3% of the axis length for 3D Cartesian and 3D Contour plots and equal to 3% of the frame height for all other plot types. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
fontheight = 4;
heightunit = 1;
ret = gpSetZAxisLabelFont(&gp,framenum,"courier",fontheight,heightunit
```

SOURCE gp.src

SEE ALSO **gpSetZAxisLabelColor**, **gpSetZAxisLabelOffset**, **gpSetZAxisLabelSkip**

gpSetZAxisLabelOffset

PURPOSE Sets the offset of the Z-axis tick labels from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisLabelOffset**(*pgp*, *framenum*, *offset*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
offset	scalar, offset of tick labels from axis line as a percentage of the median axis length.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels have an offset of 1. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
```

```
offset = 3;
ret = gpSetZAxisLabelOffset(&gp,framenum,offset);
```

SOURCE gp.src

SEE ALSO **gpSetZAxisLabelColor**, **gpSetZAxisLabelFont**, **gpSetZAxisLabelSkip**

gpSetZAxisLabelSkip

PURPOSE Sets an option to show only one in every *skipnum* Z-axis tick labels in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisLabelSkip**(*pgp*, *framenum*, *skipnum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
skipnum	scalar, show only one in every *skipnum* Z-axis tick labels.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis tick labels are shown (*skipnum* = 1). This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 3;
ret = gpSetZAxisLabelSkip(&gp,framenum,2);
```

SOURCE gp.src

SEE ALSO **gpSetZAxisLabelColor**, **gpSetZAxisLabelFont**, **gpSetZAxisLabelOffset**

gpSetZAxisMinorGridLineColor

PURPOSE Sets the color of the Z-axis minor gridlines.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisMinorGridLineColor**(*pgp*, *framenum*, *linecolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linecolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, minor gridlines are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
show = 1;
ret = gpShowZAxisMinorGridLines(&gp,framenum,show);
ret = gpSetZAxisMinorGridLineColor(&gp,framenum,"blue");
```

SOURCE `gp.src`

SEE ALSO **gpShowZAxisMinorGridLines**, **gpSetZAxisMinorGridLineThickness**, **gpSetZAxisMinorGridPattern**

gpSetZAxisMinorGridLineThickness

PURPOSE Sets the thickness of the Z-axis minor gridlines.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisMinorGridLineThickness**(*pgp*, *framenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
thickness	scalar, thickness of the Z-axis minor gridlines as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, minor gridlines have a line thickness of 0.1. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
show = 1;
thickness = 0.25;
ret = gpShowZAxisMinorGridLines(&gp,framenum,show);
ret = gpSetZAxisMinorGridLineThickness(&gp,framenum,thickness);
```

SOURCE gp.src

SEE ALSO **gpShowZAxisMinorGridLines**, **gpSetZAxisMinorGridLineColor**, **gpSetZAxisMinorGridPattern**

gpSetZAxisMinorGridPattern

PURPOSE Sets the pattern of the Z-axis minor gridlines.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisMinorGridPattern**(*pgp*, *framenum*, *pattern*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
pattern	string, name of line pattern. – or – scalar, index of line pattern in pattern list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, minor gridlines are dotted. The list of available line patterns is as follows:

1 Solid
2 Dashed
3 DashDot
4 Dotted
5 LongDash
6 DashDotDot

Line pattern names are case insensitive in **GAUSSplot**. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
show = 1;
ret = gpShowZAxisMinorGridLines(&gp,framenum,show);
ret = gpSetZAxisMinorGridPattern(&gp,framenum,"dashdot");
```

SOURCE gp.src

SEE ALSO **gpSetZAxisMinorGridPatternLength**, **gpShowZAxisMinorGridLines**, **gpSetZAxisMinorGridLineColor**

gpSetZAxisMinorGridPatternLength

PURPOSE Sets the length of the Z-axis minor grid line pattern.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisMinorGridPatternLength**(*pgp*, *framenum*, *length*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
length	scalar, length of one cycle of the line pattern as a percentage of the frame height.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Minor gridlines are drawn as dotted lines with a pattern length of 2 by default. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
show = 1;
length = 4;
ret = gpShowZAxisMinorGridLines(&gp,framenum,show);
ret = gpSetZAxisMinorGridPattern(&gp,framenum,"dashdotdot");
ret = gpSetZAxisMinorGridPatternLength(&gp,framenum,length);
```

SOURCE gp.src

SEE ALSO **gpSetZAxisMinorGridPattern**, **gpShowZAxisMinorGridLines**, **gpSetZAxisMinorGridLineColor**

gpSetZAxisRange

PURPOSE Sets the range of the Z-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisRange**(*pgp*, *framenum*, *min*, *max*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
min	scalar, minimum value for the Z-axis.
max	scalar, maximum value for the Z-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the range of the Z-axis is calculated based on the values of the data points. If the axes are dependent on one another, which is the default for 3D Cartesian and 3D Contour plots, then the range of the Z-axis will be determined by the X, Y and Z values of the data points. Otherwise, the range will be determined by only the Z values. To change the number of major and minor tick marks or the distance between them, call **gpSetZAxisTicks**. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
min = 0;
max = 65;
ret = gpSetZAxisRange(&gp,framenum,min,max);
```

SOURCE gp.src

SEE ALSO **gpSetZAxisTicks**

gpSetZAxisThickness

PURPOSE Sets the thickness of the Z-axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisThickness**(*pgp*, *framenum*, *thickness*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
thickness	scalar, thickness of the axis line as a percentage of the height of the frame.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis lines are set to a thickness of 0.4. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
ret = gpSetZAxisThickness(&gp,framenum,0.8);
```

SOURCE gp.src

SEE ALSO **gpSetZAxisColor**

gpSetZAxisTicks

PURPOSE Sets the distance between major tick marks and the number of minor tick marks on the Z-axis in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisTicks**(*pgp*, *framenum*, *majorstep*, *minornum*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
majorstep	scalar, step between major tick marks in Z-axis units.
minornum	scalar, number of minor tick marks between each set of major tick marks.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the step between major tick marks and the number of minor tick marks on the Z-axis is calculated based on the range of the axis. To modify the range of the Z-axis, call **gpSetZAxisRange**. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
majorstep = 1;
minornum = 4;
ret = gpSetZAxisTicks(&gp,framenum,majorstep,minornum);
```

This example sets the distance between major tick marks on the Z-axis to 1 unit, and the number of minor tick marks between each set of major tick marks to 4. Therefore, there will be 0.2 units from one tick mark to the next.

SOURCE `gp.src`

SEE ALSO **gpSetZAxisRange**

gpSetZAxisTitle

PURPOSE Sets the title for the Z-axis in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisTitle**(*pgp*, *framenum*, *title*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
title	string, title.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the Z-axis is titled with the name of the variable that is associated with it. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
ret = gpSetZAxisTitle(&gp,framenum,"Z");
```

SOURCE gp.src

SEE ALSO **gpSetZAxisTitleColor**, **gpSetZAxisTitleFont**, **gpSetZAxisTitleOffset**

gpSetZAxisTitleColor

PURPOSE Sets the color of the Z-axis title in the specified frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisTitleColor**(*pgp*, *framenum*, *textcolor*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
textcolor	string, name of color. – or – scalar, index of color in color list.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axis titles are black. See Section 2.2.1 for the list of colors; color names are case insensitive in **GAUSSplot**. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 2;
ret = gpSetZAxisTitleColor(&gp,framenum,"red");
```

SOURCE `gp.src`

SEE ALSO **gpSetZAxisTitle**, **gpSetZAxisTitleFont**, **gpSetZAxisTitleOffset**

gpSetZAxisTitleFont

PURPOSE Sets the type and size of the font used for the Z-axis title in the specified frame.

gpSetZAxisTitleFont

LIBRARY `gaussplot`

FORMAT *ret* = **gpSetZAxisTitleFont**(*pgp*, *framenum*, *fonttype*, *height*, *heightunit*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fonttype	string, name of font. – or – scalar, index of font in font list.
height	scalar, font height.
heightunit	scalar, one of the following: **1** font height as a percentage of the height of the frame. **2** font height in points. **3** font height as a percentage of the length of the axis line

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all Z-axis tick labels are written in a bold helvetica font (*fonttype* = "helvbold"). The default font height is equal to 3.6% of the axis length. The list of available fonts is as follows:

1 Courier
2 CourierBold
3 Helv
4 HelvBold
5 Times
6 TimesBold
7 TimesItalic
8 TimesItalicBold
9 Greek
10 Math
11 UserDef

Font names are case insensitive in **GAUSSplot**. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
height = 4.5;
heightunit = 1;
ret = gpSetZAxisTitleFont(&gp,framenum,"timesitalicbold",height,height
```

SOURCE gp.src

SEE ALSO **gpSetZAxisTitle**, **gpSetZAxisTitleColor**, **gpSetZAxisTitleOffset**

gpSetZAxisTitleOffset

PURPOSE Sets the offset of the Z-axis title from the axis line in the specified frame.

LIBRARY gaussplot

FORMAT *ret* = **gpSetZAxisTitleOffset**(*pgp*, *framenum*, *offset*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
offset	scalar, offset of title from axis line as a percentage of the median axis length.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, the offset is set to 8. This command applies only to 3D Cartesian or 3D Contour plots.

EXAMPLE
```
framenum = 2;
offset = 10;
ret = gpSetZAxisTitleOffset(&gp,framenum,offset);
```

SOURCE gp.src

SEE ALSO **gpSetZAxisTitle**, **gpSetZAxisTitleColor**, **gpSetZAxisTitleFont**

gpShow3DOrientationAxis

PURPOSE Activates or deactivates display of the 3D orientation axis in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShow3DOrientationAxis**(*pgp*, *framenum*, *show3dorientation*);

INPUT

pgp pointer to a plot control structure.

framenum scalar, frame number.

show3dorientation scalar, one of the following:

0 do not display the 3D orientation axis.

1 display the 3D orientation axis.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS The 3D orientation axis is the small set of axes that appear in the upper-right hand corner of 3D Cartesian and 3D Contour plots by default to show you how your plot is oriented. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
noshow = 0;
ret = gpShow3DOrientationAxis(&gp,framenum,noshow);
```

SOURCE gp.src

SEE ALSO **gpShowXAxis**, **gpShowYAxis**, **gpShowZAxis**

gpShowBarLayer

PURPOSE Activates or deactivates the bar layer in the specified frame(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowBarLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
activate	scalar, one of the following:
	0 deactivate the bar layer.
	1 activate the bar layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

By default the bar layer is activated in frames containing Bar plots. Calling **gpShowBarLayer** activates or deactivates the bar layer in the specified frame. To activate or deactivate the bar layer for a subset of the line maps in a frame, call **gpShowLineMapBarLayer**.

This command applies ONLY to line plots.

EXAMPLE
```
framenum = 1;
activate = 1;
ret = gpShowBarLayer(&gp,framenum,activate);
```

SOURCE `gplinemap.src`

SEE ALSO **gpShowLineMapBarLayer**, **gpAddBarFrame**, **gpSetBarColor**

gpShowBoundaryLayer

PURPOSE Activates or deactivates the boundary layer in the specified frame(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowBoundaryLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
activate	scalar, one of the following:
	0 deactivate the boundary layer.
	1 activate the boundary layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The boundary layer is displayed as a line connecting all data points around the edge of the data zone. In other words, suppose that a field zone describes the set of points (X,Y), where X and Y are both 3×3 variables:

X[1,1]	X[1,2]	X[1,3]	Y[1,1]	Y[1,2]	Y[1,3]
X[2,1]	X[2,2]	X[2,3]	Y[2,1]	Y[2,2]	Y[2,3]
X[3,1]	X[3,2]	X[3,3]	Y[3,1]	Y[3,2]	Y[3,3]

Then the boundary line for that field zone would be a line through the following points: (X[1,1],Y[1,1]), (X[1,2],Y[1,2]), (X[1,3],Y[1,3]), (X[2,3],Y[2,3]), (X[3,3],Y[3,3]), (X[3,2],Y[3,2]), (X[3,1],Y[3,1]), (X[2,1],Y[2,1]), and then back to (X[1,1], Y[1,1]).

For a field zone that describes a 3-D surface, this generally appears as a boundary line around the edge of that surface. By default, the boundary has a

line thickness of 0.4% of the frame height to distinguish it from mesh lines (if present), which have a default line thickness of 0.1% of the frame height.

Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

The boundary layer is activated in frames containing 2D and 3D Cartesian and 2D and 3D Contour plots by default. Calling **gpShowBoundaryLayer** activates or deactivates the boundary layer in the specified frame. To activate or deactivate the boundary layer for a subset of the field zones in a frame, call **gpShowFieldZoneBoundaryLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
activate = 0;
ret = gpShowBoundaryLayer(&gp,framenum,activate);
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowFieldZoneBoundaryLayer**, **gpSetBoundaryColor**

gpShowContourLayer

PURPOSE Activates or deactivates the contour layer in the specified frame(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowContourLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp pointer to a plot control structure.

framenum scalar or N×1 vector, frame number(s).

activate scalar, one of the following:

0 deactivate the contour layer.

1 activate the contour layer.

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

The contour layer is activated in frames containing 2D and 3D Contour plots by default. When the contour layer is activated in a frame, the default is for contour flooding to be displayed in each of the field zones in the frame. Calling **gpShowContourLayer** activates or deactivates the contour layer in the specified frame. To activate or deactivate the contour layer for a subset of the field zones in a frame or to use contour lines, call **gpShowFieldZoneContourLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
activate = 1;
ret = gpShowContourLayer(&gp,framenum,activate);
```

SOURCE

gpfieldzone.src

SEE ALSO

gpShowFieldZoneContourLayer, **gpAdd2DContourFrame**, **gpAdd3DContourFrame**, **gpSetContourVariable**

gpShowDrawing

PURPOSE Specifies whether or not drawing is shown.

LIBRARY `gaussplot`

FORMAT **`gpShowDrawing(`***pgp*`, `*draw*`);`**

INPUT

pgp pointer to a plot control structure.

draw scalar, one of the following:

- **1** do not show drawing; show graph only when all drawing is completed.
- **2** do not show frame drawing, but show each frame after the drawing for that frame is completed.
- **3** show all drawing.

REMARKS By default, *draw* is set to 1, and thus the graph is shown only when all drawing has been completed.

EXAMPLE
```
draw = 2;
gpShowDrawing(&gp,draw);
```

SOURCE `gp.src`

SEE ALSO **`gpFramePause`**

gpShowErrorBarLayer

PURPOSE Activates or deactivates the error bar layer in the specified frame(s).

gpShowFieldZoneBoundaryLayer

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowErrorBarLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.	
framenum	scalar or N×1 vector, frame number(s).	
activate	scalar, one of the following:	
	0	deactivate the error bar layer.
	1	activate the error bar layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

The error bar layer is never activated by default. Calling **gpShowErrorBarLayer** activates or deactivates the error bar layer in the specified frame. To activate or deactivate the error bar layer for a subset of the line maps in a frame, call **gpShowLineMapErrorBarLayer**.

If the error bar layer is activated in a frame, then the second variable in the data file is used to control the magnitudes of the error bars by default. Call **gpSetErrorBarVariable** to use a different variable.

This command applies ONLY to line plots.

EXAMPLE

```
framenum = 1;
activate = 1;
ret = gpShowErrorBarLayer(&gp,framenum,activate);
```

SOURCE `gplinemap.src`

SEE ALSO **gpShowLineMapErrorBarLayer**, **gpSetErrorBarColor**

gpShowFieldZoneBoundaryLayer

PURPOSE Activates or deactivates the boundary layer in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowFieldZoneBoundaryLayer**(*pgp*, *framenum*, *fieldzonenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
activate	scalar or N×1 vector of ones and zeros:
	0 deactivate the boundary layer.
	1 activate the boundary layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The boundary layer is activated by default in frames containing 2D and 3D Cartesian plots and 2D and 3D Contour plots.

Calling **gpShowFieldZoneBoundaryLayer** activates the boundary layer in the frame indicated by *framenum* and then activates or deactivates the boundary layer in the specified field zones depending on how *activate* is set. If *activate* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of *activate*.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure

contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpShowFieldZoneBoundaryLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the boundary layer to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate the boundary layer for all of the field zones that are contained in the plot control structure but not specified in the call to **gpShowFieldZoneBoundaryLayer**.

If you call **gpShowFieldZoneBoundaryLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the boundary layer to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether the boundary layer should be activated or deactivated.

To activate or deactivate the boundary layer in the frame alone, call **gpShowBoundaryLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
fieldzonenum = { 1,2,3,4,5 };
activate = { 1,0,0,1,1 };
ret = gpShowFieldZoneBoundaryLayer(&gp,framenum,fieldzonenum,activate);
```

This example deactivates the boundary layer for the second and third field zones in the frame, and activates it for the first, fourth and fifth field zones.

SOURCE gpfieldzone.src

SEE ALSO **gpShowBoundaryLayer**, **gpSetBoundaryColor**, **gpSetBoundaryThickness**

gpShowFieldZoneContourLayer

PURPOSE Activates or deactivates contour lines and flooding in the specified field zone(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowFieldZoneContourLayer**(*pgp*, *framenum*, *fieldzonenum*, *showflood*, *showlines*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
showflood	scalar or N×1 vector of ones and zeros: **0** deactivate contour flooding. **1** activate contour flooding.
showlines	scalar or N×1 vector of ones and zeros: **0** deactivate contour lines. **1** activate contour lines.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The contour layer is activated by default in frames containing 2D and 3D Contour plots. When the contour layer is activated in a frame, the default is for contour flooding to be displayed in each of the field zones in the frame.

Calling **gpShowFieldZoneContourLayer** activates the contour layer in the frame indicated by *framenum* and then activates or deactivates contour flooding and contour lines in the specified field zones depending on how *showflood* and *showlines* are set. If either *showflood* or *showlines* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of that vector.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpShowFieldZoneContourLayer** for a subset of the field zones in the frame with either *showflood* or *showlines* set to a scalar one (or a vector of zeros and ones), then it will assume that you want the contour layer to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate the contour layer for all of the field zones that are contained in the plot control structure but not specified in the call to **gpShowFieldZoneContourLayer**.

If you call **gpShowFieldZoneContourLayer** for a subset of the field zones in the frame with *showflood* or *showlines* set to a scalar zero, then it will assume that you want the contour layer to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when the contour layer is activated in a frame, then the contour layer is activated for each line map or field zone in that frame. In this case, contour flooding will be displayed in all of the unspecified frames because that is the default.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether contour flooding and/or contour lines should be activated or deactivated.

To activate or deactivate the contour layer in the frame alone, call **gpShowContourLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE
```
framenum = 1;
fieldzonenum = { 1,2,3,4,5 };
showflooding = { 1,0,0,1,1 };
showlines = { 0,1,0,1,0 };
ret = gpShowFieldZoneContourLayer(&gp,framenum,fieldzonenum,
                                  showflooding,showlines);
```

This example activates contour flooding for the first, fourth and fifth field zones in the frame and activates contour lines for the second and fourth field zones.

SOURCE gpcontourplot.src

SEE ALSO **gpShowContourLayer**, **gpSetContourFloodGroup**, **gpSetContourLineGroup**, **gpSetContourVariable**

gpShowFieldZoneMeshLayer

PURPOSE Activates or deactivates the mesh layer in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowFieldZoneMeshLayer**(*pgp*, *framenum*, *fieldzonenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
activate	scalar or N×1 vector of ones and zeros: **0** deactivate the mesh layer. **1** activate the mesh layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The mesh layer is activated by default in frames containing 2D and 3D Cartesian plots.

Calling **gpShowFieldZoneMeshLayer** activates the mesh layer in the frame indicated by *framenum* and then activates or deactivates the mesh layer in the specified field zones depending on how *activate* is set. If *activate* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of *activate*.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpShowFieldZoneMeshLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the mesh layer to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate the mesh layer for all of the field zones that are contained in the plot control structure but not specified in the call to **gpShowFieldZoneMeshLayer**.

If you call **gpShowFieldZoneMeshLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the mesh layer to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether the mesh layer should be activated or deactivated.

To activate or deactivate the mesh layer in the frame alone, call **gpShowMeshLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
fieldzonenum = 2;
activate = 0;
ret = gpShowFieldZoneMeshLayer(&gp,framenum,fieldzonenum,activate);
```

This example deactivates the mesh layer in the second field zone. The mesh layer will still be displayed in all other field zones in the frame.

SOURCE gpfieldzone.src

SEE ALSO **gpShowMeshLayer**, **gpSetMeshColor**, **gpSetMeshLineThickness**

gpShowFieldZoneScatterLayer

PURPOSE Activates or deactivates the scatter layer in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowFieldZoneScatterLayer**(*pgp*, *framenum*, *fieldzonenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
activate	scalar or N×1 vector of ones and zeros: **0** deactivate the scatter layer. **1** activate the scatter layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The scatter layer is never activated by default.

Calling **gpShowFieldZoneScatterLayer** activates the scatter layer in the frame indicated by *framenum* and then activates or deactivates the scatter layer in the specified field zones depending on how *activate* is set. If *activate* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of *activate*.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpShowFieldZoneScatterLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the scatter layer to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate the scatter layer for all of the field zones that are contained in the plot control structure but not specified in the call to **gpShowFieldZoneScatterLayer**.

If you call **gpShowFieldZoneScatterLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the scatter layer to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether the scatter layer should be activated or deactivated.

To activate or deactivate the scatter layer in the frame alone, call

gpShowScatterLayer.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
fieldzonenum = { 1,2,4 };
activate = 1;
ret = gpShowFieldZoneScatterLayer(&gp,framenum,fieldzonenum,activate);
```

SOURCE gpfieldzone.src

SEE ALSO gpShowScatterLayer, gpSetScatterColor, gpSetScatterShape

gpShowFieldZoneShadeLayer

PURPOSE Activates or deactivates the shade layer in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowFieldZoneShadeLayer**(*pgp*, *framenum*, *fieldzonenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
activate	scalar or N×1 vector of ones and zeros: **0** deactivate the shade layer. **1** activate the shade layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The shade layer is never activated by default.

Calling **gpShowFieldZoneShadeLayer** activates the shade layer in the frame indicated by *framenum* and then activates or deactivates the shade layer in the specified field zones depending on how *activate* is set. If *activate* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of *activate*.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpShowFieldZoneShadeLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the shade layer to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate the shade layer for all of the field zones that are contained in the plot control structure but not specified in the call to **gpShowFieldZoneShadeLayer**.

If you call **gpShowFieldZoneShadeLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the shade layer to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether the shade layer should be activated or deactivated.

To activate or deactivate the shade layer in the frame alone, call

gpShowShadeLayer.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE
```
framenum = 1;
fieldzonenum = { 3,5 };
activate = 1;
ret = gpShowFieldZoneShadeLayer(&gp,framenum,fieldzonenum,activate);
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowShadeLayer**, **gpSetShadeColor**, **gpUseFieldZoneShadeLighting** **gpUseTranslucencyEffect**,

gpShowFieldZoneVectorLayer

PURPOSE Activates or deactivates the vector layer in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowFieldZoneVectorLayer**(*pgp*, *framenum*, *fieldzonenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
activate	scalar or N×1 vector of ones and zeros: **0** deactivate the vector layer. **1** activate the vector layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The vector layer is never activated by default.

Calling **gpShowFieldZoneVectorLayer** activates the vector layer in the frame indicated by *framenum* and then activates or deactivates the vector layer in the specified field zones depending on how *activate* is set. If *activate* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of *activate*.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpShowFieldZoneVectorLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the vector layer to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate the vector layer for all of the field zones that are contained in the plot control structure but not specified in the call to **gpShowFieldZoneVectorLayer**.

If you call **gpShowFieldZoneVectorLayer** for a subset of the field zones in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the vector layer to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether the vector layer should be activated or deactivated.

To activate or deactivate the vector layer in the frame alone, call

gpShowVectorLayer.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
fieldzonenum = { 1,2,3,4,5 };
activate = { 0,0,1,0,1 };
ret = gpShowFieldZoneVectorLayer(&gp,framenum,fieldzonenum,activate);
```

This example activates the vector layer for only the third and fifth field zones in the frame.

SOURCE gpfieldzone.src

SEE ALSO **gpShowVectorLayer**, **gpSetVectorColor**, **gpSetVectorType**

gpShowLineLayer

PURPOSE Activates or deactivates the line layer in the specified frame(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowLineLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
activate	scalar, one of the following:
	0 deactivate the line layer.
	1 activate the line layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

By default the line layer is activated in frames containing XY Line and Polar Line plots. Calling **gpShowLineLayer** activates or deactivates the line layer in the specified frame. To activate or deactivate the line layer for a subset of the line maps in a frame, call **gpShowLineMapLineLayer**.

This command applies ONLY to Bar, XY Line, and Polar Line plots.

EXAMPLE
```
framenum = 1;
activate = 1;
ret = gpShowSymbolLayer(&gp,framenum,activate);
activate = 0;
ret = gpShowLineLayer(&gp,framenum,activate);
```

SOURCE gplinemap.src

SEE ALSO **gpShowLineMapLineLayer**, **gpAddXYLineFrame**, **gpSetLineColor**

gpShowLineMapBarLayer

PURPOSE Activates or deactivates the bar layer in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowLineMapBarLayer**(*pgp*, *framenum*, *linemapnum*, *activate*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	linemapnum	scalar or N×1 vector, line map number(s).
	activate	scalar or N×1 vector of ones and zeros: **0** deactivate the bar layer. **1** activate the bar layer.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The bar layer is activated by default in frames containing Bar plots.

Calling **gpShowLineMapBarLayer** activates the bar layer in the frame indicated by *framenum* and then activates or deactivates the bar layer in the specified line maps depending on how *activate* is set. If *activate* is an N×1 vector, then each of the line maps in *linemapnum* will be set according to the corresponding element of *activate*.

If you call **gpShowLineMapBarLayer** for a subset of the line maps in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the bar layer to be deactivated for all of the unspecified line maps in the frame. Therefore, it will deactivate the bar layer for all of the line maps that are not specified in the call to **gpShowLineMapBarLayer**.

If you call **gpShowLineMapBarLayer** for a subset of the line maps in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the bar layer to be activated for all of the other line maps in the frame. In this case, it will not set any options for the unspecified line maps because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

To activate or deactivate the bar layer in the frame alone, call **gpShowBarLayer**.

This command applies ONLY to line plots.

EXAMPLE
```
framenum = 1;
linemapnum = { 1,4,5 };
activate = 0;
ret = gpShowLineMapBarLayer(&gp,framenum,linemapnum,activate);
```

SOURCE gplinemap.src

SEE ALSO **gpAddBarFrame**, **gpShowBarLayer**, **gpSetBarColor**

gpShowLineMapErrorBarLayer

PURPOSE Activates or deactivates the error bar layer in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowLineMapErrorBarLayer(***pgp*, *framenum*, *linemapnum*, *activate***);**

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
activate	scalar or N×1 vector of ones and zeros:
	0 deactivate the error bar layer.
	1 activate the error bar layer.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a

frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The error bar layer is never activated by default.

Calling **gpShowLineMapErrorBarLayer** activates the error bar layer in the frame indicated by *framenum* and then activates or deactivates the error bar layer in the specified line maps depending on how *activate* is set. If *activate* is an N×1 vector, then each of the line maps in *linemapnum* will be set according to the corresponding element of *activate*.

If you call **gpShowLineMapErrorBarLayer** for a subset of the line maps in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the error bar layer to be deactivated for all of the unspecified line maps in the frame. Therefore, it will deactivate the error bar layer for all of the line maps that are not specified in the call to **gpShowLineMapErrorBarLayer**.

If you call **gpShowLineMapErrorBarLayer** for a subset of the line maps in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the error bar layer to be activated for all of the other line maps in the frame. In this case, it will not set any options for the unspecified line maps because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

To activate or deactivate the error bar layer in the frame alone, call **gpShowErrorBarLayer**.

This command applies ONLY to line plots.

EXAMPLE

```
framenum = 1;
linemapnum = { 2,5 };
activate = 1;
ret = gpShowLineMapErrorBarLayer(&gp,framenum,linemapnum,activate);
```

SOURCE gplinemap.src

SEE ALSO **gpShowErrorBarLayer**, **gpSetErrorBarColor**, **gpSetErrorBarSize**

gpShowLineMapLineLayer

PURPOSE Activates or deactivates the line layer in the specified line map(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowLineMapLineLayer**(*pgp*, *framenum*, *linemapnum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
linemapnum	scalar or N×1 vector, line map number(s).
activate	scalar or N×1 vector of ones and zeros:
	0 deactivate the line layer.
	1 activate the line layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The line layer is activated in frames containing XY Line and Polar Line plots by default.

Calling **gpShowLineMapLineLayer** activates the line layer in the frame indicated by *framenum* and then activates or deactivates the line layer in the specified line maps depending on how *activate* is set. If *activate* is an N×1 vector, then each of the line maps in *linemapnum* will be set according to the corresponding element of *activate*.

If you call **gpShowLineMapLineLayer** for a subset of the line maps in the frame with an *activate* argument set to a scalar one (or a vector of zeros and

ones), then it will assume that you want the line layer to be deactivated for all of the unspecified line maps in the frame. Therefore, it will deactivate the line layer for all of the line maps that are not specified in the call to **gpShowLineMapLineLayer**.

If you call **gpShowLineMapLineLayer** for a subset of the line maps in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the line layer to be activated for all of the other line maps in the frame. In this case, it will not set any options for the unspecified line maps because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

To activate or deactivate the line layer in the frame alone, call **gpShowLineLayer**.

This command applies ONLY to Bar, XY Line, and Polar Line plots.

EXAMPLE

```
framenum = 1;
linemapnum = { 1,4,5 };
activate = 0;
ret = gpShowLineMapLineLayer(&gp,framenum,linemapnum,activate);
```

SOURCE gplinemap.src

SEE ALSO **gpAddXYLineFrame**, **gpShowLineLayer**, **gpSetLineColor**

gpShowLineMapSymbolLayer

PURPOSE Activates or deactivates the symbol layer in the specified line map(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowLineMapSymbolLayer**(*pgp*, *framenum*, *linemapnum*, *activate*);

INPUT

pgp pointer to a plot control structure.

framenum scalar, frame number.

linemapnum scalar or N×1 vector, line map number(s).

activate scalar or N×1 vector of ones and zeros:

0 deactivate the symbol layer.

1 activate the symbol layer.

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings. The symbol layer is never activated by default.

Calling **gpShowLineMapSymbolLayer** activates the symbol layer in the frame indicated by *framenum* and then activates or deactivates the symbol layer in the specified line maps depending on how *activate* is set. If *activate* is an N×1 vector, then each of the line maps in *linemapnum* will be set according to the corresponding element of *activate*.

If you call **gpShowLineMapSymbolLayer** for a subset of the line maps in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the symbol layer to be deactivated for all of the unspecified line maps in the frame. Therefore, it will deactivate the symbol layer for all of the line maps that are not specified in the call to **gpShowLineMapSymbolLayer**.

If you call **gpShowLineMapSymbolLayer** for a subset of the line maps in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the symbol layer to be activated for all of the other line maps in the frame. In this case, it will not set any options for the unspecified line maps because, by default, when a layer is activated in a frame, it is activated for each line map or field zone in that frame.

To activate or deactivate the symbol layer in the frame alone, call **gpShowSymbolLayer**.

This command applies ONLY to Bar, XY Line, and Polar Line plots.

EXAMPLE

```
framenum = 1;
linemapnum = { 1,4,5 };
activate = 1;
ret = gpShowLineMapSymbolLayer(&gp,framenum,linemapnum,activate);
```

SOURCE gplinemap.src

SEE ALSO **gpShowSymbolLayer**, **gpSetSymbolColor**, **gpSetSymbolShape**

gpShowMeshLayer

PURPOSE Activates or deactivates the mesh layer in the specified frame(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowMeshLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
activate	scalar, one of the following:
	0 deactivate the mesh layer.
	1 activate the mesh layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by

default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

By default the mesh layer is activated in frames containing 2D and 3D Cartesian plots. Calling **gpShowMeshLayer** activates or deactivates the mesh layer in the specified frame. To activate or deactivate the mesh layer for a subset of the field zones in a frame, call **gpShowFieldZoneMeshLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
activate = 0;
ret = gpShowMeshLayer(&gp,framenum,activate);
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowFieldZoneMeshLayer**, **gpSetMeshColor**, **gpSetMeshLineThickness**

gpShowPolarAxes

PURPOSE Specifies which of the polar axes are to be displayed in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowPolarAxes**(*pgp*, *framenum*, *showtheta*, *showr*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
showtheta	scalar, one of the following:
	0 do not display the Theta-axis.

1 display the Theta-axis.

showr scalar, one of the following:

0 do not display the R-axis.

1 display the R-axis.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, both the Theta and R axes are displayed. This command applies ONLY to frames that contain Polar Line plots.

EXAMPLE
```
framenum = 1;
showtheta = 1;
showr = 0;
ret = gpShowPolarAxes(&gp,framenum,showtheta,showr);
```

SOURCE gppolarlineplot.src

SEE ALSO **gpSetPolarAxesTitles**

gpShowScatterLayer

PURPOSE Activates or deactivates the scatter layer in the specified frame(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowScatterLayer**(*pgp*, *framenum*, *activate*);

INPUT *pgp* pointer to a plot control structure.

framenum scalar or N×1 vector, frame number(s).

activate scalar, one of the following:

0 deactivate the scatter layer.

1 activate the scatter layer.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

The scatter layer is never activated by default. Calling **gpShowScatterLayer** activates or deactivates the scatter layer in the specified frame. To activate or deactivate the scatter layer for a subset of the field zones in a frame, call **gpShowFieldZoneScatterLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE
```
framenum = 1;
activate = 1;
ret = gpShowScatterLayer(&gp,framenum,activate);
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowFieldZoneScatterLayer**, **gpSetScatterColor**, **gpSetScatterShape**

gpShowShadeLayer

PURPOSE Activates or deactivates the shade layer in the specified frame(s).

LIBRARY gaussplot

FORMAT *ret* = **gpShowShadeLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp pointer to a plot control structure.

framenum scalar or N×1 vector, frame number(s).

activate scalar, one of the following:

0 deactivate the shade layer.

1 activate the shade layer.

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

The shade layer is never activated by default. Calling **gpShowShadeLayer** activates or deactivates the shade layer in the specified frame. To activate or deactivate the shade layer for a subset of the field zones in a frame, call **gpShowFieldZoneShadeLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
activate = 1;
ret = gpShowShadeLayer(&gp,framenum,activate);
```

SOURCE

gpfieldzone.src

SEE ALSO

gpShowFieldZoneShadeLayer, **gpSetShadeColor**, **gpUseFieldZoneShadeLighting**, **gpUseTranslucencyEffect**

gpShowSymbolLayer

PURPOSE Activates or deactivates the symbol layer in the specified frame(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowSymbolLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
activate	scalar, one of the following:
	0 deactivate the symbol layer.
	1 activate the symbol layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

The symbol layer is never activated by default. Calling **gpShowSymbolLayer** activates or deactivates the symbol layer in the specified frame. To activate or deactivate the symbol layer for a subset of the line maps in a frame, call **gpShowLineMapSymbolLayer**.

This command applies ONLY to Bar, XY Line and Polar Line plots.

EXAMPLE
```
framenum = 1;
activate = 1;
ret = gpShowSymbolLayer(&gp,framenum,activate);
```

SOURCE `gplinemap.src`

SEE ALSO **gpShowLineMapSymbolLayer**, **gpSetSymbolColor**, **gpSetSymbolShape**

gpShowVectorLayer

PURPOSE Activates or deactivates the vector layer in the specified frame(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowVectorLayer**(*pgp*, *framenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
activate	scalar, one of the following:
	0 deactivate the vector layer.
	1 activate the vector layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Layers are controlled on two levels: in a frame and in each of the line maps or field zones in the frame. If a layer is activated in a frame, then it is displayed by default in each line map or field zone in the frame. If a layer is not activated in a frame, then it is not displayed in any of the line maps or field zones in the frame, regardless of their specific settings.

The vector layer is never activated by default. Calling **gpShowVectorLayer** activates or deactivates the vector layer in the specified frame. To activate or deactivate the vector layer for a subset of the field zones in a frame, call **gpShowFieldZoneVectorLayer**.

This command applies ONLY to 2D and 3D Cartesian and 2D and 3D Contour plots.

EXAMPLE

```
framenum = 1;
```

```
activate = 1;
ret = gpShowVectorLayer(&gp,framenum,activate);
```

SOURCE gpfieldzone.src

SEE ALSO **gpShowFieldZoneVectorLayer**, **gpSetVectorVariables**, **gpSetVectorColor**

gpShowXAxis

PURPOSE Activates or deactivates display of an X-axis in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowXAxis**(*pgp*, *framenum*, *axisnum*, *showx*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
showx	scalar, one of the following:
	0 do not display X-axis.
	1 display X-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

EXAMPLE
```
framenum = 1;
axisnum = 1;
noshow = 0;
ret = gpShowXAxis(&gp,framenum,axisnum,noshow);
```

SOURCE gp.src

SEE ALSO **gpSetLineMapAxes**, **gpPositionXAxis**, **gpSetXAxisColor**, **gpSetXAxisTitle**

gpShowXAxisGridLines

PURPOSE Activates or deactivates display of X-axis major gridlines in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowXAxisGridLines**(*pgp*, *framenum*, *axisnum*, *showgl*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
showgl	scalar, one of the following:
	0 do not display X-axis major gridlines.
	1 display X-axis major gridlines.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an

X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, major gridlines are displayed only in 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowXAxisGridLines(&gp,framenum,axisnum,show);
```

SOURCE gp.src

SEE ALSO **gpShowXAxis**, **gpShowXAxisMinorGridLines**, **gpSetXAxisGridLineColor**

gpShowXAxisLabels

PURPOSE Activates or deactivates display of X-axis tick labels in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowXAxisLabels**(*pgp*, *framenum*, *axisnum*, *showtl*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
showtl	scalar, one of the following:
	0 do not display X-axis tick labels.
	1 display X-axis tick labels.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, tick labels are displayed.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 0;
ret = gpShowXAxisLabels(&gp,framenum,axisnum,show);
```

SOURCE gp.src

SEE ALSO **gpShowXAxis**, **gpShowXAxisTicks**

gpShowXAxisMinorGridLines

PURPOSE Activates or deactivates display of X-axis minor gridlines in a frame.

LIBRARY gausplot

FORMAT *ret* = **gpShowXAxisMinorGridLines**(*pgp*, *framenum*, *axisnum*, *showgl*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
showgl	scalar, one of the following:
	0 do not display X-axis minor gridlines.

1 display X-axis minor gridlines.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, minor gridlines are not displayed.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowXAxisMinorGridLines(&gp,framenum,axisnum,show);
```

SOURCE gp.src

SEE ALSO **gpShowXAxis**, **gpShowXAxisGridLines**, **gpSetXAxisMinorGridLineColor**

gpShowXAxisTicks

PURPOSE Activates or deactivates display of X-axis tick marks in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowXAxisTicks**(*pgp*, *framenum*, *axisnum*, *showtm*);

INPUT
pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.

showtm scalar, one of the following:

0 do not display X-axis tick marks.

1 display X-axis tick marks.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, tick marks are displayed.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 0;
ret = gpShowXAxisTicks(&gp,framenum,axisnum,show);
```

SOURCE gp.src

SEE ALSO **gpShowXAxis**, **gpShowXAxisLabels**

gpShowYAxis

PURPOSE Activates or deactivates display of a Y-axis in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowYAxis**(*pgp*, *framenum*, *axisnum*, *showy*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
	showy	scalar, one of the following:
		0 do not display Y-axis.
		1 display Y-axis.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

EXAMPLE
```
framenum = 1;
axisnum = 1;
noshow = 0;
ret = gpShowYAxis(&gp,framenum,axisnum,noshow);
```

SOURCE gp.src

SEE ALSO **gpSetLineMapAxes**, **gpPositionYAxis**, **gpSetYAxisColor**, **gpSetYAxisTitle**

gpShowYAxisGridLines

PURPOSE Activates or deactivates display of Y-axis major gridlines in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowYAxisGridLines**(*pgp*, *framenum*, *axisnum*, *showgl*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
showgl	scalar, one of the following:
	0 do not display Y-axis major gridlines.
	1 display Y-axis major gridlines.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, major gridlines are displayed only in 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowYAxisGridLines(&gp,framenum,axisnum,show);
```

SOURCE gp.src

SEE ALSO **gpShowYAxis**, **gpShowYAxisMinorGridLines**, **gpSetYAxisGridLineColor**

gpShowYAxisLabels

PURPOSE Activates or deactivates display of Y-axis tick labels in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowYAxisLabels**(*pgp*, *framenum*, *axisnum*, *showtl*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
showtl	scalar, one of the following:
	0 do not display Y-axis tick labels.
	1 display Y-axis tick labels.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, tick labels are displayed.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 0;
ret = gpShowYAxisLabels(&gp,framenum,axisnum,show);
```

SOURCE gp.src

SEE ALSO **gpShowYAxis**, **gpShowYAxisTicks**

gpShowYAxisMinorGridLines

PURPOSE Activates or deactivates display of Y-axis minor gridlines in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowYAxisMinorGridLines**(*pgp*, *framenum*, *axisnum*, *showgl*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
showgl	scalar, one of the following:
	0 do not display Y-axis minor gridlines.
	1 display Y-axis minor gridlines.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, minor gridlines are not displayed.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 1;
ret = gpShowYAxisMinorGridLines(&gp,framenum,axisnum,show);
```

SOURCE gp.src

SEE ALSO **gpShowYAxis**, **gpShowYAxisGridLines**, **gpSetYAxisMinorGridLineColor**

gpShowYAxisTicks

PURPOSE Activates or deactivates display of Y-axis tick marks in a frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowYAxisTicks**(*pgp*, *framenum*, *axisnum*, *showtm*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
showtm	scalar, one of the following:
	0 do not display Y-axis tick marks.
	1 display Y-axis tick marks.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, tick marks are displayed.

EXAMPLE
```
framenum = 1;
axisnum = 1;
show = 0;
ret = gpShowYAxisTicks(&gp,framenum,axisnum,show);
```

SOURCE `gp.src`

SEE ALSO **gpShowYAxis**, **gpShowYAxisLabels**

gpShowZAxis

PURPOSE Activates or deactivates display of the Z-axis in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowZAxis**(*pgp*, *framenum*, *showz*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
showz	scalar, one of the following:
	0 do not display Z-axis.
	1 display Z-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, all axes are displayed. This command applies only to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
noshow = 0;
ret = gpShowZAxis(&gp,framenum,noshow);
```

SOURCE gp.src

SEE ALSO **gpSetZAxisColor**, **gpSetZAxisTitle**, **gpShow3DOrientationAxis**

gpShowZAxisGridLines

PURPOSE Activates or deactivates display of Z-axis major gridlines in a frame.

LIBRARY `gaussplot`

FORMAT *ret* = **gpShowZAxisGridLines**(*pgp*, *framenum*, *showgl*);

INPUT

pgp	pointer to a plot control structure.	
framenum	scalar, frame number.	
showgl	scalar, one of the following:	
	0	do not display Z-axis major gridlines.
	1	display Z-axis major gridlines.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, major gridlines are displayed in 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
noshow = 0;
ret = gpShowZAxisGridLines(&gp,framenum,show);
```

SOURCE `gp.src`

SEE ALSO **gpShowZAxis**, **gpShowZAxisMinorGridLines**, **gpSetZAxisGridLineColor**

gpShowZAxisLabels

PURPOSE Activates or deactivates display of Z-axis tick labels in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowZAxisLabels**(*pgp*, *framenum*, *showtl*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
showtl	scalar, one of the following:
	0 do not display Z-axis tick labels.
	1 display Z-axis tick labels.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, tick labels are displayed.

EXAMPLE
```
framenum = 1;
show = 0;
ret = gpShowZAxisLabels(&gp,framenum,show);
```

SOURCE gp.src

SEE ALSO **gpShowZAxis**, **gpShowZAxisTicks**

gpShowZAxisMinorGridLines

PURPOSE Activates or deactivates display of Z-axis minor gridlines in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowZAxisMinorGridLines**(*pgp*, *framenum*, *showgl*);

INPUT

pgp	pointer to a plot control structure.

	framenum	scalar, frame number.
	showgl	scalar, one of the following:
		0 do not display Z-axis minor gridlines.
		1 display Z-axis minor gridlines.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, minor gridlines are not displayed.

EXAMPLE
```
framenum = 1;
show = 1;
ret = gpShowZAxisMinorGridLines(&gp,framenum,show);
```

SOURCE gp.src

SEE ALSO **gpShowZAxis**, **gpShowZAxisGridLines**, **gpSetZAxisMinorGridLineColor**

gpShowZAxisTicks

PURPOSE Activates or deactivates display of Z-axis tick marks in a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpShowZAxisTicks**(*pgp*, *framenum*, *showtm*);

INPUT	*pgp*	pointer to a plot control structure.
	framenum	scalar, frame number.
	showtm	scalar, one of the following:
		0 do not display Z-axis tick marks.
		1 display Z-axis tick marks.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, tick marks are displayed.

EXAMPLE
```
framenum = 1;
show = 0;
ret = gpShowZAxisTicks(&gp,framenum,show);
```

SOURCE gp.src

SEE ALSO **gpShowZAxis**, **gpShowZAxisLabels**

gpTile

PURPOSE Tiles frames in the workspace.

LIBRARY gaussplot

FORMAT *ret* = **gpTile**(*pgp*, *framesperrow*);

INPUT *pgp* pointer to a plot control structure.

framesperrow N×1 vector or scalar, number of frames in each row in the workspace.

OUTPUT *ret* scalar, return code, 0 if successful, otherwise an error code.

REMARKS If *framesperrow* is an N×1 vector, **gpTile** creates N rows of frames in the workspace, each containing the number of frames specified in the corresponding element of *framesperrow*. If *framesperrow* is a scalar, it places *framesperrow* frames in each row, generating as many rows as are needed to fit all of the frames in the graph. The frames are resized automatically, and are placed in the rows in the order in which they are numbered.

gpTile may be called only after all of the frames have been added to the graph with the appropriate **gpAdd***PlotType***Plot** commands.

By default, **gpTile** leaves 0.25 inch margins between the frames and the edges of the paper. Call **gpSetTileMargins** before **gpTile** to change those margins.

EXAMPLE

```
framesperrow = { 3,2,3 };
ret = gpTile(&gp,framesperrow);
```

This example generates three rows of frames in the graph, placing three frames in the first and third rows and two frames in the second row. The example assumes that a total of 8 frames have been added to the graph.

SOURCE gp.src

SEE ALSO **gpSetTileMargins**, **gpMoveFrame**

gpUse3DPerspective

PURPOSE Activates 3D perspective in a frame so that the shape of the plotted data is dependent on the specified field of view angle.

LIBRARY gaussplot

FORMAT *ret* = **gpUse3DPerspective**(*pgp*, *framenum*, *fieldview*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldview	scalar, 0 or 5.72958e-9 ≤ *fieldview* ≤ 180, field of view angle.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, 3D perspective is not used. If *fieldview* is set to 0, then 3D perspective is activated with a default field of view angle, which is calculated by **GAUSSplot**.

As the field of view angle increases, the size of the plot in the frame will decrease, and as the field of view angle decreases, the size of the plot will increase. To change the size of the plot in the frame without changing the field of view angle (providing that 3D perspective is activated), call **gpSet3DViewDistance** or **gpSet3DViewerPosition**. They change the apparent size of the plot by allowing you to specify the distance between the viewer and the plot.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
 framenum = 1;
fieldviewangle = 45;
 ret = gpUse3DPerspective(&gp,framenum,fieldviewangle);
```

SOURCE gpfieldzone.src

SEE ALSO **gpSet3DViewDistance**, **gpSet3DViewerPosition**

gpUseFieldZoneContourLighting

PURPOSE Activates or deactivates the lighting effect for the contour layer in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpUseFieldZoneContourLighting**(*pgp*, *framenum*, *fieldzonenum*, *activate*);

INPUT *pgp* pointer to a plot control structure.

	framenum	scalar, frame number.
	fieldzonenum	scalar or N×1 vector, field zone number(s).
	activate	scalar or N×1 vector of ones and zeros:
		0 deactivate the lighting effect for the contour layer.
		1 activate the lighting effect for the contour layer.
OUTPUT	*ret*	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The lighting effect is controlled on two levels: in a frame and in each of the field zones in the frame. If the lighting effect is activated in a frame, then both contour and shade lighting are displayed in each field zone in the frame by default. If the lighting effect is not activated in a frame, then neither contour nor shade lighting are displayed in any of the field zones in the frame, regardless of their specific settings.

By default, the lighting effect is activated in frames containing 3D Cartesian and 3D Contour plots. Calling **gpUseFieldZoneContourLighting** activates the lighting effect in the frame indicated by *framenum* and then activates or deactivates contour lighting in the specified field zones depending on how *activate* is set. If *activate* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of *activate*.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpUseFieldZoneContourLighting** for a subset of the field zones in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want contour lighting to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate contour lighting for all of the field zones that are contained in the plot control structure but not specified in the call to **gpUseFieldZoneContourLighting**.

If you call **gpUseFieldZoneContourLighting** for a subset of the field zones in the frame with an *activate* argument set to a scalar zero, then it will assume

that you want contour lighting to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when the lighting effect is activated in a frame, both contour and shade lighting are displayed in each field zone in that frame.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether contour lighting should be activated or deactivated.

Call **gpUseLightingEffect** to activate or deactivate the lighting effect for the frame.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
fieldzonenum = 2;
activate = 0;
ret = gpUseFieldZoneContourLighting(&gp,framenum,fieldzonenum,activate
```

SOURCE gpfieldzone.src

SEE ALSO **gpUseLightingEffect**, **gpUseFieldZoneShadeLighting**, **gpSetLightingEffect**

gpUseFieldZoneShadeLighting

PURPOSE Activates or deactivates the lighting effect for the shade layer in the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpUseFieldZoneShadeLighting**(*pgp*, *framenum*, *fieldzonenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
activate	scalar or N×1 vector of ones and zeros:
	0 deactivate the lighting effect for the shade layer.
	1 activate the lighting effect for the shade layer.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

The lighting effect is controlled on two levels: in a frame and in each of the field zones in the frame. If the lighting effect is activated in a frame, then both contour and shade lighting are displayed in each field zone in the frame by default. If the lighting effect is not activated in a frame, then neither contour nor shade lighting are displayed in any of the field zones in the frame, regardless of their specific settings.

By default, the lighting effect is activated in frames containing 3D Cartesian and 3D Contour plots. Calling **gpUseFieldZoneShadeLighting** activates the lighting effect in the frame indicated by *framenum* and then activates or deactivates shade lighting in the specified field zones depending on how *activate* is set. If *activate* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of *activate*.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpUseFieldZoneShadeLighting** for a subset of the field zones in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want shade lighting to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate shade lighting for all of the field zones that are contained in the plot control structure but not specified in the call to **gpUseFieldZoneShadeLighting**.

If you call **gpUseFieldZoneShadeLighting** for a subset of the field zones in the frame with an *activate* argument set to a scalar zero, then it will assume that

you want shade lighting to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when the lighting effect is activated in a frame, both contour and shade lighting are displayed in each field zone in that frame.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether shade lighting should be activated or deactivated.

Call **gpUseLightingEffect** to activate or deactivate the lighting effect for the frame.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
fieldzonenum = 2;
activate = 0;
ret = gpUseFieldZoneShadeLighting(&gp,framenum,fieldzonenum,activate);
```

SOURCE gpfieldzone.src

SEE ALSO **gpUseLightingEffect**, **gpUseFieldZoneContourLighting**, **gpSetLightingEffect**

gpUseFieldZoneTranslucency

PURPOSE Activates or deactivates the translucency effect for the specified field zone(s).

LIBRARY gaussplot

FORMAT *ret* = **gpUseFieldZoneTranslucency**(*pgp*, *framenum*, *fieldzonenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
fieldzonenum	scalar or N×1 vector, field zone number(s).
activate	scalar or N×1 vector of ones and zeros:
	0 deactivate the translucency effect.
	1 activate the translucency effect.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS

The translucency effect controls the translucency of shade and contour layers in a field zone. It is controlled on two levels: in a frame and in each of the field zones in the frame. If the translucency effect is activated in a frame, then it is displayed in each field zone in the frame by default. If the translucency effect is not activated in a frame, then it is not displayed in any of the field zones in the frame, regardless of their specific settings.

By default, the translucency effect is activated in frames containing 3D Cartesian and 3D Contour plots. Calling **gpUseFieldZoneTranslucency** activates the translucency effect in the frame indicated by *framenum* and then activates or deactivates the translucency effect in the specified field zones depending on how *activate* is set. If *activate* is an N×1 vector, then each of the field zones in *fieldzonenum* will be set according to the corresponding element of *activate*.

The number of field zones in a frame is determined by the number of zones in the data file (see Section 1.3.1 for more information). The plot control structure contains information for only those field zones for which you have specified options; thus there may be fewer field zones in the plot control structure than in the frame. If you call **gpUseFieldZoneTranslucency** for a subset of the field zones in the frame with an *activate* argument set to a scalar one (or a vector of zeros and ones), then it will assume that you want the translucency effect to be deactivated for all of the unspecified field zones in the frame. Therefore, it will deactivate the translucency effect for all of the field zones that are contained in the plot control structure but not specified in the call to **gpUseFieldZoneTranslucency**.

If you call **gpUseFieldZoneTranslucency** for a subset of the field zones in the frame with an *activate* argument set to a scalar zero, then it will assume that you want the translucency effect to be activated for all of the other field zones in the frame. In this case, it will not set any options for the unspecified field zones because, by default, when the translucency effect is activated in a frame, it is displayed in each field zone in that frame.

Because it is possible that not all of the field zones that will appear in the frame are contained in the plot control structure, the safest way to use this command is to specify for each field zone in the frame whether the translucency effect should be activated or deactivated.

Call **gpUseTranslucencyEffect** to activate or deactivate the translucency effect for the frame.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
fieldzonenum = { 1,2 };
activate = { 0,1 };
ret = gpUseFieldZoneTranslucency(&gp,framenum,fieldzonenum,activate);
```

This example activates the translucency effect in the second field zone in the first frame.

SOURCE gpfieldzone.src

SEE ALSO **gpUseTranslucencyEffect**, **gpSetTranslucency**

gpUseGridAreaLighting

PURPOSE Activates or deactivates light source shading in the grid area of a frame.

LIBRARY gaussplot

FORMAT *ret* = **gpUseGridAreaLighting**(*pgp*, *framenum*, *uselighting*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
uselighting	scalar, one of the following:
	0 use light source shading.
	1 do not use light source shading.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS By default, light source shading is used in the grid areas of frames containing 3D Cartesian and 3D Contour plots.

Call **gpFillGridArea** if you do not want the grid area to be filled, or call **gpSetGridAreaFillColor** to change the fill color.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE
```
framenum = 1;
uselighting = 0;
ret = gpUseGridAreaLighting(&gp,framenum,uselighting);
```

SOURCE gp.src

SEE ALSO **gpFillGridArea**, **gpSetGridAreaFillColor**

gpUseLightingEffect

PURPOSE Activates or deactivates the lighting effect in the specified frame(s).

LIBRARY gaussplot

FORMAT *ret* = **gpUseLightingEffect**(*pgp*, *framenum*, *activate*);

INPUT

pgp pointer to a plot control structure.

framenum scalar or N×1 vector, frame number(s).

activate scalar, one of the following:

0 deactivate the lighting effect.

1 activate the lighting effect.

OUTPUT

ret scalar, return code, 0 if successful, otherwise an error code.

REMARKS

The lighting effect is controlled on two levels: in a frame and in each of the field zones in the frame. If the lighting effect is activated in a frame, then both contour and shade lighting are displayed in each field zone in the frame by default. If the lighting effect is not activated in a frame, then neither contour nor shade lighting are displayed in any of the field zones in the frame, regardless of their specific settings.

By default the lighting effect is activated in frames containing 3D Cartesian and 3D Contour plots. Calling **gpUseLightingEffect** activates or deactivates the lighting effect in the specified frame. To activate or deactivate shade or contour lighting for a subset of the field zones in a frame, call **gpUseFieldZoneContourLighting** or **gpUseFieldZoneShadeLighting**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
activate = 0;
ret = gpUseLightingEffect(&gp,framenum,activate);
```

SOURCE

gpfieldzone.src

SEE ALSO

gpUseFieldZoneContourLighting, **gpUseFieldZoneShadeLighting**, **gpSetLightingEffect**

gpUseTranslucencyEffect

PURPOSE Activates or deactivates the translucency effect in the specified frame(s).

LIBRARY `gaussplot`

FORMAT *ret* = **gpUseTranslucencyEffect**(*pgp*, *framenum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar or N×1 vector, frame number(s).
activate	scalar, one of the following:
	0 deactivate the translucency effect.
	1 activate the translucency effect.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS The translucency effect controls the translucency of shade and contour layers in a field zone. It is controlled on two levels: in a frame and in each of the field zones in the frame. If the translucency effect is activated in a frame, then it is displayed in each field zone in the frame by default. If the translucency effect is not activated in a frame, then it is not displayed in any of the field zones in the frame, regardless of their specific settings.

By default the translucency effect is activated in frames containing 3D Cartesian and 3D Contour plots. Calling **gpUseTranslucencyEffect** activates or deactivates the translucency effect in the specified frame. To activate or deactivate the translucency effect for a subset of the field zones in a frame, call **gpUseFieldZoneTranslucency**.

This command applies ONLY to 3D Cartesian and 3D Contour plots.

EXAMPLE

```
framenum = 1;
activate = 1;
ret = gpUseTranslucencyEffect(&gp,framenum,activate);
```

SOURCE gpfieldzone.src

SEE ALSO **gpUseFieldZoneTranslucency**, **gpSetTranslucency**

gpUseXAxisLogScaling

PURPOSE Activates or deactivates log scaling on an X-axis.

LIBRARY gaussplot

FORMAT *ret* = **gpUseXAxisLogScaling**(*pgp*, *framenum*, *axisnum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, X-axis number.
activate	scalar, one of the following:
	0 deactivate log scaling on the X-axis.
	1 activate log scaling on the X-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which X-axis the option is set for, and it is used only by frames that contain Line plots. By default, one X-axis and one Y-axis are used. To display additional X axes, call **gpSetLineMapAxes** to link one or more line maps to an X-axis numbered greater than one. X axes that are added in this way are displayed by default.

By default, log scaling is deactivated.

EXAMPLE
```
framenum = 1;
axisnum = 1;
```

```
activate = 1;
ret = gpUseXAxisLogScaling(&gp,framenum,axisnum,activate);
```

SOURCE gp.src

gpUseYAxisLogScaling

PURPOSE Activates or deactivates log scaling on a Y-axis.

LIBRARY gaussplot

FORMAT *ret* = **gpUseYAxisLogScaling**(*pgp*, *framenum*, *axisnum*, *activate*);

INPUT

pgp	pointer to a plot control structure.
framenum	scalar, frame number.
axisnum	scalar, $1 \leq axisnum \leq 5$, Y-axis number.
activate	scalar, one of the following:
	0 deactivate log scaling on the Y-axis.
	1 activate log scaling on the Y-axis.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS Line plots allow up to five X and Y axes in a frame. The *axisnum* argument specifies which Y-axis the option is set for, and it is used only by frames that contain line plots. By default, one X-axis and one Y-axis are used. To display additional Y axes, call **gpSetLineMapAxes** to link one or more line maps to a Y-axis numbered greater than one. Y axes that are added in this way are displayed by default.

By default, log scaling is deactivated.

EXAMPLE

```
framenum = 1;
```

```
axisnum = 1;
activate = 1;
ret = gpUseYAxisLogScaling(&gp,framenum,axisnum,activate);
```

SOURCE gp.src

gpWait

PURPOSE Specifies whether the **GAUSSplot** plotting commands will return to **GAUSS** immediately after initializing the **GAUSSplot** GUI or wait until the **GAUSSplot** window has been closed.

LIBRARY gaussplot

FORMAT **gpWait(***pgp*, *wait***);**

INPUT

pgp	pointer to a plot control structure.
wait	scalar, one of the following:
	0 do not wait; return to **GAUSS** immediately.
	1 wait until the **GAUSSplot** window has been closed to return to **GAUSS**.

REMARKS By default, the **GAUSSplot** plotting commands (see Section 2.1.16) return to **GAUSS** immediately after initializing the **GAUSSplot** GUI.

EXAMPLE

```
wait = 1;
gpWait(&gp,wait);
```

SOURCE gp.src

SEE ALSO **gpPlot**

gpWritePlotData

PURPOSE Writes the data in a **gpData** structure to a `.plt` data file.

LIBRARY `gaussplot`

FORMAT *ret* = **gpWritePlotData**(*pgdat*, *datafile*);

INPUT

pgdat	pointer to a **GAUSSplot** data structure containing the data to be written.
datafile	the name of the data file to write.

OUTPUT

ret	scalar, return code, 0 if successful, otherwise an error code.

REMARKS If *datafile* does not have a `.plt` extension, then the extension will be appended to it.

In addition to writing the data file, **gpWritePlotData** saves the structure that *pgdat* is pointing at to a file with the same name as *datafile*, except with an `.fsr` extension instead of a `.plt` extension. This file is accessed by **GAUSSplot** commands that allow you to specify variables using variable names. These commands open the `.fsr` file to convert the input variable names to indices of the variables into the data file. If you want to specify variables using variable names in **GAUSSplot** commands, then you need to keep the `.fsr` structure file together with your `.plt` data file.

Call **gpCleanUp** after running your program to delete the structure and data files.

EXAMPLE

```
library gaussplot;
#include gp.sdf

p = pi;
x = seqa(0,p/4,15);
```

```
y = sin(x);

struct gpData gdat;
string vnames = { "X", "Y" };
gdat = gpSetPlotData(x~y,vnames);
ret = gpWritePlotData(&gdat,"mydata.plt");
```

SOURCE gpwritedata.src

SEE ALSO **gpInitPlotData**, **gpAddZone**, **gpSetPlotData**

gpXYLinePlotCreate

PURPOSE Creates a plot control structure and initializes its members to default values for an XY Line plot.

LIBRARY gaussplot

FORMAT *gp* = **gpXYLinePlotCreate;**

OUTPUT *gp* plot control structure with its members set to default values for an XY Line plot.

REMARKS You may use *gp* to plot a graph using either **gpMakeXYLinePlot** or **gpPlot**. Calling **gpXYLinePlotCreate** sets the plot type of the entire graph to XY Line. To create a graph with multiple frames that contain plots of different types, use **gpPlotCreate**.

A plot control structure initialized with **gpXYLinePlotCreate** contains one frame by default.

EXAMPLE

```
library gaussplot;
#include gp.sdf
```

```
x = seqa(0.2,0.2,20);
y = log(x);
string vnames = { "V1", "V2" };

struct gpPlotControl gp;
gp = gpXYLinePlotCreate;
ret = gpMakeXYLinePlot(&gp,x,y,vnames);
```

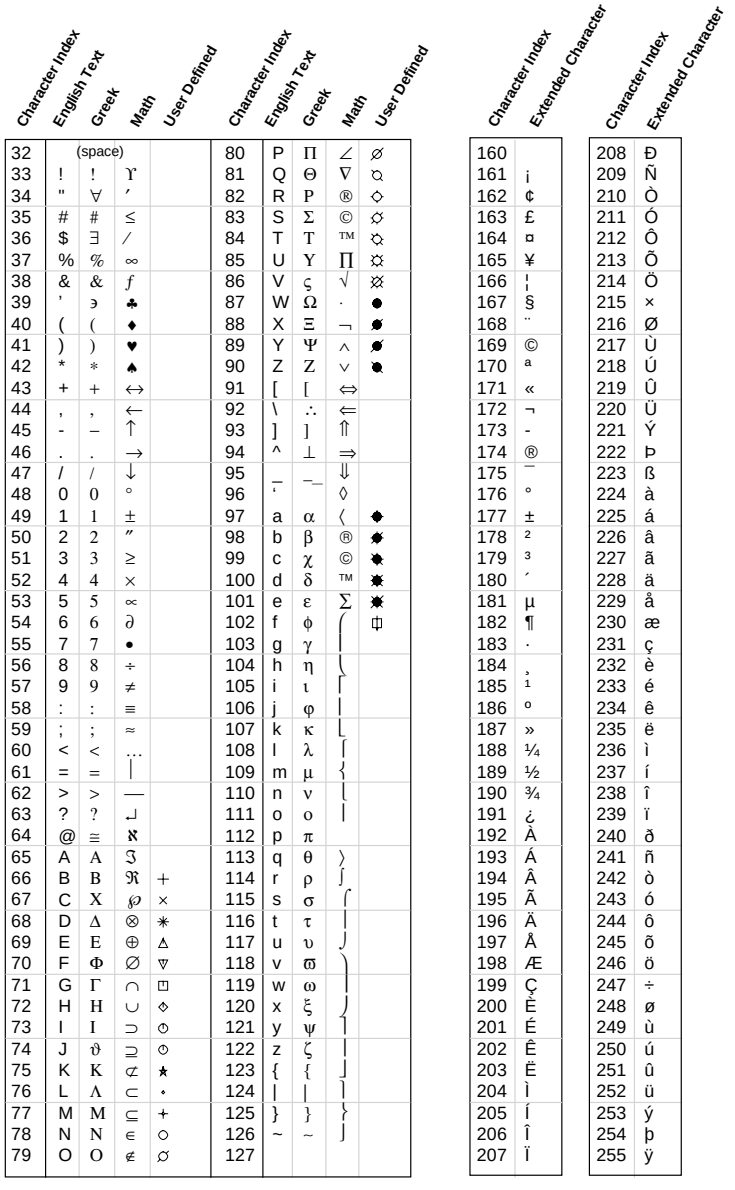
This example illustrates how you may create an XY Line plot with one line map, using defaults for all style and display options. The first variable (**V1**) will be associated with the X-axis, and the second variable (**V2**) will be associated with the Y-axis.

SOURCE gpxylineplot.src

SEE ALSO **gpMakeXYLinePlot**, **gpPlot**

Character Table A

Following is a table that lists all of the available characters in **GAUSSplot**:



Character Index	English Text	Greek	Math	User Defined
32	(space)			
33	!	!	ϒ	
34	"	∀	′	
35	#	#	≤	
36	$	∃	⁄	
37	%	%	∞	
38	&	&	ƒ	
39	'	϶	♣	
40	(	(	♦	
41	)	)	♥	
42	*	*	♠	
43	+	+	↔	
44	,	,	←	
45	-	–	↑	
46	.	.	→	
47	/	/	↓	
48	0	0	°	
49	1	1	±	
50	2	2	″	
51	3	3	≥	
52	4	4	×	
53	5	5	∝	
54	6	6	∂	
55	7	7	•	
56	8	8	÷	
57	9	9	≠	
58	:	:	≡	
59	;	;	≈	
60	<	<	…	
61	=	=	|	
62	>	>	—	
63	?	?	↵	
64	@	≅	ℵ	
65	A	A	ℑ	
66	B	B	ℜ	+
67	C	X	℘	×
68	D	Δ	⊗	✳
69	E	E	⊕	△
70	F	Φ	∅	▽
71	G	Γ	∩	◫
72	H	H	∪	◈
73	I	I	⊃	⊙
74	J	ϑ	⊇	⊙
75	K	K	⊄	★
76	L	Λ	⊂	⋆
77	M	M	⊆	+
78	N	N	∈	○
79	O	O	∉	∅

Character Index	English Text	Greek	Math	User Defined
80	P	Π	∠	⌀
81	Q	Θ	∇	◌
82	R	P	®	◇
83	S	Σ	©	☼
84	T	T	™	☼
85	U	Y	∏	☼
86	V	ς	√	☒
87	W	Ω	·	●
88	X	Ξ	¬	◙
89	Y	Ψ	∧	◙
90	Z	Z	∨	◙
91	[	[	⇔	
92	\	∴	⇐	
93	]	]	⇑	
94	^	⊥	⇒	
95	_	‾	⇓	
96	`	‾	◊	
97	a	α	〈	◆
98	b	β	®	◙
99	c	χ	©	◙
100	d	δ	™	◙
101	e	ε	∑	◙
102	f	ϕ	⎛	中
103	g	γ	⎜	
104	h	η	⎝	
105	i	ι	⎡	
106	j	φ	⎢	
107	k	κ	⎣	
108	l	λ	⎧	
109	m	μ	⎨	
110	n	ν	⎩	
111	o	o	⎪	
112	p	π		
113	q	θ	〉	
114	r	ρ	∫	
115	s	σ	⌠	
116	t	τ	⎮	
117	u	υ	⌡	
118	v	ϖ	⎞	
119	w	ω	⎟	
120	x	ξ	⎠	
121	y	ψ	⎟	
122	z	ζ	⎤	
123	{	{	⌋	
124	|	|	⎥	
125	}	}	⎦	
126	~	~	⌋	
127				

Character Index	Extended Character
160	
161	¡
162	¢
163	£
164	¤
165	¥
166	¦
167	§
168	¨
169	©
170	ª
171	«
172	¬
173	-
174	®
175	¯
176	°
177	±
178	²
179	³
180	´
181	µ
182	¶
183	·
184	¸
185	¹
186	º
187	»
188	¼
189	½
190	¾
191	¿
192	À
193	Á
194	Â
195	Ã
196	Ä
197	Å
198	Æ
199	Ç
200	È
201	É
202	Ê
203	Ë
204	Ì
205	Í
206	Î
207	Ï

Character Index	Extended Character
208	Ð
209	Ñ
210	Ò
211	Ó
212	Ô
213	Õ
214	Ö
215	×
216	Ø
217	Ù
218	Ú
219	Û
220	Ü
221	Ý
222	Þ
223	ß
224	à
225	á
226	â
227	ã
228	ä
229	å
230	æ
231	ç
232	è
233	é
234	ê
235	ë
236	ì
237	í
238	î
239	ï
240	ð
241	ñ
242	ò
243	ó
244	ô
245	õ
246	ö
247	÷
248	ø
249	ù
250	ú
251	û
252	ü
253	ý
254	þ
255	ÿ

Character indices in Tecplot.

Obsoleted Commands B

For the sake of clarity, we have renamed the following commands:

Obsoleted Command	Replacement Command
`gpShowFieldZoneBoundary`	`gpShowFieldZoneBoundaryLayer`
`gpShowFieldZoneContour`	`gpShowFieldZoneContourLayer`
`gpShowFieldZoneMesh`	`gpShowFieldZoneMeshLayer`
`gpShowFieldZoneScatter`	`gpShowFieldZoneScatterLayer`
`gpShowFieldZoneShade`	`gpShowFieldZoneShadeLayer`
`gpShowFieldZoneVectors`	`gpShowFieldZoneVectorLayer`
`gpShowLineMapBars`	`gpShowLineMapBarLayer`
`gpShowLineMapErrorBars`	`gpShowLineMapErrorBarLayer`
`gpShowLineMapLines`	`gpShowLineMapLineLayer`
`gpShowLineMapSymbols`	`gpShowLineMapSymbolLayer`

The obsoleted commands will still work for backwards compatibility, but you should use the replacement commands in new programs.

We chose to rename the commands, so it would be clear that these commands control the **layers** in field zones and line maps (see Section 1.5.2 for a description of layers).

The behavior has changed slightly from the obsoleted commands to their replacements. See the command reference pages for more information.

Index

H

L

M

O

P

R

S

T

V

W

X

Z