Graphics Language Interpreter Reference Manual

This document describes the concepts, commands and features of the Graphics Language Interpreter (GLI).

Written by Josef Heinen. Edited by Daniel Bleil and Dennis Peterson.

Revision/Update Information

November 1995. Supercedes the GLI 4.4 Manual.

Software Version

GLI 4.5

Required Operating Systems

AIX, Digital UNIX, HP-UX, IRIX, OpenVMS, RISC/Ultrix, Solaris, SunOS, UNICOS

Required Software

The GLI system requires supported operating system software. Certain GLI modules also require X Windows software and associated hardware.

Table of Contents

CHAPTER 1

Introduction

Table of Contents

GLI

The Graphics Language Interpreter (or GLI) is a complete graphics system that offers a versatile environment for plotting complex data sets and displaying images. GLI combines powerful interfaces, flexible graphics utilities, and an extensive set of programming tools to provide engineers, scientists and analysts a complete solution for data analysis and scientific visualization.

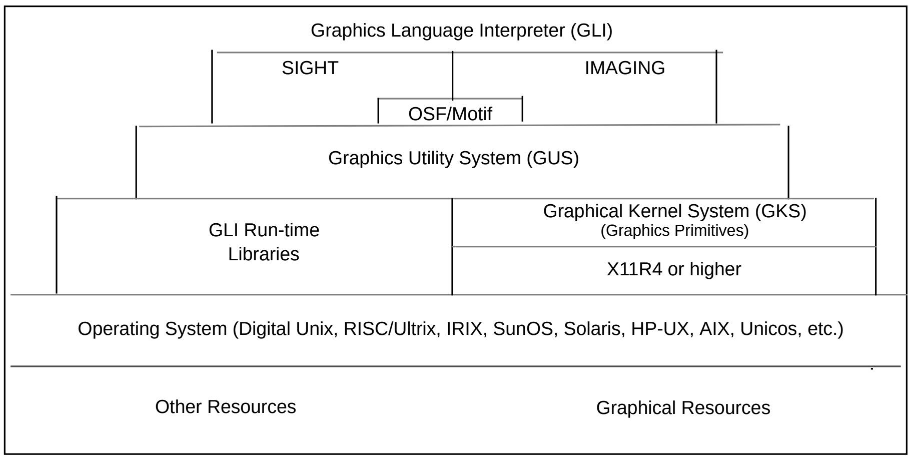
GLI commands can be entered through the keyboard, selected using a mouse or input into a menu, and the results are immediately displayed. These commands can be combined in a program to give you the ability to easily develop sophisticated graphic applications.

This *Graphics Language Interpreter Reference Manual* provides a comprehensive description of GLI's usage, structure and components. This introduction contains concepts you should know about GLI.

The GLI System

At the top of the GLI structure (Figure 1-1) are flexible interfaces, allowing you to access all components of the GLI system. Below the interfaces are the building blocks that help you to input, filter, plot, and visualize your data. You can easily customize these components for your specific requirements. The entire GLI system is highly portable, available for most computer systems, and able to support a wide variety of display devices, including PostScript, X Windows and Motif.

FIGURE 1-1 GLI's Structure


Graphics Language Interpreter (GLI)
SIGHT
IMAGING
OSF/Motif
Graphics Utility System (GUS)
GLI Run-time Libraries
Graphical Kernel System (GKS)
(Graphics Primitives)
X11R4 or higher
Operating System (Digital Unix, RISC/Ultrix, IRIX, SunOS, Solaris, HP-UX, AIX, Unicos, etc.)
Other Resources
Graphical Resources

GLI Interfaces

You interact with the GLI system through an interface that best fits you and your applications.

GLI Command Line

The GLI command line accepts simple statements from the GLI, GUS, and GKS command languages to communicate to all aspects of GLI, including your operating system. You issue English-like commands through the keyboard and GLI automatically reads, manipulates, and plots your data using sophisticated 2-D and 3-D graphic utilities. Commands can be saved in macros and implemented as command procedures to solve complex problems graphically.

AUTOPLOT

AUTOPLOT is a form-driven interface. You fill in command fields and enter a data file that AUTOPLOT reads, analyzes, and plots as a linear or logarithmic chart, according to your instructions. Titles, labels and legends can be automatically added to quickly produce presentation quality 2-D graphs. Menu interfaces such as AUTOPLOT can be customized for your specific requirements.

SIMPLEPLOT

SIMPLEPLOT provides a fast and easy method to read, filter, analyze and graph 2-D data files for applications such as signal processing. Among others, SIMPLEPLOT's functions include:

- Zoom, Select Range, Truncate
- Data modification
- Line Smoothing
- FFT, Inverse FFT
- Cubic Spline, High Pass
- Amplitude, Bandpass

X Windows User Interface (XUI)

XUI is a point-and-click version of the GLI command line interface. Using the mouse, you select commands from pop-up windows to create sophisticated 2-D and 3-D graphic applications. All GLI commands are available to you when using XUI. This interface is compatible with all properly configured computer systems that support the X Windows X11R4 software. Using XUI, you can also:

- Access other interfaces
- Construct your own customized point-and-click interfaces

Simple Interactive Graphics Handling Tool (SIGHT)

SIGHT is a robust point-and-click graphics program, combining object-oriented commands with the Motif environment. Using the mouse, you select commands from pull-down and pop-up menus that allow you to quickly create, edit and print publication quality 2-D graphs and free-hand drawings. SIGHT is compatible with systems that include X Windows X11R4 software and the Motif libraries. Using SIGHT, you can also:

- Read data files or input data through your keyboard or mouse

- Rapidly create graphic prototypes
- Edit graphs interactively

Imaging

GLI's Imaging module is a complete system for processing and displaying color and grayscale images. These images can be filtered, enhanced and animated to provide detailed insight into your data.

The Basic Usage chapter discusses each interface in greater detail.

Graphics Utilities

Graphics Utility System (GUS)

GUS includes high level utilities that create complete 2-D or 3-D graphics from simple commands. GUS automates the Graphical Kernel System (GKS, described below) to produce graphic applications quickly. Among others, GUS contains high level graphics and data handling subroutines such as:

- 2-D and 3-D Line Graphs
- Scatter Diagrams
- Error Bars
- Bar charts
- Pie Charts
- Contours
- Histograms
- 3-D Surfaces
- Image Display
- Data filters
- Linear and Logarithmic Transformations
- Automated Text Generation

Graphical Kernel System (GKS)

GLI is based upon an *open systems* concept and includes a complete set of drawing tools and output primitives from the internationally accepted graphics standard, GKS. GKS instructions control the basic elements of the graph, so highly customized applications can be prepared for unique requirements. Although GLI includes its own implementation of GKS, its components are fully compatible with GKS software available from other vendors. So, GLI can enhance existing GKS programs and minimize application development time.

GLI Facilities

Mathematical Operands and Intrinsic Functions

GLI provides mathematical operands (+, -, /, *, **) and intrinsic functions (SIN, TAN, ABS, etc.). Using these building blocks, you can define your own mathematical functions for customized data analysis.

Input/Output Functions

GLI provides tools and utilities that automatically read and write data files.

Interpretive Programming Language

GLI's command language parallels a standard programming language, offering:

- Assignment statements and symbol substitution.
- Standard operators to combine variables and constants into expressions.
- Boolean expressions.
- GOSUB subroutine calls.
- GOTO statements.

Sequences of often-used GLI commands can be recorded in a file and executed from any of GLI's interfaces as a macro. The code used to create the GLI demonstration graphs is included in the GLI directory $GLI_DEMO. You may want to review this directory and use the code as a guide for creating your own GLI applications.

Help

The GLI Help facility provides on-line documentation for GLI commands. The on-line documentation is arranged in multiple levels, allowing you to optionally specify topics and sub-topics.

CHAPTER 2

Basic Usage

Table of Contents

Table of Contents continued

About the Basic Usage Chapter

This Basic Usage section provides a general overview of the fundamental capabilities and features of GLI, and this chapter is not intended to offer a comprehensive look at advanced applications suitable for GLI. The advanced user will want to review the design samples contained in Appendix C to better understand the potential of GLI.

Starting GLI

The GLI installation procedure is discussed in Appendix A to this manual. If GLI has been installed properly, you can begin GLI by entering the following command:

```
system prompt# gli
```

GLI indicates that it is ready to accept commands by displaying the GLI prompt (gli>):

```
    G L I
    RISC version 4.5 (UNIX)
    patchlevel 4.5.4, 20 Nov 95

    Copyright @ 1986-1995, Josef Heinen, Jochen Werner
    Copyright @ 1994, ZAM, Forschungszentrum Juelich GmbH (GR-Software)

    Send bugs and comments to J.Heinen@KFA-Juelich.de

gli>
```

Important: SunOS users should always interact with GLI through a shelltool, Xterm or other ANSI-compatible terminal device. However, some inconsistencies may appear when using the shelltool. Therefore, it is suggested SunOS users invoke OpenWindows, if available, then use the Xterm command or another ANSI terminal device.

GLI Demonstration Programs

You may wish to verify the installation on your system or review GLI's capabilities by running the demonstration programs. GLI includes demonstration programs for terminal and X Window systems.

Note: If the GLI demonstrations fail to execute properly, make sure GLI is in the proper operating state (type `initialize` or `init` at the GLI command prompt) or refer to Appendix A for additional information.

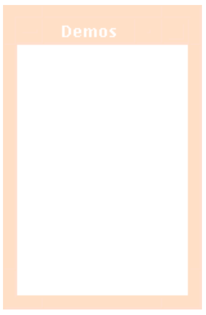
X Window Demo

This demonstration program supports devices properly configured to run X Window applications. To activate this demo, type:

```
gli> xdemo
```

A pop-up menu appears as in the figure below.

FIGURE 2-1 Demonstration Menu for X Window Devices



Use the mouse to select the demonstrations:

- Choose *the* individual demonstration files.
- Choose *Movie* to see the demonstration files in a continuous series.
- Choose *Finish* to exit the GLI Demonstration Interface.

Image Demonstration

GLI includes an imaging module that displays, filters and manipulates raster images in a variety of formats. The imaging module can also be used to animate a sequence of image files. To view a demonstration on systems properly configured to support X Windows, enter the following command at the GLI command prompt or refer to the Image chapter in this manual.

```
gli> image_demo
```

ASCII Terminal Demonstration

This demonstration program may be executed from an ASCII terminal device or an X Windows device by entering:

```
gli> demo
```

Note: When using this demonstration from an ANSI terminal, Sun and Hewlett-Packard users may experience incompatibility with the default terminal type for GLI. If this demo does not appear to function properly, contact your system manager or refer to Appendix A for information to properly configure your terminal.

A keyboard-driven demonstration interface appears as in the figure below.

FIGURE 2-2 Demonstration Menu for Terminal Devices

Use the Up and Down arrow keys to move vertically through the menu and use the RETURN key to move horizontally.

- Choose *the* individual demonstration files.
- Choose *Movie* to see a demonstration of all the files. Enter RETURN after each graph to view the next one.

- Press the Do key (PF1, or Esc + 1 on Sun keyboards) to execute your choice.
- Press the Exit key (PF4, or Esc + 4 on Sun keyboards) to exit the demonstration interface.

Quitting GLI

Enter the QUIT command whenever you are finished using the GLI system. QUIT saves your GLI session to the GLI.JOU file.

```
gli> quit
```

Enter the EXIT command to terminate GLI without creating the journal file and saving your session instructions.

```
gli> exit
```

GLI Components

Interfaces

GLI is designed for interactive use from a graphics workstation or a graphics terminal. You can access GLI from one of its interfaces:

- GLI Command Line
  Simple, English-like statements for plotting two- and three-dimensional data and creating presentation-quality graphs and sophisticated graphic applications.
- AUTOPLOT
  A “fill in the blank” form that creates XY graphs from two-dimensional data.
- SIMPLEPLOT
  An interactive interface for filtering and analyzing two-dimensional data.

- X Windows User Interface (XUI)
  An X Windows point & click version of the GLI command line interface for plotting two- and three-dimensional data.
- Simple Interactive Graphics Handling Tool (SIGHT)
  A Motif point & click graphics editor for plotting two-dimensional data.

Each of these interfaces is discussed later in this section.

GKS

The Graphical Kernel System (GKS) provides the graphics foundation for GLI, including the ability to draw the basic elements of a graph such as lines, polygons and text. GKS instructions control all aspects of your graph, and you can create highly customized and sophisticated applications with these commands.

This section contains basic examples of GKS commands.

Utilities

The Graphics Utility System (GUS) creates complete 2-D or 3-D graphics. A single GUS command issues multiple instructions to GLI which simplifies the development of graphic applications. Often a GUS command will produce a complete graphic which can then be easily customized and enhanced.

The GUS instructions discussed in this section will give you a general understanding of their capability.

Functions and Facilities

In addition to its interfaces, primitives and utilities, GLI also contains:

- Mathematical operands and intrinsic functions.
- Help facility (on-line documentation).
- An interpretive programming language for generating graphics.
- Ability to store multiple instructions in executable macros.

An overview of these functions and facilities is provided in the sections that follow.

Using the GLI Command Line

The GLI command line accepts simple, English-like statements that allow you to communicate to all aspects of the GLI system, including your operating system. By issuing these commands through the keyboard, GLI can read, filter and plot your data as a two- or three-dimensional graphic.

Commands take the form of the command name followed by parameters and qualifiers (always press the RETURN key to execute a command):

command_name *parameter [qualifiers]…*

GLI instructions can also be placed in macros and used as command procedures, allowing you to execute a series of commands quickly.

As you will see as you proceed through this chapter, `gks` invokes a Graphical Kernel System command and `gus` activates a Graphics Utility Command.

You will likely find the Help facility useful as you review GLI. For more information on Help, refer to this chapter, or enter this command at the GLI prompt (gli>):

```
gli> help
```

Plot 2-D XY Graph

Use this section as a guide to plot two-dimensional data from the GLI command line.

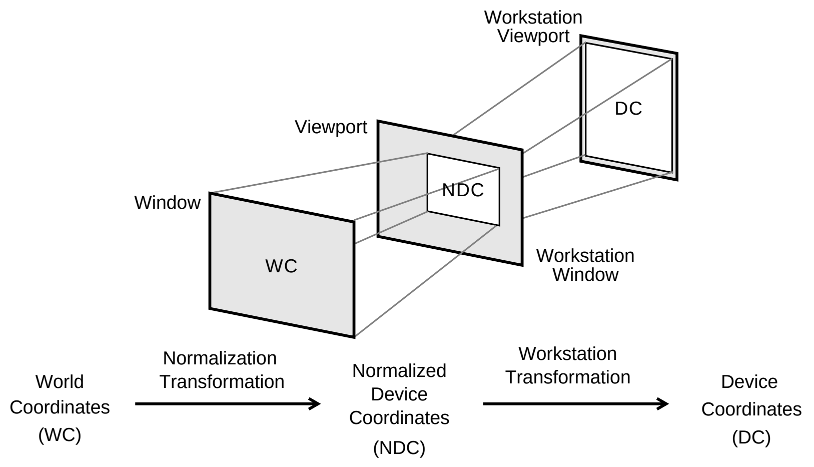
Define the Graph Area

GLI's Graphical Kernel System (GKS) provides a truly device-independent method to create graphs and graphic applications. GLI gives you the flexibility to develop your graphs without concern as to the device or devices they will ultimately be displayed on.

A general understanding of the GKS coordinate systems and transformations will help you position and size your graph properly. For a detailed discussion of this subject, refer to the GKS chapter.

The figure below illustrates the two-stage transformation process.

FIGURE 2-3 Transformation Process



A few commands will determine the size and scale of the graph and position it on any supported device. Additional commands can place multiple graphs simultaneously on the same device (or page).

As you follow this Basic Usage chapter, similar commands will be used to define the graph area each time you create a graph. Therefore, you may want to store these commands using the LEARN command. (For more on LEARN see the GLI Command Language chapter.) To set the display area and then save the instructions as a command procedure, enter the following:

```
gli> learn
gli> gks set xform wc
gli> gks set viewport 0.05 0.95 0.05 0.95
gli> gks set window 1 100 1 100
gli> save display1
```

You have now set a display area on your device and saved the instructions for future use. To invoke these commands again, simply enter:

```
gli> @display1
```

Create Data Items

Often you will use GLI to graph existing data files. However, for the purposes of this section, let's create sample data items. Define two one-dimensional arrays of real numbers by assigning values to the variables A and B. Values for A will be in the range of -1.0 to 1.0 in 0.1 increments, and B values will be defined by the sine function of the assigned A values.

```
gli> a = -1(.1)1
gli> b = sin(a)
```

Write Data to a File

Write the values of A and B to a file (TEMP.DAT) using the WRITE command. Note that you can enter a command without specifying the parameters and qualifiers, and GLI will prompt you for any additional information it requires.

```
gli> write
_File: temp.dat
_Data: a b
 21 lines written to file temp.dat
```

Or, you can enter a command and its parameters and qualifiers as a single instruction, making the above:

```
gli> write temp.dat a b
 21 lines written to file temp.dat
```

Any command may be canceled without execution by simultaneously pressing CTRL/C. If you enter erroneous or incomplete commands, GLI returns an informative message.

Read Data from a File

Read the data file into GLI using the READ command. GLI asks for the file name and how you want the data read. Read the data as X and Y values.

```
gli> read
_file: temp.dat
_variable(s): x y
 21 lines read from file temp.dat
```

Again, you could have issued the last command as:

```
gli> read temp.dat x y
 21 lines read from file temp.dat
```

GLI reads free-formatted ASCII files, including files containing text. Refer to page 2-53 for more information on data types compatible with GLI.

Plot Line Graph

The GUS PLOT command combines several instructions to create a complete line graph from a single command. PLOT automatically draws axes and sets the viewport and window appropriately for the data. As you will see later in this section, GLI also allows you to have extensive control over line graphs.

Graph the X and Y values using the GUS PLOT command. GLI prompts you for the X and Y values to create a two-dimensional graph.

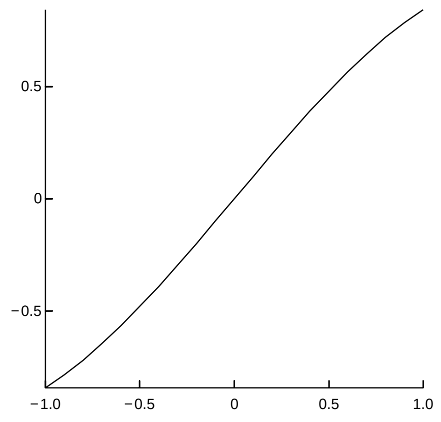
```
gli> gus plot
_x: x
_y: y
```

Or, alternatively, type:

```
gli> gus plot x y
```

The graphic is displayed on your output device.

FIGURE 2-4 Plotting an XY Line Graph



As you have seen, the GUS PLOT utility creates line graphs quickly with one command. It automatically sets the drawing space and the viewport according to the data and draws the axis and line graph.

Clear the Display

Without affecting other GLI settings, clear the display:

```
gli> gks clear_ws
```

Create a Customized Line Graph

Instead of using the automatic line graph utility (GUS PLOT), create a graph of the same data set using GKS and GUS commands.

Define the Display

Set the display using the DISPLAY1 command procedure or using instructions similar to those on page 2-8, but re-define the viewport to avoid clipping the axes.

```
gli> @display1
gli> gks set viewport 0.15 0.95 0.15 0.95
```

Set Scales

Define and draw two-dimensional scales for the graph. GUS AUTOSCALE_2D automatically scales the axes according to the data and the current display space.

```
gli> gus autoscale_2d x y
gli> gus axes_2d
```

To customize two-dimensional axes, refer to page 5-8 .

Graph the Data

Change the default line color (black), and plot the X and Y data values:

```
gli> gks set pline color blue
gli> gks polyline x y
```

Note: If this graph is not drawn, check to see if the data (TEMP.DAT) from the previous example has been deleted or modified.

Plot Another Data Set

With the preceding graph still on your display, plot a second data set.

Create Data Items

In the previous example, GLI created data items using functions or it read data items from files. In this exercise, you will see that you can input data values from the keyboard.

Create two additional example data arrays and assign them to variables XX and YY.

```
gli> xx := -.9 -.88 -.85 -.8 .1 .2 .5 .7 .9
gli> yy = sqr(xx)
```

Note: Refer to *Using the Assignment Statements* later in this chapter for a discussion on the ‘=’ and ‘:=’ operators.

Graph the Second Curve

Graph the XX and YY values using a linetype (line style) that distinguishes this data set from the previous one you plotted.

```
gli> gks set pline linetype dashed
gli> gks polyline xx yy
```

Add Grid and Text

Add a grid to the plot after changing the line color from blue to black.

```
gli> gks set pline color black
gli> gus grid
```

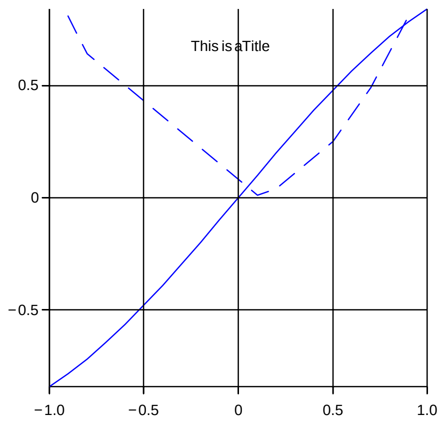
To customize grids, refer to page 5-34 . To control text and text attributes, refer to page 5-66 through page 5-71 and page 6-52 through page 6-61.

Add a title to the plot with the GUS TEXT command. Specify the X Y starting position of the text in World Coordinates.

```
GLI> gus text -.25 .65
_Text: This is a Title
```

For information on the GUS TEXT command, refer to page 5-66 .

FIGURE 2-5 Customized Line Graph



Using a combination of simple GKS and GUS commands, you can construct elaborate graphs with instructions that individually control the curve(s), text, axis, linetype, drawing space, viewport, size, grid, color and other graph attributes.

Delete Data Items and Clear Display

Before proceeding, delete the data items used in the previous examples and clear the display area. The INITIALIZE command re-sets the GLI environment to its default state which globally deletes all assigned variables or functions.

```
gli> initialize
```

Or, you can abbreviate this to:

```
gli> init
```

Later in this chapter you will see how to delete specific variables and functions and clear the display without affecting other GLI environment settings.

Plot 2-D Histogram

This section demonstrates how to use command line instructions to plot a histogram.

Define Display

Set the display area where your graphic will be drawn. In this example, we will define the viewport to be slightly larger than the one used in the previous examples.

```
gli> gks set xform wc
gli> gks set viewport 0.15 0.95 0.15 0.95
gli> gks set window 1 100 1 100
```

Create Data Items

To create the sample data for this example, use the GLI sine function. Define a two-dimensional array of data by assigning values to the variables X and Y. Y values will be in the range of 1 to 100 in increments of 1. X values will be defined by the sine function of the assigned Y values.

```
gli> y = 1(1)100
gli> x = sin(y)
```

Set Scales

GUS AUTOSCALE_2D automatically scales the axes according to the data and current display space. GUS AXES_2D draws axes for the graph.

```
gli> gus autoscale_2d x y
gli> gus axes_2d
```

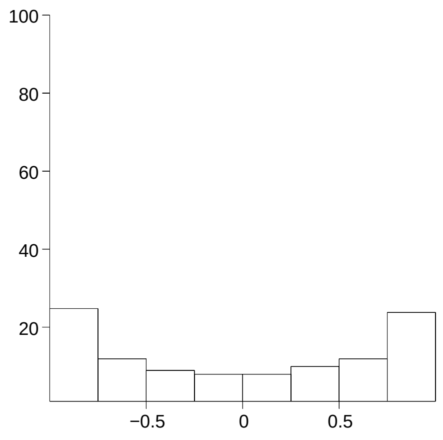
To customize two-dimensional axes, refer to page 5-8.

Plot Histogram

Plot the X values using the GUS HISTOGRAM command.

```
gli> gus histogram x
```

FIGURE 2-6 Plotting a Histogram



Delete Data Items and Clear Display

Before the next example, use the abbreviated INITIALIZE command to re-set the GLI environment to its default state which clears the display and deletes any assigned variables or functions.

```
gli> init
```

Create a Bar Graph

This next section shows how to plot a bar graph using command line instructions.

Create Data Items

To create sample data for this example. define a two-dimensional array of data by assigning values to the variables X and Y. X values will be in the range 1982 to 1992 in increments of 1 year. Y will be defined by a list of values.

```
gli> x := 1982(1)1992
gli> y := 5.1, 9.3, 3, 5.9, 6.3, 10.1, 12.9, 16.8, 20.9, 27.5, 22.4
```

Set Scales

Set two-dimensional scales to fit the data and draw axes for the graph using a text size smaller than the default.

```
gli> gks set window 1981.5, 1992.5, 0, 40
gli> gks set text height 0.015
gli> gus axes_2d 1 1 1981.5 0 1 5
```

Note: By specifying the beginning and ending labels on the X axis as 1981.5 and 1992.5, respectively, the Bars are spaced from the endpoints of the axis.

Fill the Bar Graph

You may elect to fill each bar by issuing a GKS command similar to:

```
gli> gks set fill int_style solid
```

Many GLI commands can be abbreviated. For instance, INTERIOR can be invoked by entering INT. All commands can be customized by using symbol assignments. (See Defining Symbols, Chapter 3.)

Add Color to the Bar Graph

You may also decide to change the default fill color (black) to some other color:

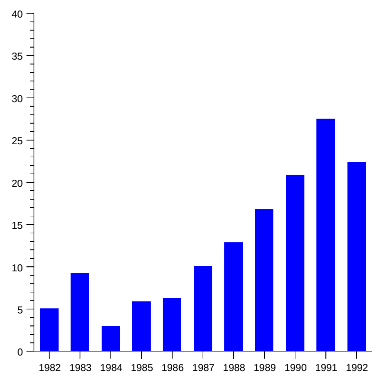
```
gli> gks set fill color blue
```

Draw a Bar Graph

Now draw the graph by entering:

```
gli> gus bar_graph x y
```

FIGURE 2-7 Create a Bar Graph



Delete Data Items and Clear Display

Before the next example, use the abbreviated INITIALIZE command to re-set the GLI environment to its default state which deletes the assigned variables.

```
gli> init
```

Plot 2-D Scatter Diagram

This section demonstrates how to use command line instructions to draw a scatter plot from two-dimensional data. You will also customize the graph using GKS instructions.

Define Display

Set the viewport and window similar to that used by the Histogram example (see Define Display on page 2-12). Again, these commands may not need to be re-entered if you have not changed the viewport or window, or if you have not quit GLI since the display commands on page 2-12 were entered.

Read Data from a File

Enter the data file into GLI using the READ command.

```
gli> read temp.dat x, y
 21 lines read from file temp.dat
```

Set Scales

Set two-dimensional scales according to the data and draw axes for the graph.

```
gli> gus autoscale_2d x, y
gli> gus axes_2d
```

Define Polymarker

Change the default polymarker type, color, and size with the following GKS commands:

```
gli> gks set pmark type circle
gli> gks set pmark color red
gli> gks set pmark size 3.0
```

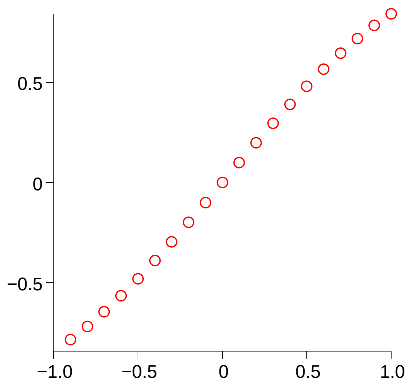
To control polymarker attributes, refer to page 6-48 through page 6-51.

Plot Polymarkers

Plot the X and Y values using the GUS POLYMARKER command.

```
gli> gus polymarker x, y
```

FIGURE 2-8 Drawing a Scatter Plot



Delete Data Items and Clear Display

Before the next example, use the abbreviated INITIALIZE command to re-set GLI to its default state which deletes any assigned variables or functions.

```
gli> initialize
```

Or, you can abbreviate this to:

```
gli> init
```

Plot a Contour Graph

This section demonstrates how to use command line instructions to create a contour graph.

Define Display

Use the DISPLAY1 command procedure (or the commands on page 8) to set the display area where the graphic will be drawn.

```
gli> @display1
```

Read Data from a File

The contouring utility requires regularly-spaced data in the X and Y directions with an assigned value Z at the intersection of each X Y point. See the GRIDIT command in Chapter 3 if you want to contour irregularly-spaced data. For the purposes of this example, you will utilize Z value data from a GLI demonstration file and create the corresponding X and Y points.

Use the READ command to enter the data contained in XDEMO5.DAT from the DEMO directory into the variable Z.

```
gli> read demo/xdemo5.dat z
 1667 lines read from file demo/xdemo5.dat
```

Note: If you started GLI from a different directory than where it was loaded, you will need to specify the absolute pathname to XDEMO5.DAT.

Use the PRINT command to determine the minimum and maximum Z values of the data. You will use these values to define other variables for creating contours.

```
gli> print min(z) max(z)
 2870 4346
```

Define Variables

Create appropriate X and Y values to correspond to the Z data. Use the assignment statement (:=) to define data values in the range (..) of 1 to 100.

```
gli> x := 1..100
gli> y := 1..100
```

Note: When defining data values with arithmetic expressions, use "=" when assigning a value to a function and ":=" when assigning a value to a variable. See page 2-54 for a discussion of the difference between these assignment statements.

Now create a variable (H) that specifies the contour intervals distributed amongst the range of Z values (from a minimum of 2800 meters to a maximum of 4400 meters in increments of 100).

```
gli> h := 2800(100)4400
```

Set the 3-D Drawing Space

Use the GUS SET SPACE command to set the limits of the abstract Z axis (used for mapping three-dimensional objects onto two-dimensional surface) and to set the rotation (0˚) and tilt (90˚) angles, in degrees.

```
gli> gus set space 2800 4400 0 90
```

Set Text for Labeling Contour Lines

GLI offers both hardware and software fonts for labeling graphics. Software fonts can be rotated and tilted and therefore produce better results for specialized labeling, such as contour labels.

Use the GKS SET TEXT FONTPREC command to change the default text font and precision to a software, stroke font (-1 Cartographic Roman).

```
gli> gks set text fontprec -1 stroke
```

Next, change the height (to 1.5% the size of the default window).

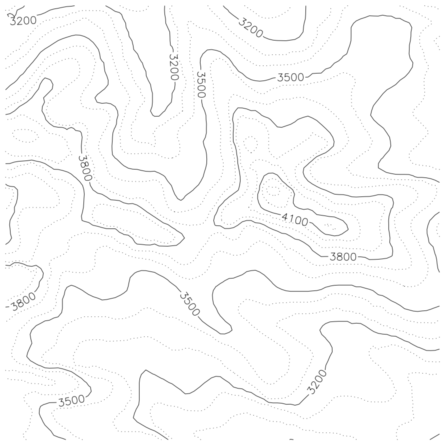
```
gli> gks set text height 0.015
```

Draw a Contour with Labeled Lines

Use the GUS CONTOUR command to plot variables X, Y, H, and Z, to specify the number of points displayed in the X direction (100), and to label every third contour.

```
gli> gus contour x y h z 100 3
```

FIGURE 2-9 Plot Contours



For more information on contour and text utilities, refer to the GUS chapter. For more information on text primitives, refer to the GKS chapter.

Clear the Display

Clear the display without deleting the data values or re-setting the GLI environment:

```
gli> gks clear_ws
```

Draw a Color-filled Contour

The GUS SURFACE utility produces a color-filled contour graph when the tilt of the 3-D space is set to 90 degrees as in the previous example. To create a color-filled contour plot from data in the last example, use the three-dimensional surface command:

```
gli> gus surface x y z colored_mesh
```

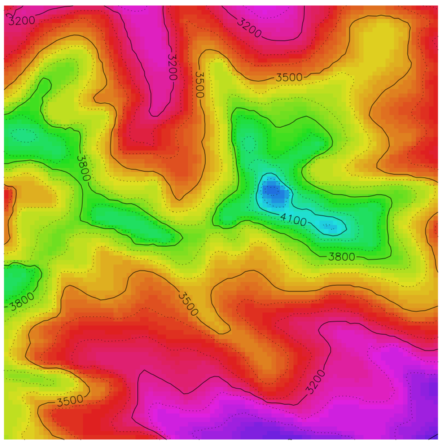
Note: If you have deleted the data set or altered commands from the proceeding example, re-enter those commands before invoking the surface utility.

Overlay Contour Lines With Labels

Overlay lines indicating the contour levels with labels. Set text height and the font for better looking labels.

```
gli> gks set text height 0.015
gli> gks set text fontprec -1 stroke
gli> gus contour x y h z 100 3
```

FIGURE 2-10 Plot Color Contours



Clear the Display

Clear the display without deleting the data values.

```
gli> gks clear_ws
```

Color-filled Contours with Legends

Using data sets from the previous example, draw a color-filled contour and corresponding legend.

Set the Viewport

Leave the SET SPACE command unchanged from the last exercise, but define the viewport to leave a margin for the legend to the right of the contour plot.

```
gli> gks set viewport 0.05 0.8 0.05 0.95
```

Draw a Color-filled Contour

The SET SPACE command from the last example defined the tilt at 90° which will produce a color-filled contour plot from the data assigned to variables X, Y, and Z when GUS SURFACE is invoked.

```
gli> gus surface x y z colored_mesh
```

Change the Viewport

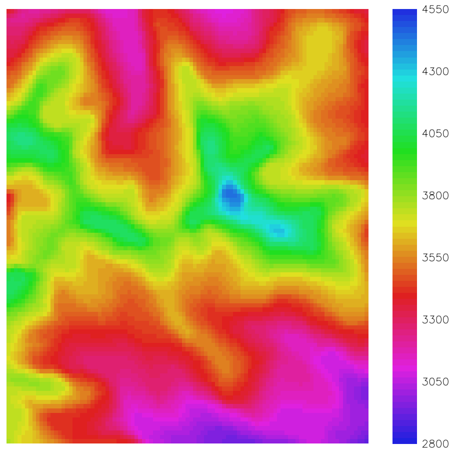
```
Now define the area containing the legend.
gli> gks set viewport .85 .9 .05 .95
```

Apply the Legend

Generate the legend using colors that corresponds to the Z values in the contour plot.

```
gli> gus colormap
```

FIGURE 2-11 Contours with Legend



Clear the Display

Clear the display without deleting variables or affecting other GLI settings.

```
gli> gks clear_ws
```

Draw a 3-D Surface

Again, using the data and most of the command settings from the proceeding contour example, you can produce a 3-D wireframe or shaded surface.

Define the Viewport and Orientation

Set the viewport to include a margin for a 3-D axis, and set the text height.

```
gli> gks set viewport 0.09 0.91 0.09 0.91
gli> gks set text height 0.021
```

Use the SET SPACE command to scale the Z axis according to the data range, and define the rotation (30°) and tilt (45°) of the graph.

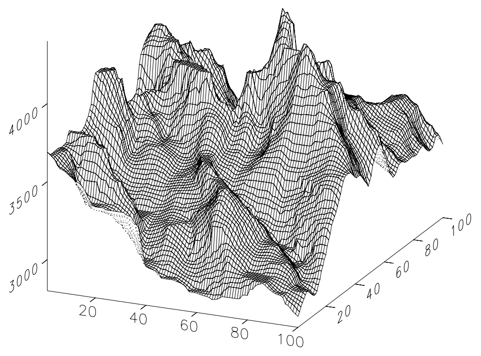
```
gli> gus set space 2800 4400 30 45
```

Plot the 3-D Surface

Draw a wireframe surface and add 3-D axis.

```
gli> gus surface x y z mesh
gli> gus axes_3d
```

FIGURE 2-12 Plot a Surface



Clear the Display

Again, clear the display without deleting the data items or changing the viewport or set space setting.

```
gli> gks clear_ws
```

Draw a Shaded Surface

Using the same data and viewport and set space settings from the preceding example, draw a color shaded surface plot with 3-D axis:

```
gli> gus surface x y z colored_mesh
gli> gus axes_3d
```

The new graph will be drawn on your screen.

Delete Data Items and Clear Display

Before the next example, use the abbreviated INITIALIZE command to re-set GLI to its default state which clears the display and deletes any assigned variables or functions.

```
gli> init
```

Plot 3-D Data

This section demonstrates how to use command line instructions to plot a line graph from three-dimensional data.

Define Display

Set the display area where your graphic will be drawn using the DISPLAY1 command procedure, or set the display area using the commands found on page 8.

```
gli> @display1
gli> gks set viewport 0.15 0.9 0.15 0.9
```

Create Data Items

Define a three-dimensional array of data by applying mathematical functions to the variables X, Y, and T:

```
gli> t := 1(.1)40
gli> x := 40+t*cos(t)
gli> y := 40+t*sin(t)
```

Set Space

Set the three-dimensional drawing space and the angle of rotation and tilt for the axis.

```
gli> gus set space 1 40 30 45
```

Set Axes

Draw three-dimensional axes according to the data range.

```
gli> gus axes_3d
```

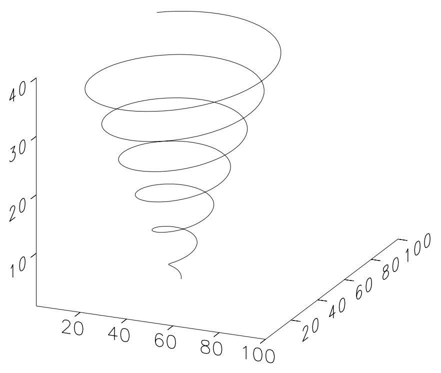
To modify axes style, refer to page 5-13 . To alter axes orientation, refer to page 5-15.

Plot XYZ Graph

Finally, plot the X, Y, and T values using the CURVE command:

```
gli> gus curve x, y, t
```

FIGURE 2-13 Plotting an XYZ Line Graph



Delete Data Items and Clear Display

To re-set the GLI environment to its default state, use the abbreviated INITIALIZE command. As you have seen from previous examples, this command also clears the display and deletes any assigned variables or functions.

```
gli> init
```

Command Line Editing

Your keyboard provides editing functions to correct any mistakes you might make while typing commands. The following table describes these keys.

TABLE 2-1 Line Editing Terminal Keys

Key	Function
CTRL/A (or [F14])	Toggles between overstrike mode and insert mode.
CTRL/B (or Up Arrow)	Displays the last command line issued. If pressed again, displays the previous command line in the recall buffer. The recall buffer stores the 20 most recently issued commands.
CTRL/C (or [F6])	During command entry, cancels command processing. In general, CTRL/C cancels the operation in progress.
CTRL/D (or Left Arrow)	Moves the cursor one character to the left.
CTRL/E	Moves the cursor to the end of the line.
CTRL/F (or Right Arrow)	Moves the cursor one character to the right.
CTRL/H (or BACKSPACE, [F12])	Moves the cursor to the beginning of the line.
CTRL/J (or LINEFEED, [F13])	Deletes the word left of the cursor.
CTRL/L	Causes the terminal to go to the beginning of the next page.
CTRL/R	Retypes the current input line and leaves the cursor positioned at the end of the line.
CTRL/T	Displays statistical information about your GLI session.
CTRL/U	Deletes characters from the beginning of the line to the cursor.
CTRL/W	Combines the function of CTRL/R and CTRL/L.
CTRL/Z (or [F10])	Signals “end of the file” for data entered from the terminal. CTRL/Z is displayed as “Exit”.
DELETE	Moves the cursor back one character (on the current line) and deletes that character.
Down Arrow	Displays most recent command line in the recall buffer.
RETURN	Transmits the current line to the system for processing.
TAB	Moves the cursor to the next tab stop on the terminal.

Using AUTOPLOT

AUTOPLOT is an easy to use, form-driven interface for ANSI terminals, or workstations with ANSI terminal emulation (e.g. xterm, VT100). You fill in blanks and enter one or more data files that AUTOPLOT reads, analyzes, and graphs as a linear or logarithmic chart.

The versatility of AUTOPLOT gives you the ability to graph multiple data sets with a few simple keystrokes. AUTOPLOT automatically adjusts axes according to the data and applies different linetypes and labels (line styles) for each data set.

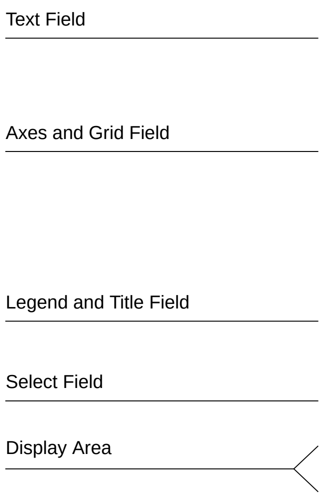
Start AUTOPLOT

Type `AUTOPLOT` on the GLI command line to invoke the interface.

```
gli> autoplot
```

GLI loads and displays the AUTOPLOT interface on your screen:

FIGURE 2-14 AUTOPLOT Screen Layout

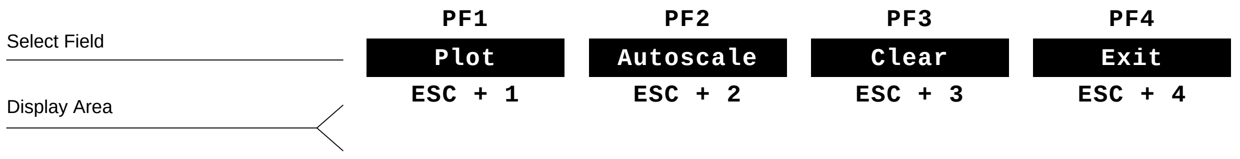


The AUTOPLOT interface is a menu of text and input fields with a status area at the bottom for displaying messages. You communicate with AUTOPLOT by specifying information in its input fields and pressing function keys:

- Text
  Use the keyboard to enter information in a text field. Press the RETURN key or use the ARROW keys to move to the next field.
- Toggle
  Use the Right and Left arrow keys to choose an item from an input field offering a list of choices.

- Select
  ANSI terminal users (e.g., DECstation, VT330, VT340) press the corresponding PF key (PF1 through PF4) to execute the specified command. When using non-ANSI keyboards (e.g. Sun, HP), press the ESC key and then the number 1 through 4 keys. See Figure 2-10, below.

FIGURE 2-15 Keyboard Select Keys



- Display
  AUTOPLOT displays informative messages in this area.

Plot 2-D Data

As many as four data sets can be plotted on one chart using AUTOPLOT. Axis, tick marks, output primitives and smoothing are selected for you, or you can easily modify these options and add a title and legend.

Read Data from a File
By default, AUTOPLOT begins with the cursor positioned in the PLOT SPECIFICATION FILE text field. Use the keyboard to enter the name of your data file (use TEMP.DAT from the Using GLI Commands example). Press RETURN to enter the data in the input field.

You can now plot the data by pressing PF1 (ESC + 1 on non-ANSI terminals) or you can change the AUTOPLOT default options as described below.

Select Output Primitive
Using the ARROW keys, move to the OUTPUT PRIMITIVE selection. Change the selection (Polyline, Polymarker, Spline, Linear Fit, FFT, Inverse FFT) by using the Left and Right arrow keys. By default, the first polyline drawn is a solid line. Subsequent line styles (up to four may appear on one graph) are various dashed line styles. Likewise, the polymarkers change. Labels for lines and markers are explained below. Press RETURN to enter the data in the input field.

Set Smoothing Level
Move to the SMOOTHING LEVEL selection using the arrow keys. From the keyboard enter a value from 1 (least smoothing applied) to 20 (maximum smoothing).

Add Axes
Use the ARROW keys to move through AUTOPLOT until you reach the AXES toggle field. Use the Right and Left ARROW keys to choose between Linear and Logarithmic (X_Log, Y_Log, and XY_Log) axes for your graph.

Axes Values
By default, AUTOPLOT draws the axes so they are even with maximum X and Y values. AUTOSCALE (PF2 or ESC +2) will extend the axes to better fit your data. Alternatively, the axes can be controlled by entering the MIN/MAX values using the keyboard.

Add Text

Move through the interface with the ARROW keys until you reach the ANNOTATION text fields.

Enter a title and, if needed, a subtitle for your chart consisting of as many as 34 characters each. Titles and subtitles are centered at the top of the plot. The X and Y axes can be labeled in a similar manner.

Add Legend

Legends, if selected, are added to the left of theY axis. The legend can consist of four labels containing two text lines (20 characters maximum) each.

Modify Tick Marks

Using the ARROW keys, position the cursor on the MAJOR Y-TICK COUNT field. Using the keyboard, input the number of minor tick marks between major Y axis tick marks. 0 or 1 imply no minor tick marks (see Table 2-3). Follow this procedure to modify all axes tick marks.

Add Grid

Move through the interface with the ARROW keys until you reach the GRID toggle field. Use the Right and Left ARROW keys to toggle between displaying or suppressing grid lines at major tick marks.

Position the Graph

Move to the WINDOW field. Use the Right and Left ARROW keys to control the position on your device where the plot will be drawn.

Size the Plot

Using the arrow keys, move to the SIZE field. The Left and Right ARROW keys allow you to size the graph on your device to correspond to sizes A3, A4 and A5.

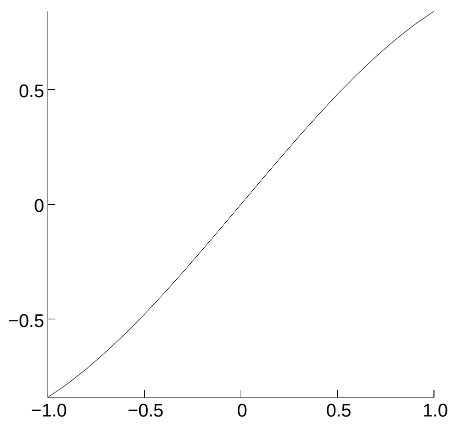
TABLE 2-2 AUTOPLOT Output Size

Size	Dimension
A3	0.420m x 0.210m (16.54” x 11.69”)
A4	0.297m x 0.210m (11.69” x 8.27”)
A5	0.210m x 0.1485m (8.27” x 5.85”)

Plot

Press the PF1 (or ESC + 1) key to plot the XY graph. The graphic is displayed on your device.

FIGURE 2-16 Plotting an XY Graph with AUTOPLOT



Clear
If you want to make changes to your graph and re-plot, press the PF3 (or ESC + 3) key to clear the workstation display area. To return the AUTOPLOT interface to your screen, enter RETURN. Always clear the display area before plotting new data files.

Plot Additional Curves
If you want to plot another data file on the same graph, simply enter the new data file name in the PLOT SPECIFICATION FILE text field. AUTOPLOT automatically selects a different polyline (or polymarker, depending on your initial output primitive choice) for subsequent plots, to a maximum of four curves. You can change the AUTOPLOT attributes at any time.

AUTOPLOT and GKS
It is possible to change the color of the polymarker or the polyline by using GKS commands before you call AUTOPLOT from the GLI command line:

```
gli> gks set pline color_index blue
gli> autoplot
```

This will change the default color of all polylines drawn by AUTOPLOT to blue.

Other Commands

AUTOPLOT's other input fields provide additional functionality for plotting and analyzing your data.

TABLE 2-3 AUTOPLOT Input Fields

Input Field	Description
File	Specify a data file to be plotted.
Output Primitive	Use the Right and Left arrow keys to select the primitive.

Smoothing Level	Specify a line smoothing level in the range 0-20.
Axes	Use the Right and Left arrow keys to select the axes.
Grid	Use the Right and Left arrow keys to display grid lines.
Minimum X	Specify the minimum value for the X axis.
Units Between X Ticks	Specify the interval between X axis tick marks.
Maximum X	Specify the maximum value for the X axis.
Major X-Tick Count	Specify the number of minor tick marks between major X axis tick marks. 0 or 1 imply no minor tick marks.
Minimum Y	Specify the minimum value for the Y axis.
Units Between Y Ticks	Specify the interval between Y axis tick marks.
Maximum Y	Specify the maximum value for the Y axis.
Major Y-Tick Count	Specify the number of minor tick marks between major Y axis tick marks. 0 or 1 imply no minor tick marks.
Main Title	Specify a title of up to 34 characters. The title is centered across the top of your display.
X Axis	Specify a title of up to 20 characters for the X axis.
Subtitle	Specify a subtitle of up to 34 characters. This title is centered just below the main title.
Y Axis	Specify a title of up to 20 characters for the Y axis.
Legends	Specify up to four legend labels each containing one or two lines of 1 to 20 alphanumeric characters. When plotting multiple data sets the first label is associated with the first plotted line, the second label with the second line, and so forth.
Window	Select a portion of the viewport for plotting the chart.
Size	Select the viewport size for your chart.
PF1 (Plot)	Press the PF1 key to plot the graphic.
PF2 (Autoscale)	Press the PF2 key to set automatic scaling.
PF3 (Clear)	Press the PF3 key to clear the graphic from the workstation window.
PF4 (Exit)	Press the PF4 key to exit AUTOPLOT.

Exit AUTOPLOT

Press the PF4 key to exit AUTOPLOT and return to the GLI prompt.

Using SIMPLEPLOT

SIMPLEPLOT is an interactive utility originally developed for signal processing applications. Using SIMPLEPLOT, you can retrieve two-dimensional data files; display, edit, and filter the data; verify the quality of the signal; and/or write modified data to a file.

SIMPLEPLOT is compatible with ANSI-standard terminals or X WIndow systems. This interface displays the plot in a default-sized window. If the plot is displayed on a X Window device, it can be iconified or repositioned on your screen. In the current version of GLI, the output window for SIMPLEPLOT cannot be re-sized.

To issue a command, enter the first letter of the appropriate command listed in the SIMPLEPLOT menu and strike RETURN.

Start SIMPLEPLOT

Type `SIMPLEPLOT` on the GLI command line to invoke the interface.

```
gli> simpleplot
```

FIGURE 2-17 SIMPLEPLOT Screen Layout

SIMPLEPLOT Design

SIMPLEPLOT follows a basic functional design which allows you to:

- Retrieve data
- Edit data
- Filter data
- Verify signal quality
- Write data to a file

The following sections will demonstrate several SIMPLEPLOT applications. At the end of this section is a flow-chart displaying SIMPLEPLOT’s overall usage.

Truncate Range of Data

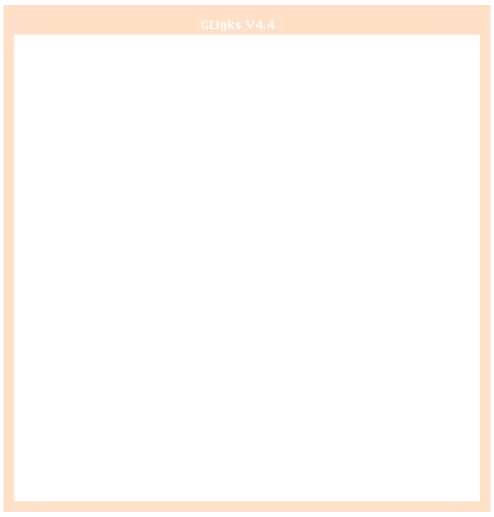
In this example of SIMPLEPLOT, you will read and display a two-dimensional data file, select a range of the data from the graph, and write it to a file.

Retrieve Data

Press the N (New file) key to load your data file. SIMPLEPLOT prompts you for the name of your data file (for example, use DEMO/DEMO13_1.DAT). When you press the RETURN key, SIMPLEPLOT reads and plots the data as an XY graph.

```
SIMPLEPLOT> n
Data file: demo/demo13_1
333 lines read from file demo/demo13_1
```

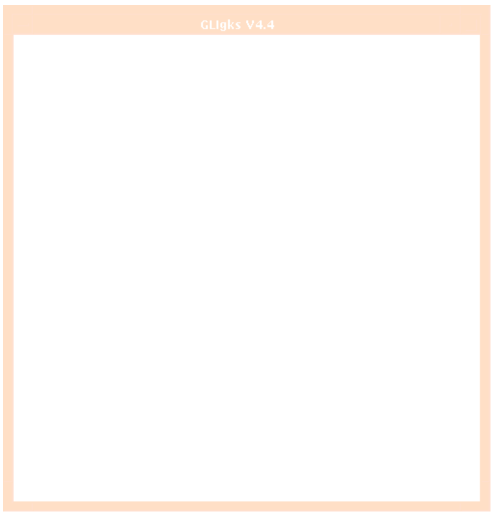
FIGURE 2-18 SIMPLEPLOT Plots an XY Graphic



Edit Data

Press the T key on your keyboard to select the Truncate function. The mouse pointer will change to a cross. Move the pointer to the first data point you want included in your truncated data file and press the left mouse button. Reposition the pointer to the left of the second point (representing the last data point you want included) and press the left mouse button.

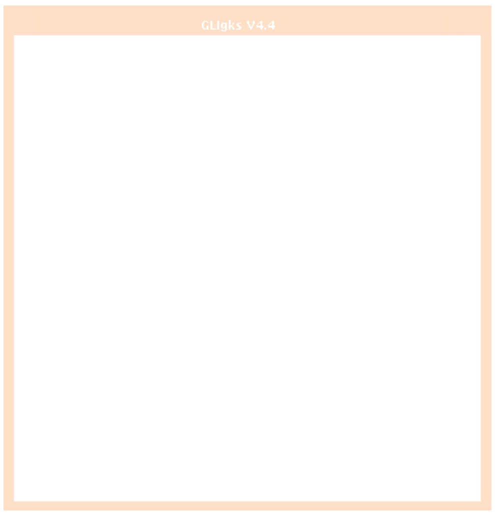
FIGURE 2-19 Select a Data Point



Note: Pressing and releasing a mouse button is called “clicking.” Positioning the mouse pointer is called “pointing.”

SIMPLEPLOT re-plots the graph using the truncated data range.

FIGURE 2-20 Truncate Data



Write Data to a File

Press the W (Write) key to write the data to a file. SIMPLEPLOT prompts you for the name of the new data file.

```
Simpleplot> w
Data file: truncate
232 lines written to file truncate.dat
```

Note: The number of lines written to the file will vary, depending on the data points selected.

Modify Data Points

Using SIMPLEPLOT, you can read and display a two-dimensional data file, select a data point on the graph, change its position, and write the modified data to a file.

Retrieve Data

Load your data file.

```
Simpleplot> n
Data file: demo/demo13_1
333 lines read from file demo/demo13_1.dat
```

Edit Data

Press the M key to select the Modify function. The pointer changes to a cross, indicating the current data point that will be modified. By default, SIMPLEPLOT begins with the first data point in the file and continues along the X axis.

Point and click the left mouse button to select the new Y position. SIMPLEPLOT re-plots the graph using the revised data. Select Modify again to change the Y position of the second data point (and so on).

If you want to modify a point other than the default first point, press the R key to select the Range function. Point and click the left mouse button on a starting point and then on an ending point. On color systems, the graph segment selected will be highlighted in red. Press the M (Modify) key to change the position of the data point (the first point in the selected range). SIMPLEPLOT re-plots the graph using the revised data. If you select Modify again, SIMPLEPLOT continues with subsequent points on the X axis until you re-select the range.

If you want to modify a variety of points in different locations, select the Range function and point and click the left mouse button twice in rapid succession (double-click) on the XY graph. Now select Modify to change the position of a data point. SIMPLEPLOT re-

plots the graph using the revised data. Hereafter, SIMPLEPLOT allows you to select any data point on the graph and change its Y value.

Write Data to a File

Press the W key and enter a file name to write the data to a new file.

Remove Linear Trend

Before plotting Fast Fourier Transformations, you may want to remove the Linear Trend from your data.

Retrieve Data

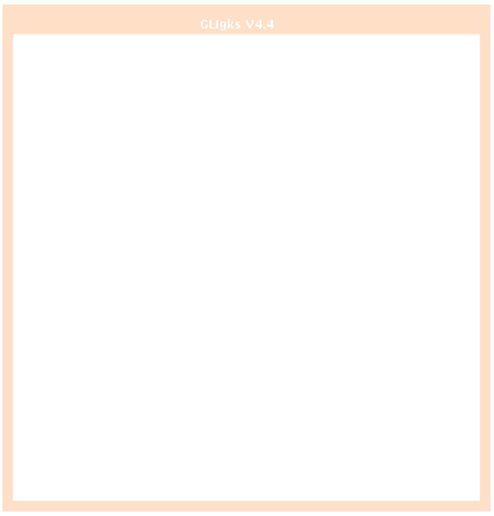
Load your data file (use the GLI demonstration file DEMO13_1.DAT).

```
Simpleplot> n
Data file: demo/demo13_1
333 lines read from file demo/demo13_1.dat
```

Edit Data
Press the L key to select the Linear Trend function. Select a data range by pointing at a data point and clicking the left mouse button once to select the beginning of the data range. Then point at an ending data point and click the left mouse button again to select the end of the data range.

SIMPLEPLOT automatically re-plots the graph.

FIGURE 2-21 Remove the Linear Trend



Select a Data Range and Apply a Filter

Using SIMPLEPLOT, you can read and display a two-dimensional data file, plot the Fast Fourier Transformation, select a portion of the data, and apply a Bandpass filter.

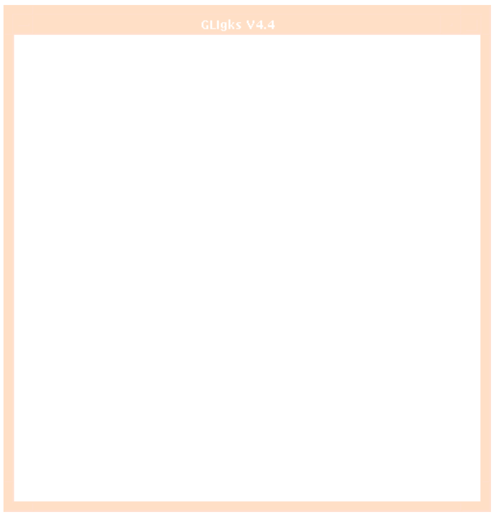
Retrieve Data
Load your data file.

```
Simpleplot> n
Data file: demo/demo13_1
333 lines read from file demo/demo13_1.dat
```

Apply Filter
Press the F key to select the Fast Fourier Transformation (FFT) function.

SIMPLEPLOT re-plots the graph using the FFT.

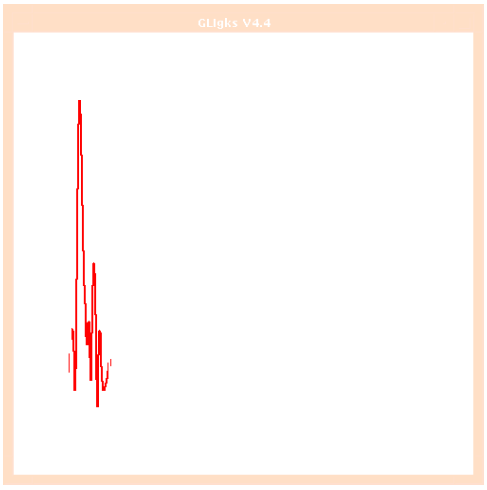
FIGURE 2-22 Fast Fourier Transformation



Press the R key for the Range function and define a range of data to filter. Now press the B key to apply the Bandpass filter to the selected data range.

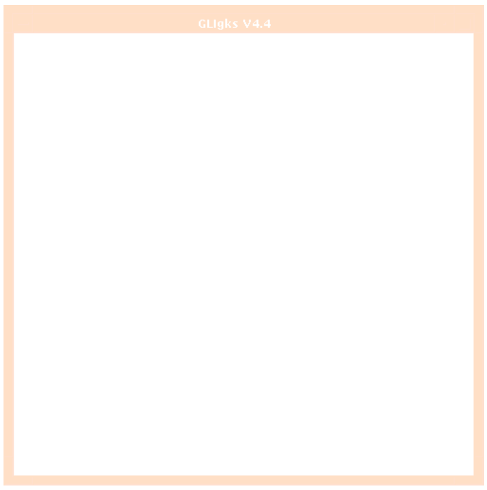
SIMPLEPLOT overlays the Bandpass filter on the graph.

FIGURE 2-23 Bandpass Filter



Press the I key to apply the Inverse FFT and to display the filtered data.

FIGURE 2-24 Inverse Fast Fourier Transformation



Zoom on a Range

Select and display only a range of data from a file.

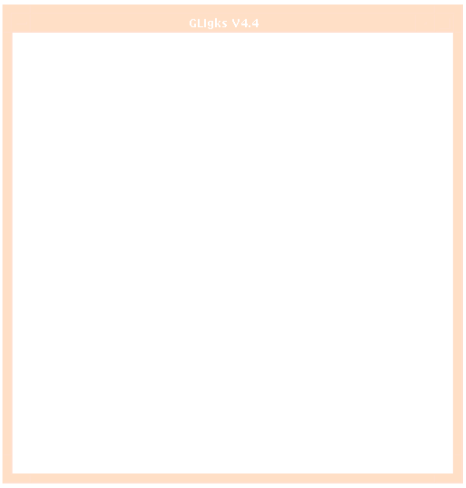
Retrieve Data
Press the O (Old File) key to restore the previous data file.

```
Simpleplot> o
333 lines read from file demo/demo13_1.dat
```

Invoke Zoom and Select Data Points
Press the Z key to select the Zoom function and click on two points on the graph. The graph will be re-drawn, displaying only the graph segment you selected.

```
Simpleplot> Z
```

FIGURE 2-25 Zoom in on Data Range



Restore the Graph
To restore the graph to it original form use the U (Undo) key. The Undo command reverses the effect of the last command and re-plots the graph.

```
Simpleplot> u
```

Other Commands

SIMPLEPLOT includes additional functions to help you plot, edit and analyze your data.

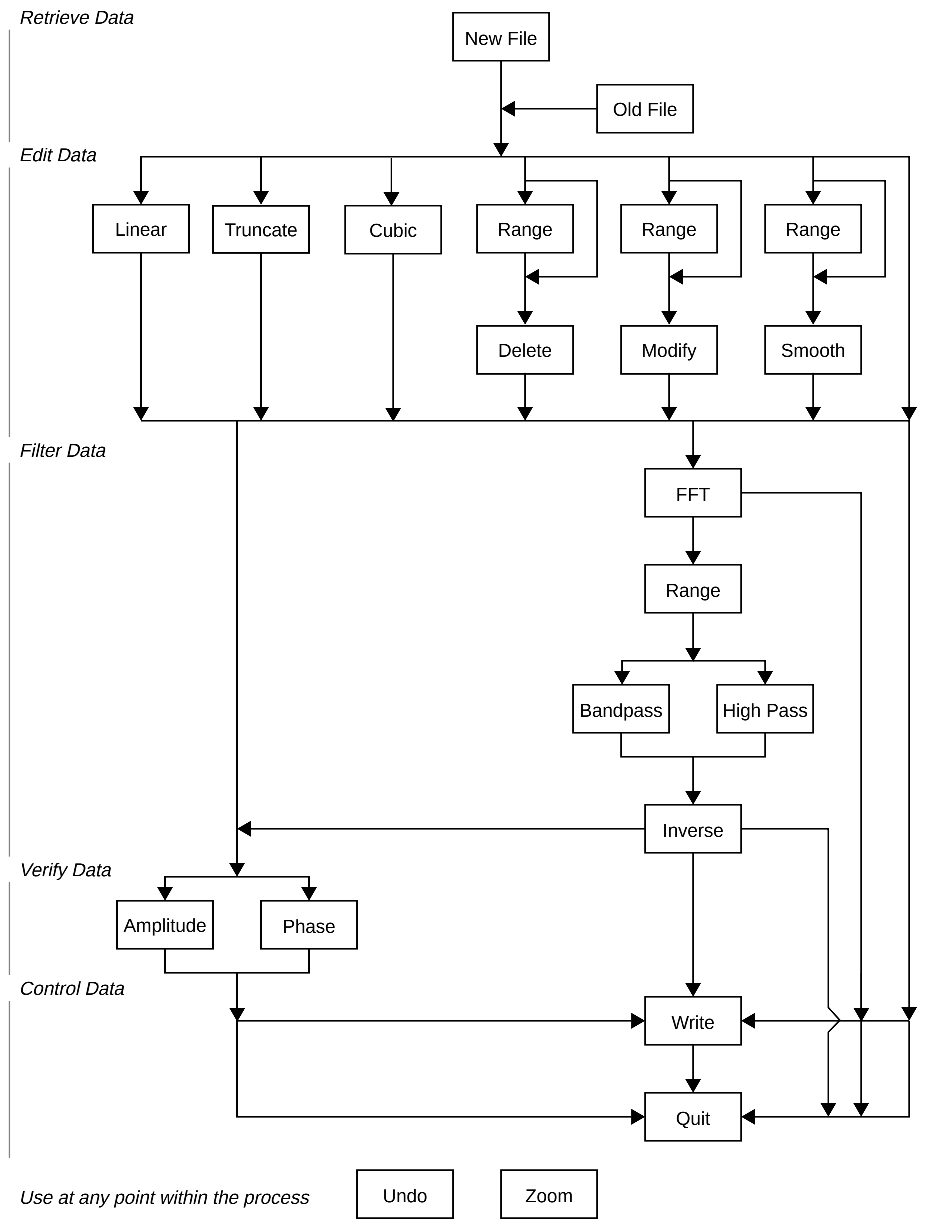
TABLE 2-4 SIMPLEPLOT Commands

Command	Description
Amplitude	Verifies signal quality by overlaying the computed amplitude over the curve.
Bandpass	Applies the bandpass filter to selected points on the curve. Select the Range command first.
Cubic Spline	Applies a cubic spline fit to points on the curve.
Delete	Deletes the selected data points. Must be used with the Range command.
FFT	Computes and displays the Fast Fourier Transformation.
High Pass	Applies the high pass filter to selected points on the curve. Select the Range command first.
Inverse FFT	Computes the Inverse Fast Fourier Transformation.
Linear Trend	Removes the linear trend from the curve. Select the start and end of the slope.
Modify	Modifies a data point on your graph. May be used with the Range command.
New file	Reads data from the specified file.
Old file	Restores the previous (original) data file.
Phase	Verifies signal quality by displaying the computed phase of the curve.
Range	Selects the starting and ending points of a range.
Smooth	Applies a smoothing routine to points on the curve. May be used with the Range command.
Truncate	Truncates anything outside of the selected range. Select the start and end of the range.
Undo	Un-does the last command.
Write file	Writes modified data to specified file.
Quit	Quits SIMPLEPLOT and returns you to the GLI prompt.
Zoom	Zooms in on the selected range of data. Select the start and end of the range.

Exit

Press the Q key to quit SIMPLEPLOT.

FIGURE 2-26 SIMPLEPLOT Flow Chart



Using XUI

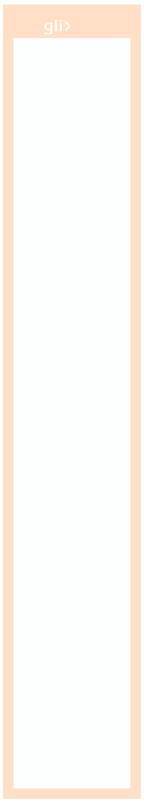
XUI is a point and click version of the GLI command line interface compatible with X Windows. Using the mouse, you select GLI, GUS, and GKS commands from pop-up windows to plot your data as two- and three-dimensional graphics.

Type XUI on the GLI command line to invoke the interface.

```
gli> xui
```

GLI loads and displays the XUI interface on your screen:

FIGURE 2-27 XUI Screen Layout



XUI and the Mouse Buttons

XUI consists of point-and-click menus that give you access to GLI commands. Your mouse buttons allow you to navigate through the XUI interface.

Left Mouse Button

Positioning the pointer and clicking the left mouse button (MB1) selects a menu item in the XUI interface.

Middle Mouse Button

Clicking the middle mouse button (MB2) returns you to the main XUI menu.

Right Mouse Button
Clicking the right mouse button (MB3) on some menu items will display popup menus that allow you to set options or parameters.

Note: You must have input focus with a menu window to interact with XUI.

Plot 2-D Data

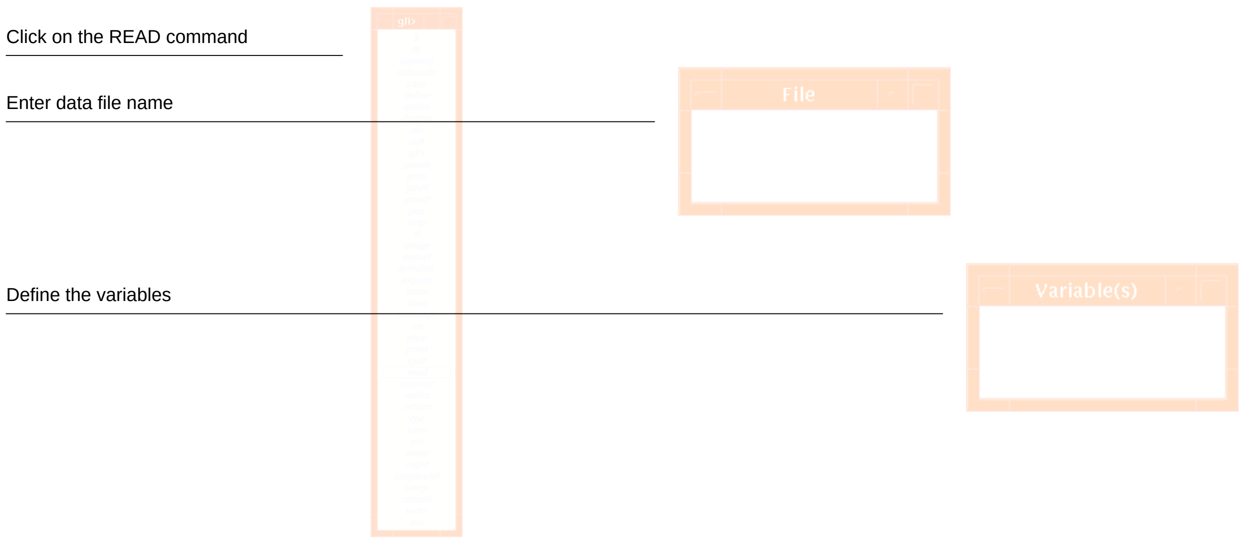
This section demonstrates how to use the XUI interface to plot a graph from two-dimensional data.

Read Data from a File
Move the mouse to point at the `READ` command in the XUI command list. Click the left mouse button to execute the command.

A pop-up dialog box appears, prompting you to enter the name of your data file from the keyboard (use the TEMP.DAT file you created earlier in this section) and press the RETURN key. Always press the RETURN key to activate XUI dialog boxes.

A second dialog box appears, asking how you want the data read. (If you are using TEMP.DAT, type `X Y` to describe the X and Y values in that data file.)

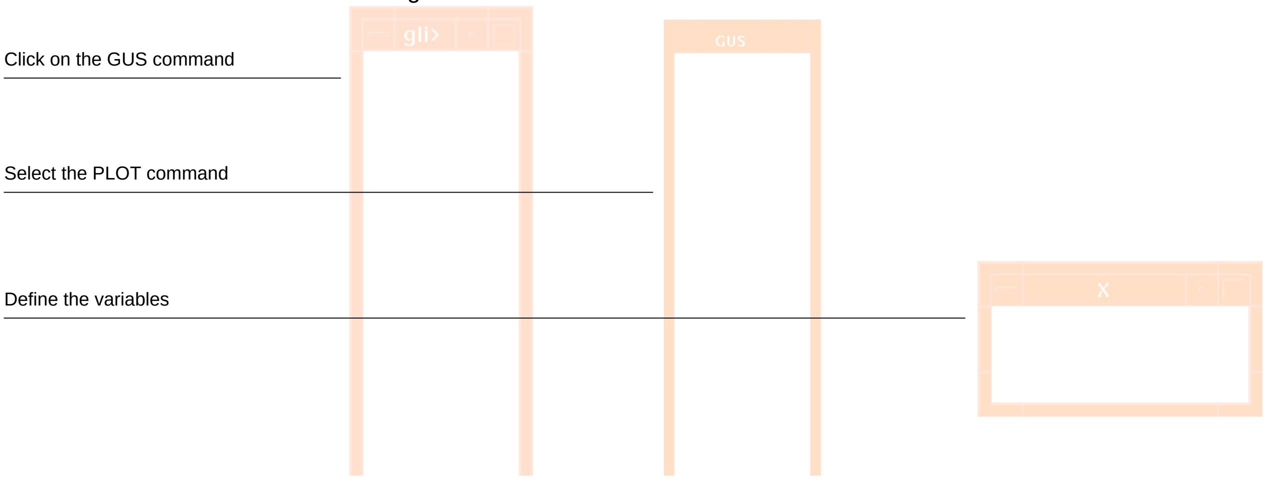
FIGURE 2-28 Reading Data with XUI



Plot Data
Click on the GUS command in the XUI command list. A pop-up menu appears, listing GUS commands.

Select GUS `PLOT`. A dialog box appears, asking how you want the data read. Type `X Y`.

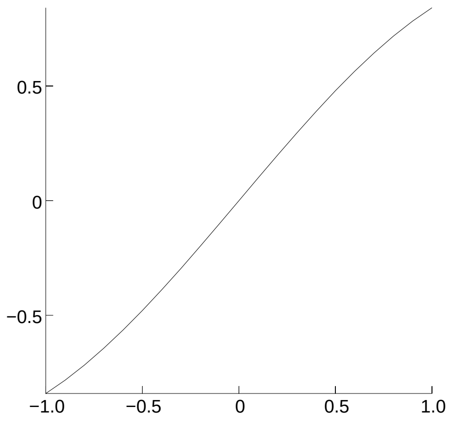
FIGURE 2-29 Plotting Data with XUI



Note: The middle mouse button returns you to the top level of the XUI interface.

The graphic is displayed on your output device.

FIGURE 2-30 Plotting an XY Graph with XUI



Other Commands

XUI provides other functions for plotting and analyzing your data. Some commands (noted with **) are <u>not</u> interactive and cannot be accessed from XUI. These commands must be written into a macro to be used (explained later in this chapter).

TABLE 2-5 XUI Commands

Command	Description
$	Escape to operating system.
@	Execute macro.
append	Append data item to an output file.
calculate	Display result of arithmetic expression.
case	Test value of specified symbol and execute commands.
define	Define an identifier as a function, symbol, or variable.
delete	Delete an identifier.
display	Displays menu loaded with the LOAD command.
do	Executes GLI instructions separated by a semicolon (;).
exit	Exit GLI, storing data structure in a file.
gks	Escape to GKS.
gosub	Transfer control to a labeled subroutine.**
goto	Transfer control to a labeled statement.**
gridit	Constructs a regular grid from irregularly distributed points.
grsoft	Invoke the GR interface (if the GR software is present).
gus	Escape to GUS.
help	Invoke GLI Help facility.
if	Test value of specified expression and execute commands.
image	Escape to IMAGE.
import	Import and incorporate an additional data file.
initialize	Re-initialize GKS and re-start GLI.
inquire	Define symbol.
learn	Begin collecting all commands for a subsequent SAVE.
load	Load menu from specified file and prepare for DISPLAY.
message	Display specified character string on screen.
on	Execute command upon specified Form Driver command.
pipe	Open a pipe for command input.
print	Print data items to the screen.
quit	Quit GLI.
read	Read input file and assign contents to variables.
recover	Read journal file and execute commands.
redim	Re-sample an array.
return	Terminate GOSUB routine.
rpc	Remote Procedure Call (RPC).
save	Save commands in specified file.
set	Set system defaults and information
show	Display identifier description.
sight	Invoke the SIGHT interface.
simpleplot	Invoke the SIMPLEPLOT interface.
sleep	Suspends execution for a time.
smooth	Smooth a sequence of data points.
write	Write data items to an output file.
xui	Exit XUI and return to the GLI prompt.

Exit XUI

Click `on XUI` to exit XUI and return to the GLI prompt (`gli>`).

If you click on EXIT, GLI records and stores the commands you issued in a journal file (GLI.JOU) and returns you to your operating system.

If you click on QUIT, GLI returns you to your operating system.

Create a Simple X Windows Interface

You can use GLI commands to create your own X Window interfaces, similar to the XUI interface, that are tailored to your specific applications. The XUI Commands chapter will assist you in using the CHOICE and CASE commands to create customized pop-up menus that execute your commands.

Using SIGHT

SIGHT is a point and click interface, combining object-oriented commands with the Motif environment. Using the mouse, you select commands from pull-down, pull-right and pop-up menus to communicate with all aspects of GLI. All commands supported by the GLI, GUS, and GKS command languages are available to you when using SIGHT.

SIGHT is compatible with systems running the Motif implementation of X Windows.

Before you begin the SIGHT example discussed in this section and while you are at the GLI prompt (gli>), create a sample data file (LOG.DAT) using the following commands:

```
gli> x = 1..20
gli> y = x**2
gli> write log.dat x,y
21 lines written to file log.dat
```

Start SIGHT

Type `SIGHT` on the GLI command line to invoke the interface.

```
gli> sight
```

GLI loads and displays the interface on your screen:

FIGURE 2-31 SIGHT Screen Layout

SIGHT Menu Interface

Menu Bar

The menu bar contains the main headings for the SIGHT commands listed below:

TABLE 2-6 SIGHT Commands

File Commands	**Description**
Create Drawing…	Create a new drawing.
Open Drawing…	Open a saved drawing and displays its contents.
Save As…	Save current drawing under a new file name.
Close Drawin	Save current drawing to file.
Read Data Frm File...	Read a Data File
Write Data To File...	Write modified data to a file.
Import...	Import CGM - Files.
Print Drawing	Print the drawing to an output device.
Capture Drawing	Print the drawing to an encapsulated PostScript file.
Quit	Close SIGHT with confirmation.
Exit	Close SIGHT, without confirmation.
Edit Command	**Description**
Redraw	Redraw last Object.
Select	Select single, group, previous or next objects.
Deselect	Deselect all currently selected objects.
Cut	Delete the current selection and copy it to the SIGHT clipboard.
Paste	Copy contents of the SIGHT clipboard into the drawing at the specified location.
Pop in Front	Move selected objects in front of other objects.
Push Behind	Move selected objects behind other objects.
Move	Move selected objects to a new location.
Remove	Delete selected objects without copying them to the SIGHT clipboard.
Pick	Transfer object-releated data to GLI frm selected object.
Digitize	Digitize data points from a digitizing tablet..
Clear Drawing	Clear the current drawing and delete all objects.
Tools Commands	**Description**
Create Tool Box	Create a graphical Tool Box.
Line	Create a polyline from data points. You can either read a data file or specify the data points using the mouse.
Spline	Create a cubic spline-fit. You can specify either a natural spline, or a smoothed spline.
Error Bars	Draw vertical or horizontal error bars.

Marker	Create a polymarker from data points. You can either read a data file or specify the data points using the mouse.
Text	Create a character string.
Fill Area	Fill the selected polygon. You can either read a data file or specify the data points using the mouse.
Bar Graph	Draw a standard Bar Graph with Y values determining the length and Y values determining position of the bars.
Plot	Create a line graph from data points. You can either read a data file or specify the data points using the mouse.
Axes…	Draw a pair of labeled axes.
Grid…	Draw a grid.

TABLE 2-7 SIGHT Commands Continued

Attribute Commands	**Description**
Create Attribute Box	Create a graphic Attribute Box.
Line Type	Change the line type.
Line Width	Change the line width.
Line Color	Change the line color.
Marker Type	Change the polymarker type.
Marker Size	Change the polymarker size.
Marker Color	Change the polymarker color.
Text Family	Change the text font.
Text Size	Change the text size.
Text Style	Change the text style.
Text Alignment	Change the text horizontal and vertical alignment.
Text Direction	Change the text writing direction.
Text Color	Change the text color of the selected text.
Fill Color	Change the fill color of the selected polygon.
Fill Style	Change the fill style of the selected polygon.
Fill Index	Change the fill index of the selected polygon

View Commands	**Description**
Select Viewport	Change the size of the viewport.
Set Window	Change the size of the window.
Orientation	Change the drawing page orientation.
Scale	Creates Linear, X Log, Y Log or XY Log axes.
Select Transformation	Set the transformation type for creating graphics.

Customize	Description
Snap Grid	Display an imaginary grid to assist graph and object placement and sizing.

Command Reference

When you issue a SIGHT command, it is stored and displayed in the command reference list. Commands from this list can be invoked by selecting them with your mouse.

Select the command from the command reference list by positioning the mouse pointing on the command (point at the command) and press and release (click) the left mouse button. The command automatically appears in the SIGHT command line (see below).

Status Line

When you select an object in your graph, `SIGHT` displays the status, or attributes, of the object on status line.

SIGHT Command Line

The `SIGHT>` prompt indicates the current command. You can edit commands with the keyboard before issuing them.

Workstation Window

Your graphic is drawn in the workstation window, beneath the SIGHT menu interface.

Plot 2-D Data

This section demonstrates how to use the SIGHT interface to plot a graph from two-dimensional data. The SIGHT Commands chapter will assist you in using SIGHT to create more elaborate graphs.

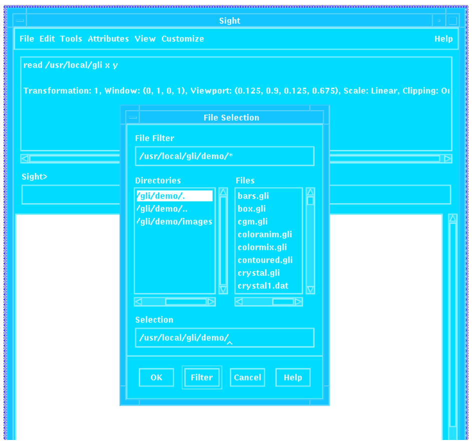
Read Data from a File

To read data into SIGHT:

1. Point at `the` FILE menu item on the SIGHT menu bar. Press and hold the left mouse button to display the FILE pull-down menu.
2. Continue holding down the mouse button and point at the `READ DATA FROM FILE`... command.
3. Release the mouse button to display the File Selections pop-up dialog box. Data files available to SIGHT appear in the File Selections scroll list.
4. Click on the name of your data file (use LOG.DAT for this example) in the scroll list. The file name appears in the File Selection text box.
5. Now click the OK button at the bottom of the File Selection dialog box. The dialog box disappears and SIGHT reads the data file.

Holding down a mouse button while pointing is called pressing. Pressing a mouse button while moving the mouse is called dragging.

FIGURE 2-32 Read Data with SIGHT



```
read /usr/local/gli x y

Transformation: 1, Window: (0, 1, 0, 1), Viewport: (0.125, 0.9, 0.125, 0.675), Scale: Linear, Clipping: On

Sight>
```

File Filter: /usr/local/gli/demo/*

Directories:
```
/gli/demo/.
/gli/demo/..
/gli/demo/images
```

Files:
```
bars.gli
box.gli
cgm.gli
coloranim.gli
colormix.gli
contoured.gli
crystal.gli
crystal1.dat
```

Selection: /usr/local/gli/demo/

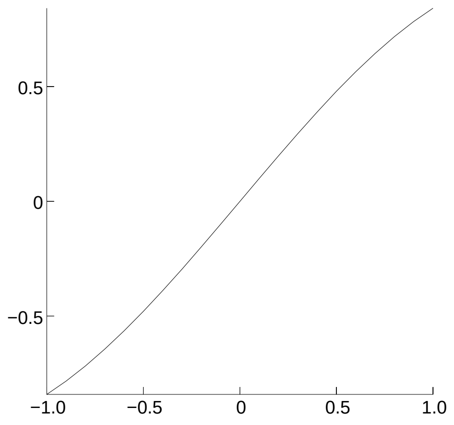
Plot Data

To graph your data:

1. Press the left mouse button on the TOOLS pull-down menu. Drag the mouse down the pull-down menu to `the PLOT menu item and point at the “>” symbol to display the PLOT pull-right` menu.
2. While holding down the mouse button, drag the mouse down to the DATA pull-right menu and release the mouse button.

Your plot will be drawn on the screen.

FIGURE 2-33 Plotting an XY Graph with SIGHT



By default, SIGHT creates graphs using black, solid lines, one point in thickness. These, and other attributes of your plot, can be easily modified as you will see below.

SIGHT Objects

Graphs created by GLI consist of individual objects (e.g., the axes, labels, curve, etc.). These objects can be selected and then copied, deleted or modified using SIGHT. To interact with an object, it must first be selected using the EDIT pull-down menu. Alternatively, an object can be selected by pointing to it and clicking the left mouse button.

An object is deselected when it is not a enclosed by a dashed line. DESELECT is an option listed under the EDIT menu.

Modify a Graph

To change the attributes of the curve on your graph:

1. Point at the EDIT menu item, click and drag to the SELECT pull-right menu.
2. Point the mouse to the OBJECT menu item and release the mouse button.
3. Move the mouse cursor to your graph, and the cross hair pattern will appear indicating SIGHT is in select mode.
4. Select the object to be modified, in this case the curve, by pointing to it and clicking the left mouse button. The curve will be enclosed in a dashed line pattern indicating it has been selected.
5. Now point at the ATTRIBUTES menu item and drag the cursor so the LINE TYPE pull-right > menu is displayed.
6. Point at the DASHED LINE selection and release the mouse button. The curve on your graph is re-drawn as a dashed line.

To reduce the mouse actions required to edit a graph, you can select an object by pointing to it and pressing the left mouse button. This automatically activates the SELECT function.

Change the Axis

To change this graph to include a Logarithmic Y axis:

1. Point at the VIEW menu item, click and drag to the pull-right > SCALE menu.
2. With the mouse button still depressed, point at the Y LOG choice and release the mouse button to modify the axis.

Add Text

To add a title to your graph:

1. Point the mouse on the TOOLS menu item, click and drag to the TEXT selection and release the mouse button.
2. Position the mouse on your graph at the point you want to begin your title and click the left mouse button to display a text marker.
3. Type your title using the keyboard. When you have finished typing, press the RETURN key to add the title to your graph.

Additional text (annotation) can be added to a graph using the above instructions.

Move an Object

To move the title on your graph:

1. Click on the EDIT menu item and drag the mouse to the SELECT pull-right (>) to display the pop-up menu.
2. Point the mouse to the OBJECT menu item and release the mouse button to place SIGHT in the select mode.
3. Position the cross hair on your title and click the left the mouse button. The title will be enclosed in a dashed line to indicate it has been selected.
4. Point at the EDIT menu item, click and move down the list to the MOVE menu item and release the mouse button.
5. Point at your title, click the left mouse button and drag the title to a new location on your graph.
6. Click the left mouse button to re-draw the title at its new location.

Follow steps similar to the above to move or modify or re-position any object.

Exit SIGHT

To exit the SIGHT interface:

1. Pull down the FILE menu item on the SIGHT menu bar.
2. Point at the `QUIT` command and release the mouse button.
3. The interface disappears and SIGHT returns you to the GLI prompt (`gli>`).

Tailoring Commands

GLI commands may be abbreviated. For example:

```
gli> H
gli> HE
gli> HEL
gli> HELP
```

…all invoke the GLI Help facility.

You can substitute your own meaningful symbols for GLI command names or parameters using assignment statements (:= and =) or the DEFINE SYMBOL command.

For instance, the command to clear your workstation display is GKS CLEAR_WS. If you would prefer to enter ERASEALL to invoke this command, type:

```
gli> define symbol eraseall = gks clear_ws
```

Alternatively, you could create a symbol using quotation marks as in:

```
gli> eraseall = “gks clear_ws”
```

You can define and use abbreviated forms of symbols using the wild card character (*). For example:

```
gli> define symbol erase*all = gks clear_ws
```

…allows you to execute the GKS CLEAR_WS command using:

```
gli> erase
```

or:

```
gli> eraseall
```

You can use symbols at places other than the beginning of a command line by using the symbol substitution operator, the apostrophe (’). The following command defines a data file named SPECTRUM.DAT as the symbol FILENAME:

```
gli> filename = ”spectrum.dat”
```

. . . then

```
gli> read ’filename’ x,y
```

…tells GLI that FILENAME is a symbol name and not a literal string. If you had not used apostrophes, the command would have looked for a file called FILENAME.DAT. The READ command always looks for the default .DAT extension unless the file name is otherwise specified.

Help Facility

The GLI Help facility provides on-line documentation for command line instructions. The on-line documentation is arranged in multiple levels, allowing you to optionally specify topics and sub-topics. At each level, the Help facility will list available topics and sub-topics and prompt you for your selection.

For example, to find more information regarding GKS, type the HELP command with GKS as a parameter. For example:

```
gli> help gks
```

If you need help on a particular command, but do not know which command to specify, enter the command HELP HINTS for a list of related command names and system information topics.

If you are a new user of GLI, enter the command HELP INTRODUCTION for an overview of GLI and a list of helpful sub-topics.

If you do not know the name of a particular topic, simply enter the HELP command to list all available topics.

Journal File

Command Recall

During a GLI session, you may enter many commands. To recall and display previously entered commands, press the Up Arrow key on your keyboard. If you want to process that command again, simply press the RETURN key. GLI remembers the last 40 commands.

Journal and Recovery

GLI’s Journal facility records all commands entered during an interactive session in a “journal” file. The journal file is stored in the current directory with the file name `GLI.JOU`. If your GLI session ends because of a system interruption, the journal file is automatically saved. When you re-start GLI, you can use the `RECOVER` command to restore your work.

Macros

Use the `SAVE` command to save commands entered during your GLI session in a file other than `GLI.JOU`. The SAVE command automatically adds a .GLI file extension unless otherwise specified.

```
gli> save plot.gli
```

Use this file as a starting point for creating command procedures (see below).

Command Procedures (Macros)

A Command Procedure is a text file that contains a sequence of command line instructions (See LEARN in the GLI Command Language chapter for more information.) Use

command procedures to catalog frequently used sequences of commands. A command procedure can contain:

- Any valid command line instruction.
- Comment lines.
- Labeled subroutines.
- A request to execute another macro.
- Access to other software on your system.

The default file type for a GLI macro is `.GLI`.

There are three ways of creating command procedure files:

- Use a text editor (e.g., EDT) to create a text file of command statements.
- Use the GLI LEARN and SAVE commands to record command statements issued from the keyboard in a text file.
- Use the $ command to access your operating system and re-name the `GLI.JOU` file as `filename.GLI` and then modify that file with a text editor.

Sample Command Procedure

The following macro text file, `SIMPSURF.GLI`, creates a surface plot from the `TEMP.DAT` file:

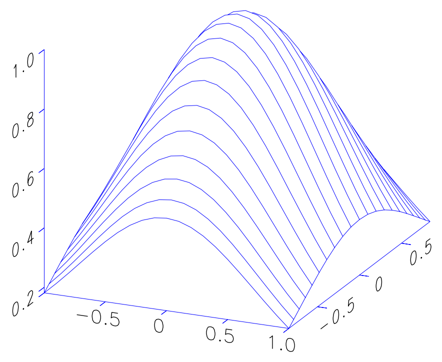
```
!create a simple surface plot
gks set xform wc
gks set viewport 0.15 0.95 0.15 0.95
gks set window 1 100 1 100
read temp.dat x y
f = exp (-x**2-y**2)
gks set pline color_index blue
gks set pline linetype solid
gks set pline linewidth 1.0
gus autoscale_3d x y f
gus axes_3d
gus surface x y f
```

Executing Command Procedures

Execute command procedures by typing an at sign (@) followed by the name of the file containing the commands. For example, to execute `SIMPSURF.GLI`, type:

```
gli> @simpsurf
```

FIGURE 2-34 Plotting a Surface with Macros



Macro Suggestions

Macros turn GLI, GUS, and GKS command statements into a powerful interpretive programming language:

- Boolean Expressions
  When you want your macro to choose a particular course of action depending on the provided information, use the `IF…THEN` command.
- GOSUB Subroutine Calls
  When you want your macro to utilize a subroutine, use the `GOSUB` command. The `RETURN` command terminates the `labeled` subroutine.
- GOTO Statements
  When you want your macro to skip a number of steps depending on given conditions, use the `GOTO` command.

Data Types

GLI data types have been used to create the sample data files graphed by the examples throughout this chapter. A GLI data item includes constants, mathematical expression, a function name, a range, a repetition, a sub-range, or a variable name. The following table lists the GLI data types and their formats.

TABLE 2-8 Data Item Types

Item Type	Example	Description
constant	3.1415	A constant expression, such as the value of pi.
expression	3.1415*R**2	A arithmetic expression.
identifier	cir = 3.14*R**2	A unique name (a combination of letters, digits, dollar signs ($), and underscores (_)) identifying a data item. A data item identifier cannot start with a digit.
range	a(n)b	A defined range of data from initial value (a) to final value (b) by a defined increment (n).
sub-range	a..b	A defined range from the lower-bound (a) to (..) the upper-bound (b), inclusive.

You can use the following techniques (and others) to create or modify a Data Type:

- Read an input file and assign its contents to variables. Use the GLI command READ and specify one or more variables.

  ```
  gli> read temp.dat x y
  ```
- Define a variable or an array to be used whenever a data type is required. Use the DEFINE VARIABLE command and/or the := assignment operator to create storage for data values.

  ```
  gli> define variable r := 2.0
  ```

  Or, the command can be simplified to:

  ```
  gli> r := 2.0
  ```
- Define arithmetic functions that combine variables and/or functions in a problem-specific way. Use the DEFINE FUNCTION command and/or the equals sign (=) assignment statement.

  ```
  gli> define function cir = 3.14*r**2
  ```

  Or, the command can be simplified to:

  ```
  gli> cir = 3.14*r**2
  ```
- Request graphical input from a locator or stroke device. Use the GKS REQUEST command to provide data input, for example, by using the mouse to select a position or range of points.

Using the Assignment Statements

There are two types of assignment statements:

- Equals sign (=) assigns a single value or defines a function. In the example:

  ```
  gli> x = 1(0.1)10
  ```

  GLI will substitute the function 1(0.1)10 in every instance it encounters X.
- Assignment operator (:=) assigns single or multiple values, including a single variable, or an array or range of values. Assignment operators create storage in memory, such that in the example:

  ```
  gli> x := 1(0.1)10
  ```

  GLI creates an array identified as X and assigns values to that array.

Arithmetic Operations

Arithmetic operations provide a formula for calculating values. You may combine numeric data items with operators, expressions, pre-defined functions, and constants. The following tables list arithmetic operators and intrinsic functions.

TABLE 2-9 Operators

Operator	Description	Example
+	Addition	A+B
[ ]	Array Index	A[B]
/	Division	A/B
=	Equals	IF A=B THEN ...
**	Exponential	A**B
>	Greater than	IF A>B THEN ...
<	Less than	IF A<B THEN ...
>=	Greater than or equal to	IF A>=B THEN ...
<=	Less than or equal to	IF A<=B THEN ...
*	Multiplication	A*B
<>	Not equal to	IF A<>B THEN ...
OR	Logical Or	IF A=B OR B=C THEN ...
AND	Logical And	IF A=B AND B=C THEN ...
-	Subtraction	A-B

Parentheses can be used in an expression to force a particular order for combining the operands. For example: 8 * 5 / 2 - 4 yields 16, while 8 * 5 / (2 - 4) yields -20.

TABLE 2-10 Intrinsic Functions

Name	Function
ABS	absolute value
ARCCOS	arc cosine
ARCOSH	hyperbolic arc cosine
ARCSIN	arc sine
ARCTAN	arc tangent
ARSINH	hyperbolic arc sine
ARTANH	hyperbolic arc tangent
COS	cosine
COSH	hyperbolic cosine
DEG	degree
E	2.71828
ERF	error function
ERFC	complementary error function

EXP	exponential function
FRAC	fractional part
GAMMA	gamma function
INT	truncated value
LN	natural logarithm
LOG	common logarithm
MAX	maximum
MEAN	mean
MIN	minimum
PI	3.14159
RAD	radian
RAN	uniformly distributed random numbers
RAND	normally distributed random numbers
SIGN	transfer of sign
SIN	sine
SINH	hyperbolic sine
SIZE	number of elements
SQR	square
SQRT	square root
STDDEV	standard deviation
TAN	tangent
TANH	hyperbolic tangent
TOTAL	sum
TRUNC	truncated value

File Formats Supported by GLI

The GLI READ command (refer to pages GLI Command Language chapter) accepts text files as described below. Future versions of GLI will accept binary data and other file formats.

Text Files Containing Numeric Data Items

If a line contains only numerical information, GLI then reads the file as data. In the following example, three columns of real number data are in a format which could be read into a GLI variable or set of variables:

```
1.04843E-01 1.04843E-01 1.04843E-01
2.12000E+01 3.00001E-03 -2.12038E-01
.
.
.
```

Integers are read into GLI using a format similar to:

```
19 1 18 2 0 0
31 5 16 4 1 1
.
.
.
```

Text Files Containing Text Items

If a file contains only text characters, then each line is considered a text string and is assigned to the symbols P1 through Pn. If the following lines were in a file, they would be read in order as text strings and assigned to P1, P2 and P3.

```
a. This is a text string
b. This is the second text string
c. This is the third text string
```

Text Files Containing Text and Numeric Data Items

If a file contains text and numeric data entries, lines beginning with text characters are read in their entirety as text strings and assigned in order to variables P1 through Pn.

```
1.04843E-01 1.04843E-01 1.04843E-01
This is text string
2.12000E+01 3.00001E-03 -2.12038E-01
This is text string
```

Lines beginning with numeric data and including text characters are read into GLI with all characters assigned to variables P1 through Pn.

```
1.04843E-01 This is text string
2.12000E+01 This is text string 3.00001E-03
```

CHAPTER 3

GLI Command Language

Table of Contents

Table of Contents continued

Overview

This chapter contains descriptions of the GLI Command Language instructions. Each description contains the command’s purpose, syntax, parameters, and an example of how the command is used. Examples containing the GLI system prompt (GLI>) indicate what you would enter at your terminal. Examples containing comment lines (! Comment…) indicate what you would enter in a macro (command procedure) file.

Tailoring Commands

GLI commands may be abbreviated. For example:

```
gli> h
gli> he
gli> hel
gli> help
```

…all invoke the GLI Help facility.

You can substitute your own meaningful symbols for GLI command names or parameters. Use assignment statements (:= and =) or the DEFINE SYMBOL command.

```
gli> define symbol eraseall = ”gks clear_ws”
```

You can define and use abbreviated forms of symbols using the abbreviation character, an asterisk (*). For example:

```
gli> define symbol erase*all = ”gks clear_ws”
```

…allows you to execute the GKS CLEAR_WS command using:

```
gli> erase
gli> eraseall
```

You can use symbols in places other than at the beginning of a command line by using the symbol substitution operator, an apostrophe (’). For example:

```
gli> filename = ”spectrum.dat”
gli> read ’filename’ x,y
```

…tells GLI that FILENAME is a symbol name and not a literal string. If you had not used apostrophes, the command would have looked for a file called FILENAME.DAT.

$(OS)

Allows communication with the operating system to execute system-specific routines and user application programs.

FORMAT

$[command-string]

PARAMETERS

command-string
Specifies an operating system command string of 132 characters or less.

DESCRIPTION

$(OS) creates a sub-process of your current GLI process.

When the command completes execution, the subprocess terminates and control returns to the GLI command language.

EXAMPLES

For example, to obtain a listing of all files ending in .GLI:

```
gli> $dir *.gli

Directory USR$USRDISK:[SMITH]
SPECTRUM.GLI;1        2          9-SEP-1991 12:10
Total of 1 file, 2 blocks.
gli>
```

@ (Execute Procedure)

Executes a GLI macro.

FORMAT

@file-name [p1, p2, p3 ..., p8]

PARAMETERS

file-name
Specifies the macro to be executed. If a file type is not specified, the system uses the default file type of `.GLI`. No wild card characters (* or %) are allowed in the file specification.

[p1 ..., p8]
Specifies from one to eight optional parameters to pass to the macro. The parameters assign character string values to the symbols named P1, P2, and so on in the order of entry, to a maximum of eight (P8). The symbols are local to the specified macro.

DESCRIPTION

@ executes a macro or file containing one or more GLI commands.

Use macros to catalog frequently used sequences of commands. A macro can contain:

- Any valid GLI command;
- Comment lines;
- Labeled subroutines;
- Request to execute another macro.

To execute a macro, type an at sign (@) followed by the name of the file containing the macro and any related parameters. If the macro is in a separate directory, provide the absolute path to the macro.

EXAMPLE

The following example executes the macro MONTH.GLI. This macro accepts the name of a data file (TEMP.DAT) as the parameter P1.Variables are automatically deleted when a GLI session is terminated. Variables can be defined in a GLISTARTUP.GLI file to provide consistent access to regularly used variables. (See Start-Up Command files in Appendix A.)For instance, the following procedure could be used to automatically plot monthly data:

```
gli> @plot Jan.dat, January
21 lines read from file temp.dat
gli>
```

APPEND

Appends one or more data items to an output file.

FORMAT

APPEND file-name data-spec

PARAMETERS

file-name
Specifies the name of the output file to be written. If you do not specify a file type, the system uses the default file type of `.DAT`. No wild card characters (* or %) are allowed in the file specification.

data-spec
Data specification. A data specification may be:

- An arithmetic expression;
- A constant;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

The APPEND command can append data to sequential files.

EXAMPLE

```
gli> append data.dat x,y
21 records written to file data.dat
gli>
```

CALCULATE

Displays the result of an arithmetic expression.

FORMAT

CALCULATE expression

PARAMETERS

expression
Any valid arithmetic expression.

DESCRIPTION

CALCULATE determines and displays the result of an arithmetic expression. An expression may represent the value of a constant, a variable, or a function designator, or it may represent a combination of values.

You can use any of the mathematical intrinsic functions supplied with GLI. (See Math Functions in this chapter.)

EXAMPLES

In this example, the variable X is first assigned a value, and then it is used in the arithmetic expression.

```
gli> x := 2.719
gli> calc gamma(2*x-x**2)
1.16067e+01
gli>
```

CASE…THEN

Tests the value of the specified symbol in a macro with a provided string. If the strings match, the command executes the instructions in the THEN clause.

FORMAT

CASE symbol-name choice THEN command

PARAMETERS

symbol-name
Defines a 1 to 31 character alphanumeric string name for the symbol. The symbol name must begin with an alphabetic character.

choice
Specifies a character string value to be compared with the symbol. A character string can have from 0 to 255 characters.

command
Any GLI command.

DESCRIPTION

CASE compares the string defined in SYMBOL-NAME to the string defined in CHOICE and executes a given command if the strings match. A valid command can include a subroutine call, a GOTO statement, an @ command or a GLI command.

EXAMPLE

Macros can be created using your favorite text editor. In this example, the CASE command tests the value of the parameter P1 and executes a macro named PLOT if the value of P1 equals PLOT. If the value of P1 equals CONTOUR, then a macro named CONTOUR is executed. If the value of P1 equals SURFACE, then a macro named SURFACE is executed.

```
! test and execute selections
case ’p1’ plot then @plot
case ’p1’ contour then @contour
case ’p1’ surface then @surface
.
.
.
exit
```

COMMAND PROCEDURE

A user-created file (macro) which contains and executes a series of GLI instructions.

FORMAT

@file-name [p1, p2, p3 . . ., p8]

PARAMETERS

@
Executes the command procedure.

file-name
Specifies the macro to be executed. If a file extension is not specified, the system appends .GLI to the file name. Wild card characters (* or %) are not allowed in the file name.

[p1, p2, p3, . . . p8]
Specifies from one to eight optional parameters to pass to the macro. The parameters assign character string values to the symbols named P1, P2 and so on, in the order of entry, to a maximum of eight. The symbols are local to the named macro.

DESCRIPTION

Use a command procedure to store and execute a series of commands. To create command procedures at the GLI prompt (gli>), use the LEARN and SAVE commands. To create them outside GLI, use a text editor (VI, EDT, etc.). The SAVE command appends .GLI as the default file extension to the macro name.

Command Procedures can be created in two separate ways:

- Using the LEARN and SAVE commands at the GLI prompt. (See LEARN for more information). Macros created using LEARN and SAVE append a .GLI extension to the file name unless otherwise specified by the user.
- Using a text editor such as VI or EDT while outside the GLI system. Since GLI looks for files with a .GLI extension, either save macros using this default extension or give the complete file name to execute the macro.

DEFINE FUNCTION

Associates an arithmetic expression with the specified function.

FORMAT

DEFINE FUNCTION function-name = expression *or* ***function-name = expression***

PARAMETERS

function-name
Defines a 1 to 31 character alphanumeric string name for the function. The function name must begin with an alphabetic character.

expression
Any valid function expression.

DESCRIPTION

DEFINE FUNCTION creates an arithmetic function for use in GLI. The function is computed each time it is referenced, and the resulting value is then made available to the expression that contains the function reference.

A list of all currently defined functions may be generated using the SHOW FUNCTION command. To delete a function use the DELETE FUNCTION command.

Functions are automatically deleted when a GLI session is terminated. Functions can be defined in a GLISTARTUP.GLI file to provide consistent access to regularly used functions. (See Start-Up Command files in Appendix A.)

EXAMPLES

In this example two functions are created named F and G. Function F is defined as the arctangent of the variable X plus one. Function G is more complex and includes the function F as input to G.

```
gli> define function f = arctan(x)+1
gli> define function g = (sqrt(1-sin(x/pi)**2)-ln(1))*2+ f
gli>
```

DEFINE LOGICAL

Defines any system logical used within the GLI environment.

FORMAT

DEFINE LOGICAL name value

PARAMETERS

name

any one of the following

TABLE 3-1 Valid Logical Names

Name	Use
GLI_WSTYPE	Default GLI workstation type
GLI_CONID	The GLI connection identifier
GLI_FIG	The name of the default figure file
GLI_GKS_CMAP_EXTENT	The GLI/GKS colormap extent
GLI_GKS_SCALE_MODE_METRIC	Enables CGM metric scale mode
GLI_GKSM	The name ofthe GKSM output file
GLI_GKS_MAGSTEP	The GLI/GKS magnification factor ($1.2^{magstep}$)
GLI_GKS_TRANSPARENT_PATTERNS	Enables GLI/GKS transparent patterns
GLI_GKS_PATTERN	The name of the pattern file
GLI_GIF	The name of the GIF file
GLI_POINTS	Specifies the maximum number of output points
GLI_GKS_DOUBLE_BUF	Enables GLI/GKS double buffering
GLI_GKS_CONVEX_SHAPE	Assumes convex shapes for filled polygons
GLI_GKS_XSHM	Enables X Shared Memory extensions
GLI_RF	The name of the Sun rle rasterfile
GLI_CGM	The GLI default CGM file name
GLI_LPR	The default print command
GLI_SERVICE	The GLI RPC service number
GLI_UIL	The name of the UIL file (User Interface Language)
GLI_ICON	The name for the ICON definition (UIL)

value

the definition of the logical

DESCRIPTION

DEFINE LOGICAL is used to define logical values that are used internally within GLI. GLI uses logicals to define system variables that are used for a variety of functions within GLI.

The use of quotes (“ “) is mandatory around items with a colon (:), or else the GLI parser will interpret the unquoted word as two items, possibly leading to errors in your application.

EXAMPLES

Use define logical to define GLI_SERVICE for use with a RPC.

```
gli> define logical GLI_SERVICE “sender:1”
```

Define logicals to generate a Motif .uil bitmap file..

```
gli> define logical GLI_UIL icons.uil
gli> gks open_ws terminal 212
.
.
.
gli> define logical GLI_ICON demo_icon
```

DEFINE SYMBOL

Assigns a symbol string to the specified symbol.

FORMAT

DEFINE SYMBOL symbol-name = expression or symbol-name = expression
or
DEFINE SYMBOL symbol-name := expression *or* ***symbol-name := expression***
or

symbol-name *= “expression”*
or

symbol-name *:= “expression”*

PARAMETERS

symbol-name
Defines an alphanumeric string name for the symbol. The symbol name must begin with an alpha character and can contain 1 to 31 alphanumeric characters. Symbols can be abbreviated using the wild card character (*) in the symbol name.

expression
Specifies a character string value to be equated to the symbol. A character string can consist of 0 to 255 alphanumeric and special characters. If the DEFINE SYMBOL command is not used, the expression must be enclosed in quotation marks.

DESCRIPTION

DEFINE SYMBOL (or the assignment operator and quotation marks) creates an entry in a symbol table by defining a symbolic name to stand for an equivalence name. GLI translates the symbol to the equivalent name each time the symbol is used.

If a symbol defines a text string, it must be enclosed in single quotation marks to indicate the text string should be substituted. If the text string is equivalent to a valid GLI command, the command will be issued.

The SHOW SYMBOL command displays a list of all currently defined symbols. To delete a symbol use the DELETE SYMBOL command.

Symbols are automatically deleted when a GLI session is terminated. Symbols can be defined in the GLISTARTUP.GLI file to provide consistent access to regularly used symbols. (See Start-Up Command files in Appendix A.)

EXAMPLES

The following defines the symbol name DATA equivalent to the filename TEMP.DAT. The next command reads the file TEMP.DAT defined as DATA containing variables X and Y.

```
gli> define symbol data = temp.dat
gli> read ’data’ x,y
```

DEFINE SYMBOL continued

This example defines the symbol name ENTER to invoke the GLI command DEFINE VARIABLE. A wild card character (*) is used below to abbreviate ENTER so that the commands EN and ENTER can be used interchangeably.

```
gli> define symbol en*ter = define variable
```

DEFINE VARIABLE

Creates a variable and assigns one or more values to it.

FORMAT

DEFINE VARIABLE variable-name = data
or

variable-name := data

PARAMETERS

variable-name
Defines a 1 to 31 character alphanumeric string name for the variable. The variable name must begin with an alphabetic character.

data
One or more data-items separated by comma(s) or blank(s).

DESCRIPTION

DEFINE VARIABLE creates an entry in a dynamic variable table by defining a symbolic name to stand for one or more data values. The assignment operator (:=) also defines a variable and requires only the parameters VARIABLE-NAME and DATA for a valid command.

A list of all currently defined variables can be displayed using the SHOW VARIABLE command. To delete a variable use the DELETE VARIABLE command.

Variables are automatically deleted when a GLI session is terminated. Variables can be defined in the GLISTARTUP.GLI file to provide consistent access to regularly used variables. (See Start-Up Command files in Appendix A.)

EXAMPLES

This command creates a variable named FREQUENCY and is assigned four values. Note the values can be separated by either commas or spaces.

```
gli> define variable frequency = 1.2,3.456 12.34,1.2e+1
```

In this instance, the variable time is created and assigned two ranges of data. The first range of data 0(.1)10 includes the 101 values between 0 and 10 in increments of .1. The second range of data specifies the values between 20 and 100, endpoints included, in increments of 1.

```
gli> define variable time = 0(0.1)10, 20..100
```

Using the assignment operator (:=), variables can be defined without an explicit DEFINE VARIABLE command.

```
gli> subrange := 10..100
```

This statement creates an array of data named X which is assigned values containing two ranges of data plus the values from the previously defined variable SUBRANGE.

```
gli> x = 0.1(0.01)1 1(0.1)10 subrange
```

DELETE FUNCTION

Deletes one or more function table entries.

FORMAT

DELETE FUNCTION function-name

PARAMETERS

function-name
Specifies one or more function names which are to be removed from the function table. The function name can have from 1 to 31 alphanumeric characters. The function name must begin with an alphabetic character. The function name specification may contain wild card characters (* or %).

DESCRIPTION

The DELETE FUNCTION command deletes one or more user-defined entries from the function table. Functions can be deleted by naming the function variable specifically, or multiple functions can be deleted by using the GLI wild card symbol (*). This command returns a message indicating the number of functions deleted.

If the DELETE FUNCTION command is used when the function(s) is not defined, the following informative message will appear:

```
%FUNCTION-W-NOFUNCTION, no functions found
```

EXAMPLES

To delete a single user-defined function:

```
gli> delete function d
1 function deleted
```

To delete multiple functions assigned to f1, f2 and f3:

```
gli> delete function f*
3 functions deleted
```

To delete all user-defined functions:

```
gli> delete function *
3 functions deleted
```

DELETE SYMBOL

Removes one or more symbols from the GLI symbol table.

FORMAT

DELETE SYMBOL symbol-name

PARAMETERS

symbol-name
Specifies one or more symbol names which are to be deleted. The symbol name can have from 1 to 31 alphanumeric characters. The symbol name must begin with an alphabetic character. The symbol name specification may contain wild card characters (* or %).

DESCRIPTION

DELETE SYMBOL deletes one or more entries from the symbol table. Symbols can be deleted by naming the symbol variable specifically, or multiple symbols can be deleted by using the GLI wild card character (*). The DELETE SYMBOL command returns a message indicating the number of symbols deleted.

If the DELETE SYMBOL command is used when a symbol(s) is not defined, the following informative message will appear:

```
%SYMBOL-W-NOSYMBOL, no symbols found
```

EXAMPLES

To delete a single user-defined symbol named PAGE:

```
gli> delete symbol PAGE
1 symbol deleted
```

To delete all user-defined symbols:

```
gli> delete symbol *
3 symbols deleted
```

DELETE VARIABLE

Deletes one or more dynamic variables.

FORMAT

DELETE VARIABLE variable-name

PARAMETERS

variable-name
Specifies one or more variable names which are to be deleted. The variable name can have from 1 to 31 alphanumeric characters. The variable name must begin with an alphabetic character. The variable name specification may contain wild card characters (* or %).

DESCRIPTION

DELETE VARIABLE deletes one or more variables from GLI.Variables can be deleted by naming the variable specifically, or multiple variables can be deleted by using the GLI wild card character (*). The DELETE VARIABLE command returns a message indicating the number of variables deleted.

If the DELETE VARIABLE command is used when variable(s) are not defined, the following informative message will appear:

```
%VARIABEL-W-NOVARIABLE, no variable found
```

EXAMPLES

To delete a single user-defined variable named X:

```
gli> delete variable x
1 variable deleted
```

To delete all user-defined variables:

```
gli> delete variable *
3 variables deleted
```

DISPLAY

Displays on the device a menu or sub-menu loaded by the LOAD command.

FORMAT

DISPLAY [key]

PARAMETERS

key
Option specifying the index of the menu to be displayed.

DESCRIPTION

DISPLAY requests the Form Driver to display a loaded menu on the screen. The Form Driver is an interactive utility that GLI uses to access user-created menu templates. When the form is displayed, the Form Driver does the following:

1. Establishes a connection with the user’s terminal.
2. Displays the form and accepts user input.
3. Refers to data specifications contained in the form to check that user input is valid.
4. Responds to user requests (events).
5. Displays or erases data in the form.

Any data contained in an active form may be accessed using the symbols P1 to P*n*.

The Form Driver chapter describes the Form Driver and its calls in detail.

EXAMPLE

In this example, the DISPLAY command is used with the LOAD command in a macro to load and display a user-created menu.

```
! load menu interface
load ’gli_demo’demo.fdv
.
.
.
on f1 then return do
on f4 then return exit
.
.
.
display
.
.
.
exit
```

DO

Executes more than one GLI command defined in a symbol.

FORMAT

DO command1; command2; command3; ...

PARAMETERS

command
Any valid GLI command string or strings. Multiple strings are separated by semicolons.

DESCRIPTION

DO executes in sequence multiple commands separated by a semicolon as defined in a symbol. The GLI command language can be customized using DO to equate a series of instructions as one symbolic name.

Either the DEFINE SYMBOL command or quotations marks define strings of commands as one symbol. GLI only recognizes straight or right facing quotation marks (i.e. " or ”).

EXAMPLE

The DEFINE SYMBOL statement indicates DO will create a symbol consisting of the valid GLI instructions.

In this example, the symbolic name GRAPH is set to equal two GKS commands. These commands draw a polyline and polymarkers at the X and Y coordinates. Henceforth, whenever the GRAPH symbol is invoked, a polyline and polymarkers are plotted at the current X and Y coordinates.

```
gli> define symbol graph = do gks polyline x, y; gks polymarker x, y
```

Or, the same symbol could be created without the DEFINE SYMBOL but using quotation marks:

```
gli> graph = “do gks polyline x, y; gks polymarker x, y”
```

EXIT

Terminates execution of a macro or quits the GLI program.

FORMAT

EXIT [file-name]

PARAMETERS

file-name
Specifies the journal file name (GLI.JOU by default). Wild card characters (* or %) are not allowed in the file specification.

DESCRIPTION

When used on the GLI command line (gli>), EXIT terminates your GLI session and preforms the following:

- Saves all GLI commands issued during that session to the journal file GLI.JOU, unless otherwise specified, and then closed.
- Deletes all symbols, variables and functions.
- Closes the Graphical Kernel System (GKS).

Use the QUIT command to terminate a GLI session without creating a journal file.

When used as part of a macro, EXIT returns control to the GLI prompt (gli>).

EXAMPLE 1

Use EXIT in a macro to escape to GLI:

```
! sample macro
.
.
.
exit
gli>
```

EXAMPLE 2

Use EXIT on the command line to leave GLI and return to the operating system.

```
gli> exit
$
```

GKS

Invokes the Graphical Kernel System command language interpreter which controls GKS instructions.

FORMAT

GKS parameter

PARAMETERS

parameter
Any valid GKS command(s). Refer to the GKS chapter for a complete list of GKS commands.

DESCRIPTION

GKS invokes the GKS interpreter and issues a GKS command. GKS instructions are entered at the GLI prompt and executed when the RETURN key is pressed.

See the GKS Command Language chapter for more information.

EXAMPLE

This example would draw a polyline from variables X and Y.

```
gli> gks polyline x, y
```

GOSUB

Transfers control to a labeled subroutine. The RETURN command terminates the GOSUB subroutine.

FORMAT

GOSUB label

PARAMETERS

label
A valid label contains only 1 to 31 alphanumeric characters without blanks and terminated by a colon) in the macro.

DESCRIPTION

Use the GOSUB command in a macro to transfer control to a local subroutine specified by the label. The RETURN command terminates the subroutine procedure, returning control to the command following the calling GOSUB statement.

When creating a label for a subroutine within a macro, the label must be the first item on a line, and it must be terminated by a colon (:). Labels cannot contain embedded blanks.

EXAMPLE

In this example macro, the GOSUB command transfers control to the subroutine labeled AXES. After drawing the axes, the subroutine is terminated by the RETURN command. Control is then returned to the command following the calling GOSUB statement (the GKS POLYLINE X, Y command).

```
! create a simple plot
read data.dat x, y
gosub axes
gks polyline x, y
exit
! subroutine axes plots a pair of axes
axes:
gus autoscale x, y
gus axes_2d
return
```

GOTO

Transfers control to a labeled statement in a macro.

FORMAT

GOTO label

PARAMETERS

label
A valid label (1 to 31 alphanumeric characters, terminated by a colon) in the macro. A label may not contain embedded blanks and must be the first item on the line. When the GOTO command is executed, control passes to the command following the specified label.

DESCRIPTION

GOTO transfers control to a labeled statement in the macro.

EXAMPLE

In this example, the CASE command checks the parameter P1 when it is passed to the macro. If the string is PLOT, the GOTO command is executed and control is passed to the line labeled PLOT. Otherwise, the procedure continues executing until the EXIT command is encountered.

Note: The following is contained in a macro:

```
! Test symbols
case p1 plot then goto plot
.
.
.
exit
.
.
.
! if p1 equals plot, then plot
plot:
gus plot x, y
exit
```

GR SOFTWARE COMMANDS

FORMAT

GR command

PARAMETERS

command

Any valid GR-software command.

TABLE 3-2 GR-software command keywords

Name	Description
ARRW	Draw an arrow
AXS	Draw a pair of linearly spaced axes
AXSL	Draw a pair of logarithmically spaced axes
AXLIN	Linear axis
AXLOG	Logarithmic axis
BLD	Draw a chart
CHN<C>	Draw a polyline with symbols
CHRC	Set text height and angle
CLP	Set clipping indicator
CRCL	Draw an arc
DCUR	Get graphics cursor
DEL	Clear picture
DN	Set metric scale DIN format
DRAX	Three-dimensional axes
DRDM	Draw a surface top view
DRDU	Draw a surface bottom view
DRHS	Three-dimensional histogram
DRNE	Draw a three-dimensional grid
DRKU	Three-dimensional curve
DRLG	Annotation for three-dimensional axes
DRW	Draw a line
DRWS	Draw a line with a symbol
DSH	Set line pattern
END/ENDE	Close GR-Software
FILL	Draw a filled polygon
FONT	Set text font
FRBN	Set colors
GFLD	Draw gradient field

GR-COMMANDS continued

HHFL	Draw colored iso-surfaces
HHNL	Draw iso-surfaces
JMP	Set position
JMPS	Draw s symbol
LGND	Draw a legend
LN<CN>	Draw a polyline
MRKS	Set marker size
MSKN	Begin picture mask
MSKF	End picture mask
NWSG	Create a segment
NWPN	Set color index
NXTF	Display picture and clear
PTS	Draw symbols at given points
SCAX	Annotation for three-dimensional axes
SCDL	Draw a line with text
SCLC	Set viewport limits
SCLP	Set metric scale
SCLV	Set window limits
SHD	Filllpolygon
SHOW	Display picture
SPHR	Draw spheres
SPTS	Set linewidth
STRT	Open GR-Software
TXT	Draw a text
TXTC	Continuate a text
VAR	Draw error bars
VFLD	Draw streamlines
WIN C	Select output window
00DG	Set landscape orientation
90DG	Set portrait orientation

GRIDIT

Constructs a regular grid from irregularly distributed points given by x, y, and z data.

FORMAT

GRIDIT x-data, y-data, z-data [resolution_x[resolution_y]]

PARAMETERS

x-data, y-data, z-data
Specifies variable names for the irregularly-distributed points to be gridded.

resolution_x, resolution_y
Optional parameters specifying the size of the grid to which the data is to be fitted. The default value for both parameters is 40, describing a 40 by 40 grid.

DESCRIPTION

GRIDIT accepts three arrays (X, Y and Z) of randomly-spaced data and produces a virtual grid of interpolated values which replaces the original data. The gridded data can then be used as input values to the GUS CONTOUR and GUS SURFACE routines.

The GRIDIT algorithm has been taken from the Association for Computing Machinery library (with modification). If the number of data points to be gridded is less than or equal to 100, cubic interpolation is used, otherwise GRIDIT uses linear interpolation.

Common uses for GRIDIT include gridding data collected by acquisition systems, experiments or tests where the data is not uniformly distributed.

EXAMPLE

This example creates a contour plot from irregularly spaced data.

Set text alignment, define the display transformation for the title, position and draw the title, and set the display transformation for the graph:

```
gli> gks set text align center base
gli> gks set xform ndc
gli> gus text .5 .96 Demonstration Plot For Random Contours
gli> gks set xform wc
```

Enter random data for the variables X, Y and Z and define height (H) for contour levels:

```
gli> x := 3 3 10 18 18 10 10 5 1 15 20 5 15 10 7 13 16
gli> y := 3 18 18 3 18 10 1 5 10 15 15 15 20 20 8
gli> h := -5(2)25
gli> z := 25 25 25 25 25 25 -5 1 1 1 1 1 1 1 1 1 25
gli> gus set space -28 35 0 90
```

Grid the data using GRIDIT, scale the window, and draw the contours and axes:

```
gli> gridit x y z
gli> gus autoscale_3d x y z
gli> gus set space 0 1 0 90
gli> gus contour x y h z
gli> gus axes_2d
```

GUS

Invokes the Graphics Utility System command language interpreter (GUS).

FORMAT

GUS parameter

PARAMETERS

parameter
Any valid GUS command parameter.

DESCRIPTION

GUS commands are entered at the GLI prompt (gli>) and invoke the GUS interpreter providing access to specialized graphical routines.

See the GUS Command Language chapter for more information.

EXAMPLE

This example creates a complete line graph using a single GUS command.

X and Y values are read into GLI from a file named DATA.DAT.

```
gli> read data.dat x, y
21 lines read from file data.dat
```

GUS PLOT graphs the X and Y values and invokes AUTOSCALE _2D and AXES_2D to scale the data to the window and draw the axes, respectively.

```
gli> gus plot x, y
```

HELP

Invokes the GLI Help Facility to display information about a GLI command or topic.

FORMAT

HELP [topic [sub-topic]]

PARAMETERS

topic [sub-topic]
Specifies one or more keywords that refer to the topic or sub-topic on which you want information. To use the Help facility in its simplest form, enter the HELP command without any parameter.

DESCRIPTION

HELP returns information on topics or sub-topics stored in the Help libraries. Information within the Help libraries is arranged in a hierarchical manner. If you do not specify a keyword, the Help facility displays a list of topics and prompts you for input.

In response to the prompt, you can:

- Type INTRODUCTION if you are unfamiliar with the GLI system;
- Type HINTS if you are not sure of the name of the command or topic for which you need help;
- Type a question mark (?) to re-display the most recently requested text;
- Type the name of the command or topic;
- Press the RETURN key one or more times to exit from the Help facility.

You can abbreviate any topic name, although ambiguous abbreviations result in all matches being displayed.

Press the RETURN key again to exit the facility and return to the GLI prompt.

EXAMPLE

In this example, the HELP command is entered without any parameters. This produces a display of the HELP topics available.

```
  gli> help
```

TABLE 3-3 Help Keywords

$	**@**	**Append**	**Calculate**	**Case**	**Define**	**Delete**
Display	**Do**	**Exit**	**GKS**	**Gosub**	**Goto**	**GRsoft**
Gridit	**GUS**	**Help**	**If**	**IMAGE**	**Import**	**Initialize**
Inquire	**Learn**	**Load**	**Message**	**On**	**Pipe**	**Print**
Quit	**Read**	**Recover**	**Redim**	**Return**	**RPC**	**Save**
Set	**Show**	**SIGHT**	**SimplePlot**	**Sleep**	**Smooth**	**Write**
XUI						

IF…THEN

Tests the value of the specified expression in a macro and executes the commands in the THEN clause if the expression is true.

FORMAT

IF condition THEN command

PARAMETERS

condition
Boolean expression. Expressions in IF commands are automatically evaluated during execution of the command. The command interpreter does not execute an IF command when it contains an undefined symbol. Instead, the command interpreter issues a warning message and executes the next command in the procedure.

command
The GLI command to be executed, when the result of the expression is true.

DESCRIPTION

IF tests the value of an expression and executes a given command if the result of the expression is true.

EXAMPLE

This example demonstrates how to establish a loop in a macro, using a variable named COUNT, an IF statement, and a labeled GOTO statement. The IF statement checks the value of COUNT and performs an EXIT command when the value of COUNT is greater than 10. Otherwise, the macro repeats the commands or routines within the loop.

```
! set count to 0
count := 0
.
.
.
! start counting loop
loop:
count := count+1
.
.
.
! if count is less than 10, repeat loop
if count <= 10 then goto loop
exit
```

IMPORT

Imports data files in *cameca* format into GLI.

FORMAT

IMPORT filename

PARAMETERS

filename
Name of the file.

DESCRIPTION

IMPORT reads into GLI files in cameca format, which are used in some chemistry applications.

The file is imported when the RETURN is pressed.

EXAMPLE

```
gli> import cameca.dat a b
21 lines read from file newfile.dat
gli>
```

INITIALIZE

Initializes the GLI system and invokes the STARTUP.GLI command file.

FORMAT

INITIALIZE

PARAMETERS

None

DESCRIPTION

INITIALIZE performs the following actions:

- Deletes all symbols, variables and functions;
- Closes the Graphical Kernel System (GKS);
- Invokes any user-created GLI system start-up procedure.

The INITIALIZE command resets GLI to the state specified by the STARTUP.GLI command file.

EXAMPLE

Below, the abbreviated INITIALIZE command resets GLI to its default settings.

```
gli> init
gli>
```

INQUIRE

Requests interactive assignment of a value for a symbol during the execution of a command procedure (macro).

FORMAT

INQUIRE symbol-name [prompt-string]

PARAMETERS

symbol-name
Specifies a 1 to 31 character alphanumeric symbol to be assigned a value.

prompt-string
Specifies the prompt to be displayed when the INQUIRE command is executed. If a prompt-string is not specified, GLI uses the symbol-name as the prompt.

DESCRIPTION

INQUIRE is used to dynamically assign values to symbols or to create symbols in GLI. If a symbol has already been defined, INQUIRE will re-define that symbol. If a symbol has not been created, INQUIRE will create it and assign a value to that symbol.

Prompt-string allows an informative message to be displayed for the user requesting input of a file name, a string or a GLI command. If the user hits RETURN without typing in a value, no assignment is made and an empty symbol is created.

INQUIRE would best be used in command procedures to prompt users for information.

RESTRICTION

A symbol must be given a value on definition since GLI does not allow empty symbols to be defined.

EXAMPLE

In this example, the symbol *file* is defined and given the value *empty.* The INQUIRE command causes *filename?* to be displayed as a reminder to the user to input the name of the file. The user's entry will be assigned to the symbol *file*, which the READ commands then uses as the name of the file to be read.

```
gli> define symbol file = empty
gli> inquire file filename?
gli> read ‘file’ x y
.
.
.
```

LEARN

Collects subsequent commands for a SAVE operation thereby creating a macro.

FORMAT

LEARN

PARAMETERS

None.

DESCRIPTION

Use the LEARN command to store input in a macro. To create a macro:

- Invoke the LEARN command;
- Enter the instructions to be executed in the macro; and
- Store all collected instruction input using the SAVE command.

LEARN executes commands as it records them. If LEARN encounters an invalid command or an undefined symbol, function or variable, an error message will appear and the invalid command will not be included in the file.

EXAMPLE

This example uses LEARN to create a macro from instructions entered on the GLI command line.

Invoke LEARN and read X and Y values from a file named TEMP.DAT:

```
gli> learn
gli> read temp.dat x, y
21 lines read from file temp.dat
```

Graph X and Y and save the instructions as a macro named LINEGRAPH:

```
gli> gus plot x, y
gli> save linegraph
```

To execute the series of instructions contained in the macro LINEGRAPH, enter:

```
gli> @linegraph
```

GLI automatically saves macros with a file name extension of .GLI. If you add a file extension other than .GLI, remember to enter the complete file name following the @ instruction to execute the appropriate macro.

LOAD

Loads a Form Driver menu description from the specified file and establishes a communication environment for subsequent DISPLAY commands.

FORMAT

LOAD file-name [display]

PARAMETERS

file-name
Specifies the name of the menu description file to be loaded. If you do not specify a file type, the system uses the default file type of `.FDV`. No wild card characters (* or %) are allowed in the file specification.

display
Specifies the name of the terminal which should be used for input and output operations. The default is TT.

DESCRIPTION

LOAD is used with Form Driver routines in a macro to load a Form Driver menu description into an internal work-space. Use the DISPLAY command to display the loaded menu.

The Form Driver is an interactive utility that GLI uses to access user-created menu templates.

EXAMPLE

In this example, the LOAD command is used with the DISPLAY command in a macro to load and display a user-created form named MENU.

```
! load menu interface
load 'gli_demo'demo.fdv
.
.
.
on f1 then return do
on f4 then return exit
.
.
.
display
.
.
.
exit
```

MATH FUNCTIONS (Standard)

Provides functions for calculating values. Tables 3-4 through 3-5 describe the mathematical functions available.

FORMAT

Standard Functions
+, -, /, *, **, abs(), arcos(), arcosh(), arcsin(), arctan(), arsinh(), artanh(), cos(), cosh(), deg(), erf(), erfc(), exp(), frac(), gamma(), int(), ln(), log(), max(), mean(), min(), rad(), ran(), rand(), sign(), sin(), sinh(), size(), sqr(), sqrt(), stddev(), tan(), tanh(), total(), trunc()

Predefined constants
pi, e

Parenthesis
(and)

Special operators
a..b, a(b)c

PARAMETERS

Special. See below

DESCRIPTION

MATH FUNCTIONS provide a convenient method to calculate values. Numeric data items can be combined with operators, expressions, pre-defined functions, and constants.

Usage Of Parenthesis
Parentheses can be included in an expression to force a particular order for combining the operands. For example: 8 * 5 / 2 - 4 yields 16, while 8 * 5 / (2 - 4) yields -20.

Usage Of Special Operators
Special operators are used to “count” in slightly different ways. These functions give the user a flexible means to generate data for iterative operations within GLI.

The statement X = 1..100 creates a function X which counts from 1 to 100.

The function X = 1(.1)10 counts from 1 to 10 by adding 0.1 to 1, saving the result (1.1). When the next iteration of the function occurs, 0.1 is added to the saved result and the new value is then stored. The process is repeated onward until 10 is reached one hundred values later. Negative iterations are created as in -1(-.1)-5.

RESTRICTION

All trigonometric functions expect input in radians, not degrees. RAD() converts degrees to radians.

MATH FUNCTIONS continued

TABLE 3-4 Arithmetic Operators

Operator	**Description**	**Example**
+	Addition	A+B
[ ]	Array Index	A[B]
/	Division	A/B
=	Equals	IF A=B THEN ...
**	Exponential	A**B
>	Greater than	IF A>B THEN ...
<	Less than	IF A<B THEN ...
>=	Greater than or equal to	IF A>=B THEN ...
<=	Less than or equal to	IF A<=B THEN ...
*	Multiplication	A*B
<>	Not equal to	IF A<>B THEN ...
OR	Logical Or	IF A=B OR B=C THEN ...
AND	Logical And	IF A=B AND B=C THEN ...
-	Subtraction	A-B

MATH FUNCTIONS continued

TABLE 3-5 Intrinsic Functions

Name	**Function**
ABS()	The absolute value of a number.
ARCOS()	The arccosine of a value.
ARCOSH()	The hyperbolic arccosine of a value.
ARCSIN()	The arcsine of a value.
ARCTAN()	The arctangent of a value.
ARSINH()	The hyperbolic arcsine of a value.
ARTANH()	The hyperbolic arctangent of a value.
COS()	The cosine of a value.
COSH()	The hyperbolic cosine of a value.
DEG()	Changes radians to degrees.
E	2.71828 (Constant)
ERF()	The error function of X where ERF(X)= $2/\sqrt{\Pi}\int_0^x e^{-t2}\,dt$
ERFC()	The complement of the error function ERFC(X) = 1 - ERF(X).
EXP()	Raises E to the power of X where X is a number or equation defined inside the parenthesis.
FRAC()	Returns the fractional portion of a floating point value.
GAMMA()	The gamma function.
INT()	Returns the integer portion of a floating point number.
LN()	The natural logarithm function.
LOG()	Logarithm to the base 10.
MAX()	Maximum value of an array.
MEAN()	The mean of an array.
MIN()	Minimum value in an array.
PI	3.14159 (Constant)
RAD()	Converts degrees to radians.
RAN()	Returns uniformly distributed random numbers in the range of [0,1]. The argument for the function is the seed value for the function. Zero (0) is not allowed as a seed value.
RAND()	Returns normally distributed random numbers with a mean of 0 and a standard deviation of 1. The argument for the function is the seed value for the function. Zero (0) is not allowed as a seed value.
SIGN()	Returns the sign of a number if you have the call SIGN(X) and X = 0 then SIGN returns 0, if X > 0 SIGN returns a 1, if X < 0 SIGN returns a value of -1.
SIN()	Sine of a value.
SINH()	Hyperbolic sine of a value.
SIZE()	Returns the size (number of elements) in an array.
SQR()	Squares a number.
SQRT()	Square root of a number.
STDDEV()	Computes the standard deviation of an input array of numbers.
TAN()	Tangent function.
TANH()	Hyperbolic tangent of a number.
TOTAL()	Returns the total of all elements in an array.
TRUNC()	See the definition of INT() above.

MESSAGE

Displays a character string on the screen.

FORMAT

MESSAGE message

PARAMETERS

message
Message text to be displayed. If no menu is currently displayed (see the Form Driver chapter), GLI will write the message on the current command line.

DESCRIPTION

MESSAGE is used in a macro to display a character string, normally on line 24 of the screen. Line 24 is always deleted before the text is displayed. If the message does not fit on the current screen, it is truncated.

EXAMPLE

In this example, MESSAGE informs the user that a data file is being read.

```
! read data file
read data.dat x, y
message ”reading data...”
.
.
.
exit
```

ON…THEN

Executes a command upon the occurrence of the specified Form Driver event within a macro.

FORMAT

ON event THEN command

PARAMETERS

event
Event specification. E1 to E8 represent user-defined events 1 to 8. F1 to F4 represent the function keys PF1 to PF4 (or F1 to F4 keys). REFRESH and CTRL_Z represent the standard Form Driver events.

command
Any valid GLI command.

DESCRIPTION

ON is used in a macro to specify ON conditions previous to displaying a Form Driver menu (see the Form Driver chapter). During the display of a menu, the Form Driver may signal several events. With the ON command, you can establish a course of action for GLI to take. A RETURN statement is needed when a pre-defined Form Driver event is to be done in GLI.

EXAMPLE

```
 ! Load menu interface
 load ’gli_demo’demo.fdv
 .
 .
 .
on f1 then return do
on f4 then return exit
 .
 .
 .
display
 .
 .
 .
exit
```

PRINT

Prints a variable or the result of a function on the display.

FORMAT

PRINT parameter…

PARAMETERS

parameter
You may use the PRINT command with one or more of the following:

- An arithmetic expression;
- A constant;
- A variable or function;
- An array.

DESCRIPTION

PRINT is used to display the values assigned to variables or the result of a GLI function. If more than one values is assigned to a variable (i.e., the variable is an array) all values will be printed, in order, starting with the first value and ending with the last.

Print can be used to determine the maximum and minimum values as input for the SET SPACE command when producing contour pr surface plots.

EXAMPLE

The following would display the minimum (1200) and maximum (3900) values assigned to variable X.

```
gli> print min(x) max(x)
1200 3900
```

This page intentionally left blank.

QUIT

Exits the GLI command language and returns control to the calling process.

FORMAT

QUIT

PARAMETERS

None.

DESCRIPTION

QUIT performs the following actions before exiting the GLI command language and returning to your operating system.

The following actions are performed:

1. Close the journal file.
2. Delete all symbols, variables and functions.
3. Close the Graphical Kernel System (GKS).

QUIT differs from EXIT in that QUIT does not store data in the GLI.JOU file.

EXAMPLES

Use QUIT on the command line to leave GLI and return to your operating system.

```
gli> quit
$
```

READ

Reads an input file and assigns its contents to the specified variables.

FORMAT

READ file-name var-name,…

PARAMETERS

file-name
Specifies the name of the input file to be read. If you do not specify a file type, the system uses the default file type of `.DAT`. No wild card characters (* or %) are allowed in the file specification.

var-name
Specifies a 1 to 31 alphanumeric-character variable name.

DESCRIPTION

READ inputs free-formatted data from files. READ handles text as follows:

- If a line in the file only contains numerical information (e.g., 1 2 3), the numerical information is read as data.
- If a line in the file only contains textual information (e.g., TEXT), the line is considered a text string and is assigned to one of the string symbols, P1 to Pn.
- If a line in the file begins with text (e.g., TEXT 1 2 3), the entire line is considered a text string and is assigned to one of the string symbols, P1 to Pn.
- If a line in the file begins with data (e.g., 1 2 3 TEXT), only the textual information on the line is considered a text string and is assigned to one of the string symbols, P1 to Pn. The numerical information is read as data.

EXAMPLE

In this example, GLI reads DATA.DAT as X and Y variables. The message indicates the file contains non-numerical data that is not read.

```
gli> read data.dat x,y
100 records read from file data.dat
textual information has been ignored (2 records)
gli>
```

RECOVER

Reads the GLI.JOU file and repeats all the instructions it contains.

FORMAT

RECOVER

PARAMETERS

None.

DESCRIPTION

RECOVER allows reconstruction of a graphics image after an abnormal exit from the GLI system. As instructions are issued on the command line, the GLI system records them in a journal file. RECOVER executes all previously issued commands.

EXAMPLE

```
gli> CTRL/Y
[interrupt]
% gli

 G L I
 ALPHA version 4.5 (OSF/1)
 patchlevel 4.5.4, 20 Nov 95

 Copyright @ 1986-1995, Josef Heinen, Jochen Werner
 Copyright @ 1995, ZAM, Forschungszentrum Juelich GmbH (GR-Software)

 Send bugs and comments to J.Heinen@KFA-Juelich.de

gli> recover
           .
           .
           .
gli> q
```

REDIM

Re-sample an array to given dimensions.

FORMAT

REDIM array start_x dimension_x start_y dimension_y x1 xn y1 yn

PARAMETERS

file-name
Data specification for the array to be re-sampled.

Specifies a 1 to 31 alphanumeric-character variable name.

start_x dimension_x start_y dimension_y
Integers specifying the dimensions of the original array.

x1, xn, y1, yn
Integers specifying the dimensions of the re-sampled array.

DESCRIPTION

REDIM allows re-sampling of a given array to any size. The resulting sub-array may exceed its original dimensions. Regions outside the original array dimensions are filled with zeros.

RETURN

Terminates a labeled subroutine in a macro and returns control to the subroutine caller.

FORMAT

RETURN [status]

PARAMETERS

status
Optional status code to be returned to the Form Driver. See the Form Driver chapter.

DESCRIPTION

RETURN terminates a labeled subroutine in a macro and transfers control to the command following the calling GOSUB command.

EXAMPLE

In this example, the GOSUB command transfers control to the subroutine labeled AXES in the macro. After drawing the axes, the subroutine is terminated by the RETURN command. Control is then returned to the command following the calling GOSUB statement (the GKS POLYLINE X, Y command).

```
gosub axes
gks polyline x, y
exit
! subroutine axes plots a pair of axes
axes:
gus autoscale x, y
gus axes_2d
return
```

RPC

The command which starts a GLI Remote Procedure Call.

FORMAT

RPC “command” [variable1, variable2, ...]

PARAMETERS

command

the command, and any arguments needed by the command enclosed in quotation marks.

[variable1, variable2, ...]

up to 32 optional variables may be passed into and out of a GLI Remote Procedure Call.

DESCRIPTION

The GLI RPC command calls another currently running program from within the GLI environment. RPC’s may be used to perform many functions that GLI can’t, or to provide access to other devices and/or computers. For more information see Appendix I, GLI Remote Procedure Calls.

EXAMPLE

Read in data from a file using a GLI RPC that has been included with the program.

```
gli> rpc “read temp.bin” x
```

SAVE

Saves the contents of the GLI.JOU file in a file other than the default.

FORMAT

SAVE file-name

PARAMETERS

file-name
Any valid file specification. If you do not specify a file type, the system uses the default file type of `.GLI`. No wild card characters (* or %) are allowed in the file specification.

DESCRIPTION

SAVE saves all collected commands in a file other than GLI.JOU journal file. Use SAVE with the LEARN command to create a macro:

1. Invoke the LEARN command.
2. Enter the commands you want to be executed in your macro.
3. Store all collected command input using the SAVE command.

EXAMPLE

This example saves the journal file to a file named NEWPLOT.GLI.

```
gli> learn
gli> read data.dat x, y
21 lines read from file data.dat
gli> gus autoscale x, y
gli> gus axes_2d
gli> gus polyline x, y
gli> save newplot.gli
```

SET DEFAULT

Sets your default device and directory specifications.

FORMAT

SET DEFAULT [device-name[:]][directory-spec]

PARAMETERS

device-name[:]
The name of the device you want to use (i.e., disk: DISK$1:).

directory-spec
The name of the directory you want to go to. A directory name must be enclosed in brackets ([ ]). Use the minus sign (-) to specify the next higher directory from the current default.

DESCRIPTION

Sets device and directory specifications, allowing you to go to other directories and/or storage devices. You must either specify the device-name parameter or the directory-spec parameter.

Note: VMS users may have to specify another disk if you are moving from one disk to another (i.e., DISK$1:[] to DISK$2:[]).

EXAMPLES

In this example, SET DEFAULT changes the default directory to [SMITH]. The default disk device does not change.

```
gli> set default [smith]
```

In this example, SET DEFAULT changes the default directory to the SMITH.DATA sub-directory on $FLOPPY1.

```
gli> set default $floppy1:[smith.data]
```

SET HOST

Connects the GLI system to a remote system through an out-going terminal line. Exit from the remote system by typing CTRL/\ (that is, type a back-slash (\) while holding down the CTRL key).

Any TEKtronix 4014 terminal control commands received from the remote system are translated into GKS commands.

Note: This requires the ability to assign a channel to the terminal port specified.

FORMAT

SET HOST terminal-name

PARAMETERS

terminal-name
Specifies the name of an out-going terminal line, which connects your system directly to another system or to a modem.

DESCRIPTION

SET HOST allows you to connect your terminal to another system through an out-going terminal line (rather than through a network).

To exit from the remote node, type CTRL/\; that is, type a back-slash (\) while pressing the CTRL key.

To log in on lines that expect a break rather than a carriage return, type CTRL/] (that is, type a bracket (]) while holding down the CTRL key) to generate a break.

EXAMPLE

SET HOST in this example connects the user terminal to the out-going terminal line TTA2:.

```
gli> set host tta2:
Connection established to remote port, type ^\ to exit
.
.
.
[CTRL/\]
Control returned to local node
gli>
```

SET LOG

Controls the logging information displayed to the terminal.

FORMAT

SET[NO]LOG

PARAMETERS

None.

DESCRIPTION

SET LOG causes the system to display available logging information (such as the number of data points read) for each command as it is executed.

SET NOLOG turns off logging. System responses and error messages, however, are always displayed.

EXAMPLE

In this example, when the macro `PLOT.GLI` is executed, the logging information is displayed at the terminal. then, SET NOLOG turns logging off.

```
gli> set log
gli> @plot
.
.
.
gli> set nolog
```

SET PROMPT

Replaces the default GLI prompt (gli>) with the specified text string.

FORMAT

SET PROMPT prompt

PARAMETERS

prompt
Specifies a new prompt. The following rules apply:

- All valid ASCII characters can be used;
- No more than 32 characters are allowed;
- To include spaces, enclose the string in quotation marks. Otherwise, leading and trailing spaces are removed.

DESCRIPTION

SET PROMPT customizes the GLI prompt for your GLI session. If the prompt parameter is not given with the SET PROMPT command, the default GLI prompt (gli>) is restored.

EXAMPLE

In this example, SET PROMPT replaces the default GLI prompt with the phrase “What’s next?”

```
gli> set prompt what’s next?
what’s next?
.
.
.
what’s next? set prompt
gli>
```

SET UPDATE

Updates the screen every 2 seconds for X window displays without a backing store.

FORMAT

SET [NO]UPDATE

PARAMETERS

None.

DESCRIPTION

SET [NO]UPDATE is used on X window displays which do not have a backing store. When SET UPDATE is enabled the workstation will be updated every 2 seconds, re-drawing obscured portions of the display. This feature is disabled by default. Typing SET UPDATE at the command line enables this feature, while typing SET NOUPDATE turns off this feature.

EXAMPLE

```
  gli> set update
```

SET VERIFY

Displays the command lines in the macro as they are executed, allowing for verification.

FORMAT

SET[NO]VERIFY

PARAMETER

None.

DESCRIPTION

SET VERIFY causes the system to display each command in a macro as it reads it. When verification is in effect, the command interpreter displays each command line after it has completed initial scanning and before the command is parsed and executed.

The default setting for macros executed interactively is SET NOVERIFY. System responses and error messages are, however, always displayed.

EXAMPLE

In this example, SET VERIFY turns command verification on. When the macro `PLOT.GLI` is executed, the command lines are displayed at the terminal. Then the SET NOVERIFY command turns off command verification.

```
gli> set verify
gli> @plot
.
.
.
gli> set noverify
```

SHOW FUNCTION

Displays the expression for a specified function.

FORMAT

SHOW FUNCTION function-name,…

PARAMETERS

function-name
Specifies one or more function names for which expressions are to be displayed. The function name can have from 1 to 31 alphanumeric characters. The function name must begin with an alphabetic character. Wild card characters (* and %) are allowed.

DESCRIPTION

SHOW FUNCTION displays function expressions. If you specify a function name, its expression is displayed. If you do not specify a function name, all functions and associated expressions are displayed.

EXAMPLE

```
gli> show function
f = sin(1/x)*x
g = 1/f**2+f
total of 2 functions.
gli>
```

SHOW SYMBOL

Displays the value for a specified symbol.

FORMAT

SHOW SYMBOL symbol-name,…

PARAMETERS

symbol-name
Specifies one or more symbol names for which equivalent values are to be displayed. The symbol name can have from 1 to 31 alphanumeric characters. The symbol name must begin with an alphabetic character. Wild card characters (* and %) are allowed.

DESCRIPTION

SHOW SYMBOL displays symbol values. If you specify a symbol name, its value is displayed. If you do not specify a symbol name, all symbols and associated values are displayed.

EXAMPLE

```
gli> show symbol *
pag*e = ”gks clear_ws”
axe*s = ”gus axes_2d”
total of 2 symbols.
gli>
```

SHOW VARIABLE

Displays information for all variables currently defined.

FORMAT

SHOW VARIABLE variable-name,…

PARAMETERS

variable-name
Specifies one or more variable names for which information is to be displayed. The variable name(s) can have from 1 to 31 alphanumeric characters and must begin with an alphabetic character. Wild card characters (* and %) are allowed.

DESCRIPTION

SHOW VARIABLE displays variables and allocation information. If a variable name(s) parameter is specified, the associated allocation is displayed. If variable name(s) parameter is not specified, all variables are displayed.

EXAMPLE

```
gli> x := 0(0.1)10
gli> y := sin(x)
gli> show variable
x (101 values)
y (101 values)
total of 2 variables, 202 values.
gli>
```

SIGHT

Invokes the Simple Interactive Graphics Handling Tool (SIGHT).

FORMAT

SIGHT

PARAMETERS

None.

DESCRIPTION

Invokes the SIGHT interface. See the SIGHT chapter in the GLI documentation.

SIGHT requires workstation hardware and Motif X Window software.

EXAMPLE

```
gli> sight
```

SIMPLEPLOT

Invokes the SIMPLEPLOT interface.

FORMAT

SIMPLEPLOT

PARAMETERS

None.

DESCRIPTION

Invokes the SIMPLEPLOT interface which can be used to quickly plot, filter and manipulate 2-D data files. See SIMPLEPLOT in the Basic Usage chapter.

EXAMPLE

```
  gli> simpleplot
```

SLEEP

Pauses execution of GLI for a specified number of seconds.

FORMAT

SLEEP seconds

PARAMETER

seconds
a whole number of seconds for which GLI is to wait

DESCRIPTION

SLEEP suspends the execution of GLI for a specific number of seconds. After GLI is done ‘sleeping’, GLI will resume execution. Pressing control-c will interrupt GLI’s slumbers, and GLI cannot sleep for fractions of a second.

EXAMPLE

GLI will ‘sleep’ for three seconds.

```
gli> sleep 3
```

SMOOTH

Smooth a given sequence of data points.

FORMAT

SMOOTH variable [smoothing-level, [segment-size]]

PARAMETERS

variable
A variable specification.

smoothing-level
Specifies a smoothing level in the range 1 to 20.

segment-size
Segment size of the variable.

DESCRIPTION

SMOOTH smooths a given sequence of data points. The smoothing procedure is non-linear. The variable is assumed to be a sequence of observed values of a function of equally spaced intervals.

See Appendix E for a description of GLI’s math functions.

EXAMPLE

```
gli> smooth y
```

VIEW

Invokes the GLI FileViewer.

FORMAT

VIEW filename, *format, interface*

PARAMETERS

filename
Name of the file

format

Graphics file format (TEK, CGM_BINARY, CGM_CLEAR).

interface

GKS, SIGHT

DESCRIPTION

Invokes the GLI File Viewer.

The file is displayed when you press the RETURN key.

By default, the GLI File Viewer expects to display TEK4010 files.

EXAMPLE

```
gli> view tek.file
```

WRITE

Writes one or more variables into an output file.

FORMAT

WRITE file-name data-spec, ...

PARAMETERS

file-name
Specifies the name of the output file to be written. If you do not specify a file type, the system uses the default file of .DAT. No wild card characters (* or %) are allowed in the file name.

data-spec
Data specification. A data specification may be:

- An arithmetic expression;
- A constant;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

WRITE writes data to sequential files. The maximum size of any record that can be written is 256 bytes.

EXAMPLE

```
gli> write data.dat x,y
100 records written to file usr$usrdisk:[smith] data.dat; 1
gli>
```

XUI

Invokes the XUI interface.

FORMAT

XUI

PARAMETERS

None.

DESCRIPTION

XUI invokes the XUI interface, allowing use of an X Window, point-and-click interface instead of the keyboard. Workstation hardware and X WIndows is required to support XUI.

EXAMPLE

```
gli> xui
```

XUI CHOICE

Builds pop up menus for customized X Window interfaces.

FORMAT

CHOICE name [|] choice1[|] choice2…

PARAMETERS

name
Symbolic name representing the menu. The name also appears as the title of the pop-up menu containing the command choices.

[|]

the ‘|’ character is an option used to allow the user to create multiple word ‘choices’ in a menu

choice1, choice2…
Alphanumeric string name for the command to be executed from the menu.

DESCRIPTION

XUI CHOICE builds a pop-up menu containing commands that, when selected, may be executed. In your command procedure, define XUI CHOICE with the name of your menu and the choices for commands.

Use of the ‘|’ character allows the definition of multiple word menu items in the choice command. A ‘|’ placed after the name of the choice menu box enables multi-word mode so that menu entries of more than one word can now be made. Entries are separated by a ‘|’. The two menu modes cannot be mixed, and the maximum length of a string of characters in both menu modes cannot exceed 256 characters in length.

The choices may be tested and executed using the GLI CASE command.

Note: The XUI CHOICE command is compatible with systems properly configured to support X Windows.

EXAMPLE

In this example, a menu named “Plots” is defined, containing commands to execute an XY plot, a contour graph, a surface representation, and to exit the menu.

```
gli>xui choice Plots XY_Graph Contour Surface Exit
```

In this example a multiple word menu is created.

```
gli> xui choice Honeymooners | Ralph and Alice | Ed and Trixie
```

CHAPTER 4

SIGHT Command Language

Table of Contents

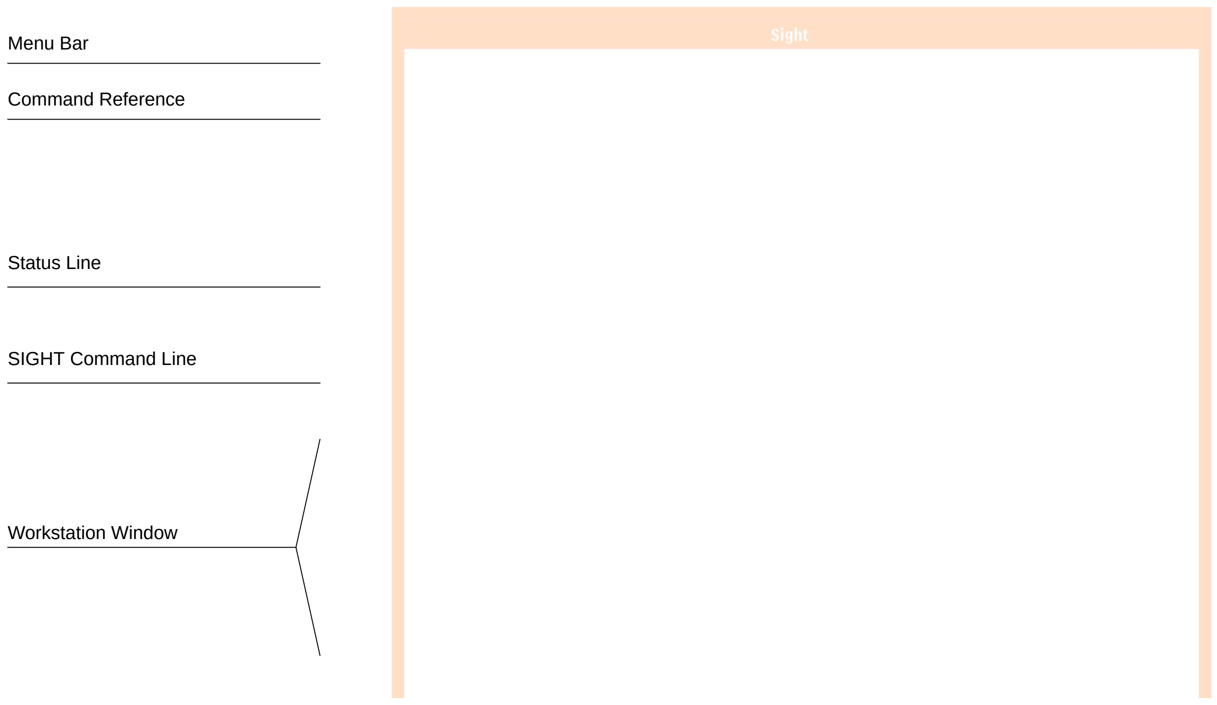
Table of Contents continued

Overview

This chapter describes the Simple Interactive Graphics Handling Tool (SIGHT) commands. The commands are arranged in alphabetical order. Each description contains the command’s purpose, menu bar location, parameters, and basic usage.

SIGHT is a high level graphics editor using in a Motif and X Windows point-and-click interface. The interface utilizes a combination of pull down and pull right menus, pop up dialog boxes, and scroll lists. Refer to your Motif and X Windows system documentation for complete instructions on using these features.

FIGURE 4-1 SIGHT Screen Layout



SIGHT Menu Interface

Menu Bar
The menu bar contains the main headings for the different types of SIGHT commands

Command Reference
SIGHT maintains a listing of the commands you issue, allowing you to select them using the mouse and issuing them again.

Select the command by clicking on it with the left mouse button. The command automatically appears in the SIGHT Command Line (see below).

Status Line
When you select an object in your drawing SIGHT describes the status, or attributes, of the object on the line above the SIGHT Command Line.

SIGHT Command Line
The SIGHT> prompt indicates the current command. You can select and then edit commands with the keyboard before issuing them.

Workstation Window
Your graphic is drawn in the workstation window, beneath the SIGHT Menu Interface.

SIGHT Usage Notes

Click on a previous command to make it appear on the SIGHT command line, allowing you to edit it. Press RETURN to execute the command.

Double-click on a previous command to execute it again (same as clicking on the command and pressing RETURN).

When you click on an object in your drawing, the Status Line displays the object's attributes.

Mouse Usage Notes

Clicking the left mouse button on an object selects it. Clicking on a selected object with the left mouse button deselects it.

Clicking the middle mouse button will cut the current selection, storing it on the SIGHT clipboard.

Clicking the right mouse button will paste what is on the SIGHT clipboard to the current mouse pointer position.

Command Descriptions

The following command descriptions are unique to SIGHT.

AXES

Draws x and y coordinate axes using linearly or logarithmically spaced tick marks.

LOCATION

TOOLS

PARAMETERS

x-tick, y-tick
Interval length between tick marks for both axes. For example, an x-tick or y-tick set to 1 when the window it defined from 0 to 10 will place tick marks at every interger point on the axes (0, 1, 2, 3, . . ., 10).

x-origin, y-origin
The coordinates of the origin (point of intersection) of the two axes.

major-x, major-y
Integer values that define the spacing for major tick marks. Values of 0 or 1 imply no minor ticks. For instance, if a window is defined from 0 to 100 for an axis, and major-x or major-y is set to 5 for that axis, every 5th tick mark will be labeled a major tick mark (0, 5, 10, 15, . . ., 100).

tick-size
The length of minor tick marks specified in normalized device coordinate units. Major tick marks are twice as long as minor tick marks. A negative value reverses the tick marks on the axes from inward facing to outward facing.

USAGE

Select AXES from the menu to draw a pair of labeled coordinate axes on your graph using default settings. Tick marks are positioned along each axis so that major tick marks fall on the axes origin. Axes and tick marks are drawn using solid lines.

For finer control, enter the SIGHT AXES command with its parameters on the SIGHT command line:

```
  SIGHT> sight axes 0.1, 0.1, 0, 0, 2, 2, -0.02
```

CAPTURE DRAWING

Captures the drawing in an encapsulated PostScript file (GLI.EPS).

LOCATION

FILE

PARAMETERS

None.

USAGE

Select CAPTURE DRAWING to write the drawing to an encapsulated PostScript file (GLI.EPS) in your current directory. The file can then be printed using the printer commands for your system.

Change the GLI_FIG variable in the GLISETUP file to define a different save directory for the encapsulated PostScript files.

CLEAR DRAWING

Clears the current drawing and deletes all objects.

LOCATION

EDIT

PARAMETERS

None.

USAGE

Select CLEAR DRAWING to delete all objects on the display surface. Cntl C is the keyboard equivalent to this command.

This is the same as the GKS command, CLEAR_WS (Clear Workstation).

CLOSE DRAWING

Saves the current drawing to a file.

LOCATION

FILE

PARAMETERS

None.

USAGE

Select CLOSE DRAWING to save the current drawing instructions.

Unless you opened the current drawing with CREATE DRAWING and specified a different file name, the instructions will be written to the default file, SIGHT.SIGHT.

CLOSE_SIGHT

Closes the Simple Interactive Graphics Handling Tool (SIGHT) and returns the user to GLI in the state GLI was in when SIGHT was started.

LOCATION

GLI COMMAND LINE

PARAMETERS

None.

USAGE

Type SIGHT CLOSE_SIGHT in at the GLI command line to return to GLI in the state GLI was in prior to starting a SIGHT session with a SIGHT OPEN_SIGHT command.

CREATE ATTRIBUTE BOX

Allows you to define attributes (i.e., color, size, style, etc.) for quick assignment to lines, markers, fill areas, and text.

LOCATION

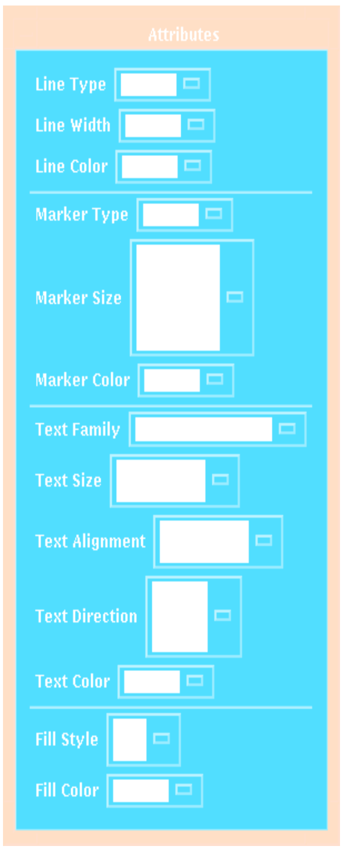
ATTRIBUTES

PARAMETERS

None.

USAGE

To modify the attribute of primitives, such as lines, markers, areas or text, select the Attribute menu choice and display the Create Attribute Box shown below. Click on the appropriate attribute to reveal the choices available, then click on the selection to invoke that attribute for subsequent primitives.



CREATE DRAWING

Creates a new drawing, and clears the current drawing.

LOCATION

FILE

PARAMETERS

file-name
Specifies the name of the file to be created.

USAGE

Select CREATE DRAWING to create a new drawing. Unless you specify a particular directory and file name, GLI creates a new SIGHT drawing in the current directory named SIGHT.SIGHT.

You can use the directories scroll box to:

- Select a drawing from the current directory.
- Select from another directory.

You can add wild cards in the selection box and click on the Filter button to:

- Specify a wild card selection (e.g., *.drawing) in the current directory.
- Specify a wild card selection in another directory.

After you specify your SIGHT drawing, click the OK button to open the drawing. Otherwise, click the CANCEL button to abort the command.

By default, GLI uses the current directory and displays files with the .SIGHT suffix.

CREATE DRAWING will clear the display area of any previous drawing.

CREATE TOOL BOX

Allows you to define commonly used tools (i.e., lines, axes, error bars, points, grids, etc.) for quick assignment to your graph.

LOCATION

TOOLS

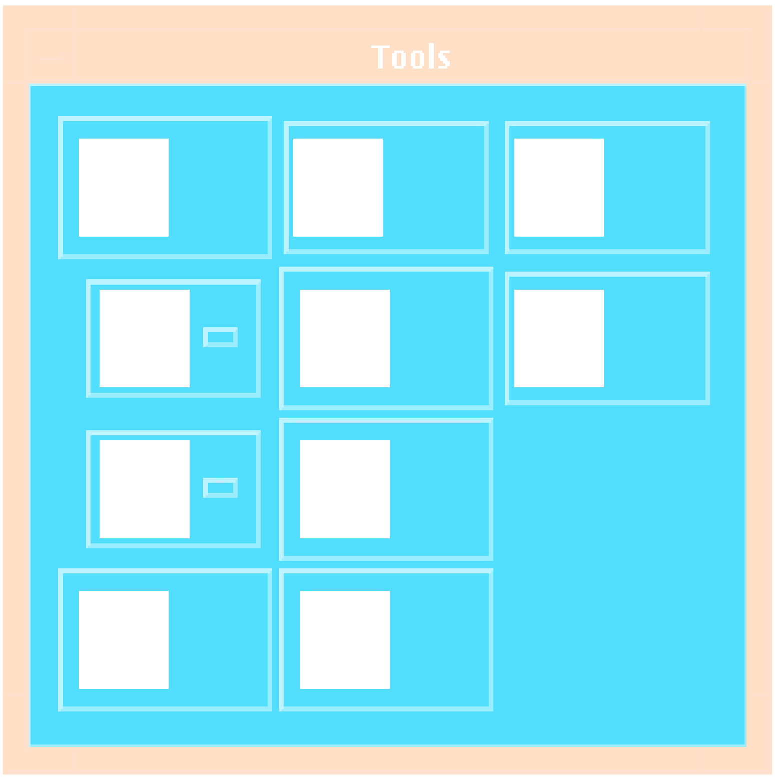
PARAMETERS

None.

USAGE

To display the graph tools available from SIGHT, select the Attributes menu and click on the Create Attribute Box choice. A palette of SIGHT tools are displayed. To apply a tool to your graph, click on the appropriate choice to reveal that tool's options.

Important: Before a tool is applied, the window containing the graph must first be activated.



CUT

Deletes the current selection and copies it to the SIGHT clipboard.

LOCATION

EDIT

PARAMETERS

None.

USAGE

You must first SELECT an object or group of objects before you may use the CUT command.

Use CUT to place an object in the SIGHT “clipboard” buffer. Then select the PASTE command to replicate the clipboard contents in new locations.

Only one selection may be stored on the clipboard at a time.

DESELECT

Deselects all currently selected objects.

LOCATION

EDIT

PARAMETERS

None.

USAGE

Select DESELECT to cancel the SELECT command, immediately deselecting all currently selected objects.

When using SIGHT, you may deselect an object by positioning the mouse pointer over a selected object and clicking the left mouse button.

DIGITIZE

Reads data from a real or simulated digitizing tablet.

LOCATION

EDIT

PARAMETERS

input area
Define the input area of the digitizing tablet.

points
Digitize data points (X and Y).

bars
Digitize data points (X and Y) for use with Error Bars (E1 and E2).

USAGE

Select DIGITIZE INPUT AREA to define the input area of your digitizing tablet. SIGHT prompts you to select the three points that define the page's X and Y axes. Once the conceptual square is matched with the actual chart square, SIGHT can process the data points accordingly. If SIGHT does not have the required points for the input area, it will request them, regardless of the parameter you select (INPUT AREA, POINTS, or BARS).

Select DIGITIZE POINTS to input X and Y data by selecting points on the digitizing tablet. Click each point with the left mouse button to record it.

Select DIGITIZE BARS to input X and Y data with associated error determination arrays (E1 for negative range, E2 for positive range). Specify how the error bars (see ERROR BARS) will be used: horizontally or vertically. Then, for each point, you will click the left mouse button once to record the point, a second time to specify the positive error, and a third time to specify the negative error.

ERROR BARS

Compute and display the error values at the specified points.

LOCATION

TOOLS

PARAMETERS

horizontal
Plot error bars horizontally.

vertical
Plot error bars vertically.

USAGE

You need to specify X and Y data and associated error determination arrays (E1 and E2) before using ERROR BARS.

Select ERROR BARS to plot horizontal or vertical error bars for each data point.

EXIT

Closes SIGHT.

LOCATION

FILE

PARAMETERS

None.

USAGE

Select EXIT to close the SIGHT interface. All issued graphics commands are saved in the SIGHT.SIGHT file. Cntl Z is the keyboard equivalent for this command.

EXIT differs from the QUIT command in that EXIT does not ask for confirmation before closing SIGHT.

FILL AREA

Fills the defined polygon, according to the current interior style.

LOCATION

TOOLS

PARAMETERS

locate
Define polygon with the locator device (e.g., mouse).

data
Define polygon with current data.

USAGE

Select FILL AREA to fill a polygon according to the current interior style specifications. The default fill type is HOLLOW. You may want to change fill area attributes with the FILL STYLE command.

When using a locator device to define data points, select beginning and subsequent data points with the left mouse button. Select the ending data point with the middle mouse button. The polygon is then drawn and filled.

Note: Specifying data with a locator redefines the current X and Y values.

If parts of the area are clipped, the new boundaries generated become part of the area boundaries. Multiple sub-areas may be generated.

FILL COLOR

Sets the current fill area color entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

color
Specifies one of the available colors (see Table 4-1, below).

TABLE 4-1 SIGHT Color Chart

Color	Red	Green	Blue
WHITE	1.000	1.000	1.000
BLACK	0.000	0.000	0.000
RED	1.000	0.000	0.000
GREEN	0.000	1.000	0.000
BLUE	0.000	0.000	1.000
CYAN	0.000	1.000	1.000
YELLOW	1.000	1.000	0.000
MAGENTA	1.000	0.000	1.000

USAGE

Select FILL COLOR to change the color used for the solid fill of the currently selected object.

All subsequent solid filled polygons will be drawn using the new fill color.

FILL INDEX

Sets the type of hatching or fill to be used in filling an area.

LOCATION

ATTRIBUTES

PARAMETERS

1-8
Numbers corresponding to the type of pattern or hatching used to fill a figure.

USAGE

After you select FILL STYLE, either the pattern or hatch styles, the user may use FILL INDEX to vary the style of hatch or pattern used to fill an area.

FILL STYLE

Sets the current fill area style entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

style
Specifies one of the available fill styles (see Table 4-2, below).

TABLE 4-2 Fill Styles

Switch	**Description**
HOLLOW	No filling, only draw the bounding polyline.
PATTERN	A pattern is used to fill the polygon.
HATCH	A hatched pattern is used to fill the polygon.
FILLED	Fill the interior of the polygon.

USAGE.

Select FILL STYLE to change how the interior of the current selected object is to be filled.

All subsequent polygons will be drawn using the new fill style.

FORMAT

Defines the page size for printing your drawing.

LOCATION

VIEW

PARAMETERS

A3
0.42 x 0.297 meters

A4
0.297 x 0.21 meters

USAGE

Select FORMAT to change the page size when you print your output file (see CAPTURE DRAWING and PRINT DRAWING).

Important: This option is not available in GLI version 4.4

GRID

Draws a linearly or logarithmically-spaced grid.

LOCATION

TOOLS

PARAMETERS

x-tick, y-tick
Interval length between grid lines.

x-origin, y-origin
The coordinates of the origin (point of intersection) of the grid lines.

major-x, major-y
Integer values specifying the number of minor tick intervals between grid lines. Values of 0 or 1 imply no grid lines.

USAGE

Select GRID to draw a grid on your graph using default settings. Grid lines are positioned along each axis at the major tick marks. Major grid lines are drawn using solid lines; minor grid lines are drawn using dashed lines.

For finer control, enter the GRID command with its parameters on the SIGHT command line:

```
  SIGHT> sight grid 0.1, 0.1, 0, 0, 2, 2
```

HELP

Provides basic on-line documentation for using SIGHT.

LOCATION

HELP

PARAMETERS

None.

USAGE

Click on the HELP menu button and information about SIGHT can be accessed from the window in which SIGHT was started. Topics and sub-topics can be displayed.

IMPORT

Reads in a CGM metafile, either in clear text or binary format.

LOCATION

FILE

PARAMETERS

CGM Binary

Import a CGM file that has been saved in binary format.

CGM Clear Text

Import a CGM file that has been saved in clear text format.

USAGE

Select Import CGM Binary to read in a binary encoded CGM file.

Select Import CGM Clear Text to read in a text encoded Computer Graphics Metafile. The text encoded files can be edited by any standard text editor to change the file contents, and therefore the picture drawn by the file.

The Computer Graphics Metafile import command supports reading in of CGM files that have been encoded in accordance with the ANSI X3.122-1986 computer graphics file standard.

Note: The default file format is binary.

INQUIRE

The inquiry function displays information about a selected object.

LOCATION

GLI COMMAND LINE

PARAMETERS

NONE.

USAGE

First select an object and then type SIGHT INQUIRE at the command line. Information about the selected object will be displayed on the SIGHT status line.

LINE

Generates a sequence of connected straight lines, starting from the first point and ending at the last point using the current polyline attributes.

LOCATION

TOOLS

PARAMETERS

locate
Define polyline with the locator device (e.g., mouse).

data
Define polyline with current data.

USAGE

Select LINE to draw a polyline, starting from the first data point and ending at the last data point, using the current polyline attributes.

When using a locator device to define data points, select beginning and subsequent data points with the left mouse button. Select the ending data point with the middle mouse button.

Note: Specifying data with a locator redefines the current X and Y values.

LINE COLOR

Sets the current line color entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

color
Specifies one of the available colors (see Table 4-3, below).

TABLE 4-3 SIGHT Color Chart

Color	Red	Green	Blue
WHITE	1.000	1.000	1.000
BLACK	0.000	0.000	0.000
RED	1.000	0.000	0.000
GREEN	0.000	1.000	0.000
BLUE	0.000	0.000	1.000
CYAN	0.000	1.000	1.000
YELLOW	1.000	1.000	0.000
MAGENTA	1.000	0.000	1.000

USAGE

Select LINE COLOR to change the line color of the currently selected object.

All subsequent lines will be drawn using the new color.

LINE TYPE

Sets the current line type entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

line-type
Specifies one of the available line types (see Table 4-4, below).

TABLE 4-4 Polyline Line Types

Line Type	Description
SOLID	Solid line.
DASHED	Dashed line.
DOTTED	Dotted line.
DASH-DOTTED	Dashed_dotted line.
DASH, 2 DOTS	Sequence of one dash followed by two dots.
DASH, 3 DOTS	Sequence of one dash followed by three dots.
LONG DASH	Sequence of long dashes.
LONG, SHORT DASH	Sequence of a long dash followed by a short dash.
SPACED DASH	Sequence of dashes double spaced.
SPACED DOT	Sequence of dots double spaced.
DOUBLE DOTS	Sequence of pairs of dots.
TRIPLE DOTS	Sequence of groups of three dots.

USAGE

Select LINE TYPE to change the line pattern of the currently selected object.

All subsequent polylines will be drawn using the new line type.

LINE WIDTH

Sets the current line width entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

width
Specifies one of the available line widths (see Table 4-5, below).

TABLE 4-5 Line Widths

Width	Description
1	1 point
2	2 point
3	3 point
4	4 point
5	5 point
6	6 point

USAGE

Select LINE WIDTH to change the line thickness of the currently selected object.

All subsequent polylines will be drawn using the new line width.

MARKER COLOR

Sets the current polymarker color entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

color
Specifies one of the available colors (see Table 4-6, below).

TABLE 4-6 SIGHT Color Chart

Color	Red	Green	Blue
WHITE	1.000	1.000	1.000
BLACK	0.000	0.000	0.000
RED	1.000	0.000	0.000
GREEN	0.000	1.000	0.000
BLUE	0.000	0.000	1.000
CYAN	0.000	1.000	1.000
YELLOW	1.000	1.000	0.000
MAGENTA	1.000	0.000	1.000

USAGE

Select MARKER COLOR to change the polymarker color of the currently selected object.

All subsequent polymarkers will be drawn using the new color.

MARKER SIZE

Sets the current marker size scale factor entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

size
Specifies one of the available marker sizes (see Table 4-7, below).

TABLE 4-7 Marker Sizes

Size	Description
6	6 points
8	8 points
10	10 points
12	12 points
14	14 points
18	18 points
24	24 points
36	36 points
48	48 points
72	72 points

USAGE

Select MARKER SIZE to change the polymarker size of the currently selected object.

All subsequent polymarkers will be drawn using the new size.

MARKER TYPE

Sets the current marker type entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

marker-type
Specifies one of the available marker types (see Table 4-8, below).

TABLE 4-8 Marker Types

Marker Type	Description
DOT	Smallest displayable dot
CROSS	Cross
ASTERISK	Asterisk
DIAGONAL CROSS	Diagonal cross
CIRCLE	Hollow circle
SOLID CIRCLE	Filled circle
SQUARE	Hollow square
SOLID SQUARE	Filled square
TRIANGLE UP	Hollow triangle pointing upward
SOLID TRIANGLE UP	Filled triangle pointing upward
TRIANGLE DOWN	Hollow triangle pointing downward
SOLID TRIANGLE DOWN	Filled triangle pointing downward
BOWTIE	Hollow bowtie
SOLID BOWTIE	Filled bowtie
HOURGLASS	Hollow hourglass
SOLID HOURGLASS	Filled hourglass
DIAMOND	Hollow diamond
SOLID DIAMOND	Filled diamond
STAR	Hollow star
SOLID STAR	Filled star
TRIANGLE UP DOWN	Star of David
SOLID TRIANGLE LEFT	Solid triangle pointing left
SOLID TRIANGLE RIGHT	Solid triangle pointing right
HOLLOW PLUS	Hollow plus
O-MARKER	O marker

USAGE

Select MARKER TYPE to change the polymarker pattern of the currently selected object.

All subsequent polymarkers will be drawn using the new marker type.

MOVE

Moves all currently selected objects.

LOCATION

EDIT

PARAMETERS

None.

USAGE

You must first SELECT an object or group of objects before you may use the MOVE command.

Select a position in your drawing for the starting vector point by clicking the left mouse button. Extend the vector line in the direction and to the length you want the selected drawing position to be moved. Click the left mouse button again to move the object along the vector.

MOVE does not move objects drawn with logarithmically scaled coordinates.

OPEN DRAWING

Opens a saved SIGHT drawing and displays its contents on the screen.

LOCATION

FILE

PARAMETERS

file-name
Specifies the name of the drawing to be opened.

USAGE

Select OPEN DRAWING to select a SIGHT drawing to display. Cntl. O is the keyboard equivalent for this command.

You can use the directories scroll box to:

- Search for a drawing file in the current directory.
- Search in another directory.

You can add wild cards in the selection box and click on the Filter button to:

- Specify a wild card search (e.g., *.drawing) in the current directory.
- Specify a wild card search in another directory.

After you specify your SIGHT drawing, click the OK button to open the drawing. Otherwise, click the CANCEL button to abort the command.

OPEN DRAWING will replace the current drawing on the display.

OPEN_SIGHT

Saves the state of GLI prior to starting SIGHT.

LOCATION

GLI COMMAND LINE

PARAMETERS

None.

USAGE

Prior to starting SIGHT, typing in SIGHT OPEN_SIGHT at the GLI command line will save the current state of GLI before starting SIGHT. The state of GLI will be restored to its condition prior to starting SIGHT with a SIGHT CLOSE_SIGHT command issued at the GLI command line after exiting SIGHT.

ORIENTATION

Defines the page orientation for printing your drawing.

LOCATION

VIEW

PARAMETERS

landscape
Page width is greater than page height.

portrait
Page height is greater than page width.

USAGE

Select ORIENTATION to change how the page is oriented when you print your output file (see CAPTURE DRAWING and PRINT DRAWING).

PASTE

Copies the contents of the SIGHT clipboard into the drawing at the specified location.

LOCATION

EDIT

PARAMETERS

None.

USAGE

You must first SELECT and CUT an object or group of objects before you may use the PASTE command.

Select the PASTE command to replicate an object or group of objects. Use the mouse to define the new location of the object or group of objects to be pasted. Click the left mouse button to PASTE.

You can repeat this action repeatedly by reselecting the PASTE command, until you use the CUT command to place a new object or group of objects onto the clipboard.

PICK DATA

Selects a single object and transfers object related data to GLI. The type of data that is transferred depends on the selection.

LOCATION

EDIT

PARAMETERS

None.

USAGE

Select PICK DATA to put world coordinates of an object on the display surface into the variables X Y.

The data returned depends on the object selected (see Table 4-9, below).

TABLE 4-9 Transferred Data

Object	Transferred Data
POLYLINE, POLYMARKER, SPLINE, FILL_AREA	Variables X and Y for the points selected.
ERROR_BAR	Variables X and Y for the points selected, variables E1 and E2 for the error bar values.
TEXT	Variables X and Y for the position, symbol TEXT for the text string.

PLOT

Produces a graph of XY data.

LOCATION

TOOLS

PARAMETERS

locate
Define polyline with the locator device (e.g., mouse).

data
Define polyline with current data specification.

USAGE

Select PLOT to produce a polyline, starting from the first data point and ending at the last data point, using the current polyline attributes. PLOT automatically scales and labels the axes to produce readily interpretable graphs.

When using a locator device to define data points, select beginning and subsequent data points with the left mouse button. Select the ending data point with the middle mouse button. The graph is then drawn and labeled.

Note: Specifying data with a locator redefines the current X and Y values.

POP IN FRONT

Moves selected objects in front of other objects.

LOCATION

EDIT

PARAMETERS

None.

USAGE

Select POP IN FRONT to move the selected object or group of objects in front of the other objects on the display. This changes the drawing order: objects in front are drawn last.

If you move a grouped object in front, SIGHT moves all objects in the group, but maintains the drawing order within the group.

PRINT

Prints the drawing.

LOCATION

FILE

PARAMETERS

None.

USAGE

Select PRINT to send the current drawing to a PostScript printer.

The drawing is printed using the UNIX environment variable and/or VMS logical GLI_LPR located in the UNIX .cshrc file and/or the VMS LOGIN.COM file. GLI_LPR controls the print command and printer filter.

PUSH BEHIND

Moves selected objects behind other objects.

LOCATION

EDIT

PARAMETERS

None.

USAGE

Select PUSH BEHIND to move the selected object or group of objects behind other objects on the display. This changes the drawing order: objects in back are drawn first.

If you move a grouped object behind, SIGHT moves all objects in the group, but maintains the drawing order within the group.

QUIT

Closes SIGHT.

LOCATION

FILE

PARAMETERS

None.

USAGE

Select QUIT to close the SIGHT interface. SIGHT will ask for confirmation.

READ DATA FROM FILE

Opens a file and assigns variables X and Y to the data read in from the file.

LOCATION

FILE

PARAMETERS

file-name
Specifies the name of the file to be read.

USAGE

Select READ DATA FROM FILE to import a data specification from a file.

You can use the directories scroll box to:

- Select a data file from the current directory.
- Select from another directory.

You can add wild cards in the selection box and click on the Filter button to:

- Specify a wild card selection (e.g., *.drawing) in the current directory.
- Specify a wild card selection in another directory.

After you specify your data file, click the OK button to import. Otherwise, click the CANCEL button to abort the command.

By default, SIGHT uses the current directory and displays files with the .DAT extension.

REDRAW

Clears the screen and redraws the graph.

LOCATION

EDIT

PARAMETERS

NONE

USAGE

Select REDRAW to clear the screen and then redraw the graph the user is currently working on, changing no parameters used in making the drawing.

REMOVE

Deletes selected objects without copying them to the SIGHT clipboard.

LOCATION

EDIT

PARAMETERS

None.

USAGE

You must first SELECT an object or group of objects before using REMOVE.

Select REMOVE to delete all selected objects without saving them to the SIGHT clipboard.

The SIGHT clipboard is unaffected by remove.

SAVE AS

Saves the current file under a file name other than the default.

LOCATION

FILE

PARAMETERS

file-name
Specifies the name of the file to be saved.

USAGE

Select SAVE AS to save a SIGHT drawing under a different file name. Enter the new file name in the selection edit box.

You can use the directories box to:

- Select a drawing file from the current directory.
- Select from another directory.

You can add wild cards in the selection box and click on the Filter button to:

- Specify a wild card selection (e.g., *.drawing) in the current directory.
- Specify a wild card selection in another directory.

By default, GLI uses the current directory and displays files with the .SIGHT suffix.

After specifying your file, click the OK button to save. Otherwise, click the CANCEL button to abort the command.

SCALE

Sets the type of transformation to be used for producing output primitives.

LOCATION

VIEW

PARAMETERS

scale
Specifies one of the available scale indicators (see Table 4-10, below).

TABLE 4-10 Scale Indicators

Scale	**Description**
LINEAR	Linear scale
X_LOG	Logarithmic X-scale
Y_LOG	Logarithmic Y-scale
XY_LOG	Logarithmic X-, Y-scale

USAGE

Select SCALE to change the scale type (linear or logarithmic) of the current object.

All subsequent objects will be drawn using the new scaling.

SELECT

Select an object or group of objects in the drawing.

LOCATION

EDIT

PARAMETERS

all
Select all objects in the drawing.

excluded
Select all but the previously selected objects.

group of objects
Select all objects which are completely within a user defined rectangle.

next object
Deselect the current object and select the next object.

object
Select a single object.

previous object
Deselect the current object and select the previous object.

USAGE

When you select an object, you can move, copy, delete, or change its attributes. Selected objects are denoted by a dashed line rectangle surrounding the object or group of objects.

When using SIGHT, you may select an object by positioning the mouse pointer over the object and clicking the left mouse button. Clicking the left mouse button on a selected object deselects it.

Choose SELECT GROUP to use the mouse to draw a bounding rectangle specifying a group of objects to be selected. Position the mouse cursor at one corner of the bounding rectangle and press and hold the left mouse button while dragging the mouse to stretch the bounding rectangle. Release the mouse button to complete the selection.

Choose SELECT NEXT and SELECT PREVIOUS to select objects based on the order in which they were created.

Choose SELECT OBJECT to position the mouse cursor over an object and click the left mouse button to select the object. If the object is already selected, it will be deselected. If the object is part of a segment you may have to select it several times.

SELECT TRANSFORMATION

Select the transformation number. Allows multiple coordinate transformations to be used in the SIGHT environment.

LOCATION

VIEW

PARAMETERS

1-8

Up to eight different transformations can be used in the SIGHT environment.

USAGE

Select SELECT TRANSFORMATION to put multiple graphs in the same SIGHT drawing space, or to put axes with different parameters on the same graph.

SELECT VIEWPORT

Determines drawing space for displaying the graphic.

LOCATION

VIEW

PARAMETERS

None.

USAGE

Choose SELECT VIEWPORT when you want to set the viewport limits of the normalization transformation in the SIGHT state list. All objects which are given in NDC values (like text) are repositioned relative to the middle of the new defined viewport.

To reset the viewport, position the mouse pointer on a position that will define an initial corner of the new viewport and click the left mouse button. Extend the selection box to the desired size and click the middle mouse button.

SET WINDOW

Determines what part of the graphic you want displayed (zoom and pan).

LOCATION

VIEW

PARAMETERS

None.

USAGE

Select SET WINDOW when you want to set the normalization transformation by defining the limits of the world coordinate system (window) which are to be mapped onto a specified area of the normalized device coordinate space (viewport).

To reset the window, position the mouse pointer on a position that will define an initial corner of the new viewport and click the left mouse button. Extend the selection box to the desired size and click the middle mouse button.

SNAP GRID

Draws a grid of dots with different densities on the screen.

LOCATION

CUSTOMIZE

PARAMETERS

TABLE 4-11 Snap Grid Parameters

PARAMETER	EFFECT
OFF	Default, no grid drawn.
0.25 cm	Grid drawn with points 0.25 centimeters apart.
0.5 cm	Grid drawn with points 0.5 centimeters apart.
1 cm	Grid drawn with points 1.0 centimeters apart.

USAGE

Snap Grid is used to size a graph for publication. For example, if you have a graph that has to fit in a 5 cm by 5 cm section on a printed page, Snap Grid can be used to draw a grid on the screen to help size the graph properly.

SPLINE

Smooths a given sequence of data points using a natural or weighted cubic spline fit, starting from the first point and ending at the last point, using the current line attributes in the SIGHT state list.

LOCATION

TOOLS

PARAMETERS

natural
Fit a curve along each point.

smooth
Apply weights to each point and fit a curve along the median of the points.

B-spline
Apply a B-spline to each point and fit a curve along the points.

USAGE

Before using SPLINE, you must first SELECT the line you want smoothed.

The smoothing procedures is non-linear. X variables are assumed to be a sequence of observed values of a function of equally spaced intervals.

SYMBOL

Generates a sequence of polymarkers to identify all the given positions using the current polymarker attributes.

LOCATION

TOOLS

PARAMETERS

locate
Define polymarker locations with the locator device (e.g., mouse).

data
Define polymarker locations with current data specification.

USAGE

Select SYMBOL to draw polymarkers at the specified locations, starting from the first data point and ending at the last data point, using the current polymarker attributes.

When using a locator device to define data points, select beginning and subsequent data points with the left mouse button. Select the ending data point with the middle mouse button.

Note: Specifying data with a locator redefines the current X and Y values.

TEXT

Generates a character string at the specified location.

LOCATION

TOOLS

PARAMETERS

None.

USAGE

Select TEXT to draw text on your graphic. The text starting position is specified by the current mouse position when you click the left mouse button.

A text position box appears, using the default text attributes, allowing you to enter the character string from the keyboard.

The text assumes the current text attributes, as given by the SIGHT state list, when you press the RETURN key.

SIGHT calls the GUS TEXT facility. You may use the same keywords for employing special symbols and expressions. See the GUS chapter.

TEXT ALIGNMENT

Sets the current text horizontal alignment entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

hor-align
Specifies the horizontal alignment (see Table 4-12, below).

TABLE 4-12 Horizontal Alignments

Horizontal Alignment	Description
LEFT JUSTIFIED	Left justify
CENTERED	Center justify
RIGHT JUSTIFIED	Right justify

USAGE

Select TEXT ALIGNMENT to change the text justification of the currently selected text object.

All subsequent text will be drawn using the new horizontal alignment.

TEXT COLOR

Sets the current text color entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

color
Specifies one of the available colors (see Table 4-13, below).

TABLE 4-13 SIGHT Color Chart

Color	Red	Green	Blue
WHITE	1.000	1.000	1.000
BLACK	0.000	0.000	0.000
RED	1.000	0.000	0.000
GREEN	0.000	1.000	0.000
BLUE	0.000	0.000	1.000
CYAN	0.000	1.000	1.000
YELLOW	1.000	1.000	0.000
MAGENTA	1.000	0.000	1.000

USAGE

Select TEXT COLOR to change the text color of the currently selected text object.

All subsequent text will be drawn using the new color.

TEXT DIRECTION

Sets the current text writing direction entry in the SIGHT state list to the value specified by the parameter.

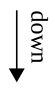
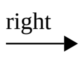
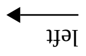
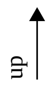
LOCATION

ATTRIBUTES

PARAMETERS

direction
Specifies one of the available text directions (see Table 4-14, below).

TABLE 4-14 Text Directions

Direction	**Description**	**Direction**	**Description**
DOWN		RIGHT	
LEFT		UP	

USAGE

Select TEXT DIRECTION to change the text path of the currently selected text object.

All subsequent text will be drawn using the new text direction.

TEXT FAMILY

Sets the current text font in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

font
Specifies one of the available text fonts native to X Windows (see Table 4-15, below).

TABLE 4-15 Available Fonts

Font Name and Example Output

USAGE

Select TEXT FAMILY to change the text font of the currently selected text object.

All subsequent lines will be drawn using the new text font. All fonts except Symbol come with four styles: normal, bold, italic, and bold italic.

SIGHT TEXT FAMILY uses Motif and X-Window dependent fonts. These fonts are different from those described in the GKS chapter, and given in Appendix B.

If you prefer to use the GKS device independent fonts, you must issue GKS commands from the SIGHT command line to both define the text attribute and plot the text. You cannot use GKS commands to alter text plotted using SIGHT commands.

Note: The available text fonts are Motif and X-Window dependent and may not be available on your particular system.

TEXT SIZE

Sets the current text size entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

size
Specifies one of the available text sizes (see Table 4-16, below).

TABLE 4-16 Text Sizes

Size	Description
8	8 points
10	10 points
12	12 points
14	14 points
18	18 points
24	24 points

USAGE

Select TEXT SIZE to change the text size of the currently selected text object.

All subsequent text will be drawn using the new size.

TEXT STYLE

Sets the current text style entry in the SIGHT state list to the value specified by the parameter.

LOCATION

ATTRIBUTES

PARAMETERS

style
Specifies one of the available text styles (see Table 4-17, below).

TABLE 4-17 Text STyles

Style	**Description**
Normal	Normal Text
Bold	**Bold** text
Italic	*Itali*c text
Bold Italic	Bold italic text

USAGE

Select TEXT STYLE to change the text style of the currently selected text object.

All subsequent text will be drawn using the new style.

WRITE DATA TO FILE

Records data in a file.

LOCATION

FILE

PARAMETERS

file-name
Specifies the name of the file to be written.

USAGE

Select WRITE DATA TO FILE to write a data specification to a file. Cntl. R is the keyboard equivalent for this command.

You can use the directories box to:

- Select a data file from the current directory.
- Select from another directory.

You can add wild cards in the selection box and click on the Filter button to:

- Specify a wild card selection (e.g., *.drawing) in the current directory.
- Specify a wild card selection in another directory.

When you select your data file, click the OK button to write to the file. Otherwise, click the CANCEL button to abort the command.

By default, SIGHT uses the current directory and displays files with the .DAT suffix.

CHAPTER 5

Graphics Utility System Command Language

Table of Contents

Table of Contents continued

Graphics Utility System

GLI utilizes its own portable graphics utilities, the Graphics Utility System (GUS), to automate and complement the Graphical Kernel System (GKS). A single GUS instruction issues multiple GKS commands which simplify complex graphics applications.

GUS is layered on an ISO-conformant GKS implementation of level 0b or higher.

GUS offers many high level graphic utilities and data handling subroutines:

- Two- and three-dimensional Line Graphs;
- Histograms:
- Scatter Diagrams;
- Error Bars;
- Bar Charts;
- 3-D Surfaces:
- Contours;
- Linear or logarithmic axes and grids;
- Automatic 2-D and 3-D axis scaling;
- Logarithmically-scaled windows;
- Automatic legend generation;
- Technical text output;
- Curve smoothing, interpolation, and data filtering;
- Statistical analysis, least square fits, and robust straight line fits;
- Menu-driven chart generation;
- Menu-driven data filtering;
- Three-dimensional transformations.

GUS AUTOPLOT

Invokes the GUS AUTOPLOT interface.

FORMAT

GUS AUTOPLOT

PARAMETERS

None

DESCRIPTION

AUTOPLOT is a form-driven graphics plotting utility which allows you to quickly and easily plot numeric data files in the form of linear or logarithmic presentation quality charts.

`GUS AUTOPLOT` loads a blank chart menu on the screen.

To create a simple chart you merely:

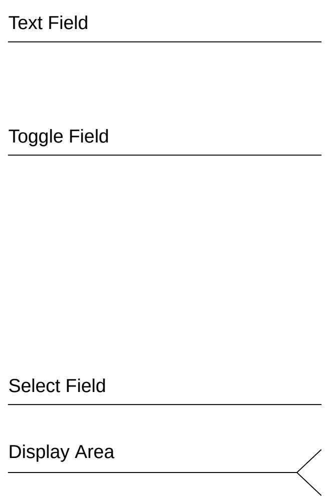
1. Specify a data file to be plotted;
2. Fill in other menu options (axes, annotations, legend labels) or use the defaults;
3. Press PF1 (ESC +1) to plot and PF4 (ESC + 4) to exit.

Note: Refer to the AUTOPLOT section in chapter 2 for more information.

EXAMPLE

```
gli> gus autoplot
```

FIGURE 5-1 AUTOPLOT Menu



GUS AUTOSCALE_2D

Resets the boundaries of the World Coordinate space according to the current two-dimensional data specifications.

FORMAT

GUS AUTOSCALE_2D *x-data, y-data*

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression;
- A constant;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS AUTOSCALE_2D automatically sets the window so axes and graphs are drawn using a scale appropriate to the data. GUS AUTOSCALE_2D determines the minimum and maximum data values to be graphed and automatically redefines the rectangular portion of the World Coordinate space (the window) accordingly. This command modifies the window (and any axes or graph drawn later) to provide a convenient margin around the graph.

To manually scale the window (and any associated axes or graph), determine the data min. and max. values using CALC MIN and MAX math function for each variable. Then apply these values to the `GKS SET WINDOW` instruction to set the window as required.

GUS AUTOSCALE_2D is a building block that is implicitly invoked by the GUS PLOT command. AUTOSCALE_2D can be used separately to automatically scale the window for customized graphs without invoking GUS PLOT.

EXAMPLE

Create sample data items and assign them to variables X and Y.

```
gli> x := -.9, 1.5, 1.75, 2.1, 2.6
gli> y := 1, 1.6, 1.95, 2.05, 2.1
```

GUS AUTOSCALE_2D scales the window properly for the range of data.

```
gli> gus autoscale_2d x y
```

GUS utilities are used to automatically draw the axes and curve.

```
gli> gus axes_2d
gli> gus polyline x y
```

GUS AUTOSCALE_3D

Resets the boundaries of the World Coordinate space according to the current three-dimensional data specifications.

FORMAT

GUS AUTOSCALE_3D *x-data, y-data, z-data*

PARAMETERS

x-data, y-data, z-data
Data specification. A data specification may be:

- An arithmetic expression;
- A constant;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS AUTOSCALE_3D automatically sets the window so axes and graphs are drawn using a scale appropriate to the data. GUS AUTOSCALE_3D determines the minimum and maximum data values to be graphed and automatically redefines the rectangular portion of the World Coordinate space (the window) accordingly. This command modifies the window (and any axes or graph drawn later) to provide a convenient margin around the graph.

GUS AUTOSCALE_3D assumes that the X, Y and Z values are gridded (e.g. the number of Z values is equal to the number of X values times the number of Y values). If the X, Y and Z values are not gridded, use the GRIDIT routine (refer to chapter 3).

As an alternative to this utility, manually scale the window (and any subsequent axes or graph) by determining the minimum and maximum data values using MIN and MAX math function for each variable. Then apply these values to the GKS SET WINDOW instruction to set the window as required.

RESTRICTION

Assumes values for X, Y and Z are gridded. Use the GRIDIT routine for non-gridded data.

GUS AUTOSCALE_3D continued

EXAMPLE

In this example, sample gridded data is created and assigned to variables X, Y and Z.

```
gli> x := -1(.1)1
gli> y := sin(x)
gli> z = exp(-x**2-y**2)
```

GUS AUTOSCALE_3D scales the window properly for the range of data. GRIDIT grids the data for graphing.

```
gli> gridit x y z
gli> gus autoscale_3d x y z
```

GUS utilities are used to automatically draw the 3-D axes and a 3-D curve from the data assigned to X, Y and Z.

```
gli> gus axes_3d
gli> gus surface x y z
```

GUS AXES_2D

Draws X and Y coordinate axes with linearly and/or logarithmically spaced tick marks.

FORMAT

GUS AXES_2D *x-tick, y-tick, x-org, y-org, major-x, major-y, tick-size*

PARAMETERS

x-tick, y-tick
The interval between minor tick marks on each axis. An interval less than or equal to 1 implies that axes labels will be added using either scientific notation (i.e., 1.1E -5) or floating point format (i.e., 0.001), whichever format is shortest. An interval greater than 1 indicates axes labels will use exponential notation.

x-org, y-org
The world coordinates of the origin (point of intersection) of the X and Y axes.

major-x, major-y
Unitless integer values specifying the number of minor tick intervals between major tick marks. Values of 0 or 1 imply no minor ticks. Negative values specify no labels will be drawn for the associated axis.

tick-size
The length of minor tick marks specified in a normalized device coordinate unit. Major tick marks are twice as long as minor tick marks. A negative value reverses the tick marks on the axes from inward facing to outward facing (or vice versa).

DESCRIPTION

GUS AXES_2D draws X and Y coordinate axes with linearly or logarithmically spaced tick marks. Tick marks are positioned along each axis so that major tick marks fall on the axes origin (whether visible or not). Major tick marks are labeled with the corresponding data values. Command parameters for this utility may optionally be omitted.

Axes are drawn according to the scale of the window (Refer to GKS SET WINDOW in chapter 6). Therefore, if little is known about the data set to be plotted, it is suggested that the GUS AUTOSCALE_2D be used to scale the window according to the data before GUS AXES_2D is invoked. Axes and labels appropriate for the data will then be added automatically.

Axes and tick marks are drawn using solid lines. The default axes line color and width can be modified using GKS SET PLINE commands.

GUS AXES_2D draws axes according to the linear or logarithmic transformation established by the GUS SET SCALE command. The default axes transformation is linear.

RESTRICTIONS

Minor tick marks are not labeled. If logarithmic axes are used, the GKS SET WINDOW minimum value must be greater than 0 in the direction of the log axes.

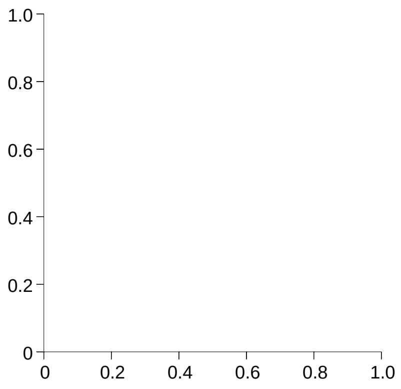
GUS AXES_2D continued

EXAMPLE 1

Draw standard left and bottom linear axes using default values.

```
gli> gus axes_2d
```

FIGURE 5-2 Default 2-D Axes Style

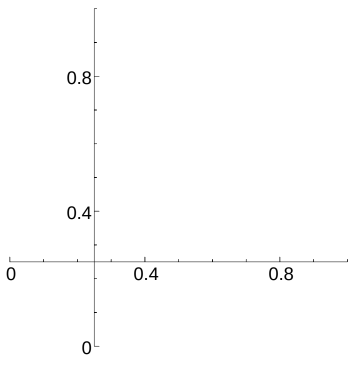


EXAMPLE 2

Draw linear axes intersecting at a specified origin with modified tick marks.

```
gli> gus axes_2d 0.1 0.1 0.25 0.25 4 4 0.005
```

FIGURE 5-3 Customized 2-D Axes



EXAMPLE 3

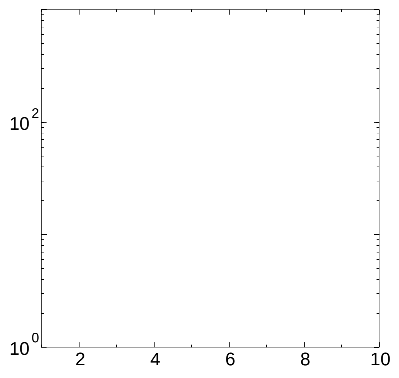
Draw box style, log/linear axes with modified tick marks. Scale the window arbitrarily to better fit the log axes.

```
gli> gks set window 1 10 1 1000
```

Set the transformation to logarithmic for the Y axis and draw the customized axes.

```
gli> gus set scale y_log
gli> gus axes_2d 1 100 1 1 2 2 0.005
gli> gus axes_2d 1 100 10 1000 -2 -2 -0.005
```

FIGURE 5-4 Box Style Log/Linear 2-D Axes



Note: Refer to GUS SET SCALE for linear and logarithmic transformations.

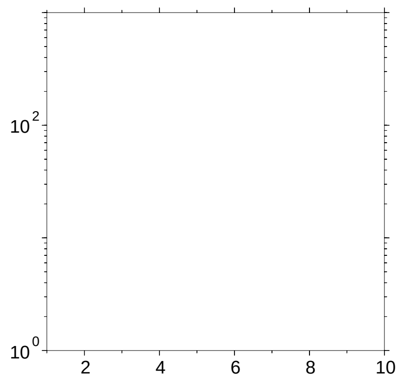
GUS AXES_2D continued

EXAMPLE 4

Draw box style axes with outwardly pointing tick marks. Draw the axes with outward facing tick marks.

```
gli> gus axes_2d 1 100 10 1000 -2 -2 0.005
gli> gus axes_2d 1 100 1 1 2 2 -0.005
```

FIGURE 5-5 Outward Facing Tick Marks



Note: For top and right axes, a negative tick size parameter draws inward facing tick marks. For bottom and left axes, a negative tick size parameter draws outward facing tick marks.

EXAMPLE 5

Draw axes with scientific notation on the y-axis. Modify the default window to a size requiring scientific notation on the y-axis. **Note**: The size of the GKS window determines the axes endpoints. AUTOSCALE_2D automatically scales the window and, therefore, the axes endpoints.

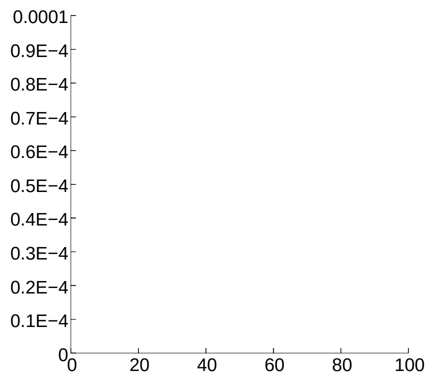
```
gli> gks set window 0 100 0 0.0001
```

Draw axes with ***x-tick*** parameter more than 1, indicating a linear X axis, and with ***y-tick*** parameter less than 1, indicating a Y axis with scientific or floating point notation, whichever is shorter.

```
gli> gus axes_2d 20, 0.00001, 0, 0, 1, 1, 0.005
```

GUS AXES_2D continued

FIGURE 5-6 Axes with Scientific Notation



GUS AXES_3D

Draws X, Y and Z labeled coordinate axes with linearly and/or logarithmically spaced tick marks.

FORMAT

GUS AXES_3D ***x-tick, y-tick, z-tick, x-org, y-org, z-org, major-x, major-y, major-z, tick-size***

PARAMETERS

x-tick, y-tick, z-tick
The interval between minor tick marks for each axes. An interval less than or equal to 1 implies that axes labels will be added using either scientific notation (i.e., 1.1E -5) or floating point format (i.e., 0.001), whichever format is shortest. An interval greater than 1 indicates axes labels will use exponential notation.

x-org, y-org, z-org
The world coordinates of the origin (point of intersection) of the X, Y and Z axes.

major-x, major-y, major-z
Unitless integer values specifying the number of minor tick intervals between major tick marks. Values of 0 or 1 imply no minor ticks. Negative values specify no labels will be drawn for the associated axis.

tick-size
The length of the minor tick marks expressed in a normalized device coordinate unit. Major tick marks are twice as long as minor tick marks. A negative value reverses the tick marks on the axes from inward facing to outward facing (or vice versa).

DESCRIPTION

GUS AXES_3D draws labeled X, Y and Z coordinate axes with linearly or logarithmically spaced tick marks. Tick marks are positioned along each axis so that major tick marks fall on the axes origin (whether visible or not). Command parameters for this utility may optionally be omitted.

Axes are drawn according to the scale of the window (Refer to GKS SET WINDOW in chapter 6). Therefore, if little is known about the data set to be plotted, it is suggested that the GUS AUTOSCALE_3D be used to scale the window according to the data before GUS AXES_3D is invoked. Axes and labels appropriate for the data will then be added automatically.

Axes and time marks are drawn using solid lines. GKS SET PLINE commands can be used to modify the default axes line color and width.

GUS AXES_3D draws axes according to the linear or logarithmic transformation established by the GUS SET SCALE command. The GUS SET SPACE instruction defines the rotation and tilt (viewing angle) of the axes and sets the Z-axis minimum and maximum values for the window.

RESTRICTIONS

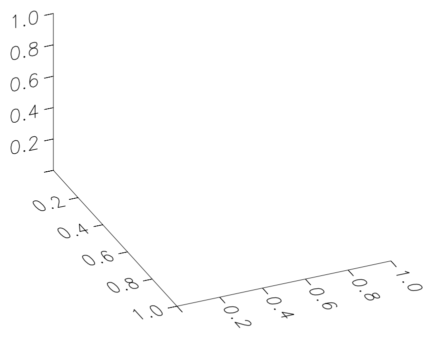
Minor tick marks are not labeled. If logarithmic axes are used, the GKS SET WINDOW minimum value must be greater than 0 in the direction of the log axes.

EXAMPLE 1

This example draws standard X, Y and Z axis using the default values for GUS AXES_3D. GUS SET SPACE defines the minimum and maximum values for the window in the Z direction.

```
gli> gus set space 0 1
gli> gus axes_3d
```

FIGURE 5-7 Default 3-D Axes Style



Note: If the axes in EXAMPLES 1 or 2 were used to plot actual data values, GUS SET SPACE would generally be assigned parameters based on the Z values of the data to be graphed. The user could invoke MIN and MAX math functions to determine the actual Z minimum and maximum values to be input as parameters to GUS SET SPACE.

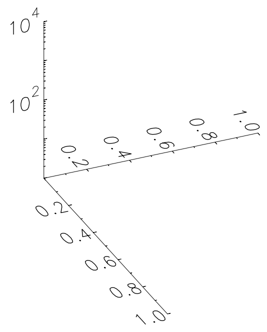
GUS AXES_3D continued

EXAMPLE 2

Draw linear 3-D axes intersecting at a specified origin with modified tick marks. GUS SET SPACE defines the minimum and maximum values for the Z axis. GUS SET SCALE specifies a logarithmic axis.

```
gli> gus set space 1 10000
gli> gus set scale z_log
gli> gus axes_3d 0.1 0.1 10 0 0 1 2 2 2 0.005
```

FIGURE 5-8 Customized 3-D Axes



EXAMPLE 3

Draw axes with scientific notation and floating point label format. Since the default parameter for the Z axis is a range of 0 to 1, change this range using the GUS SET SPACE instruction to a size where scientific notation is appropriate. **Note**: The size of the GKS window determines the axes endpoints. AUTOSCALE_3D automatically scales the window according to the data and, therefore, scales the axes endpoints.

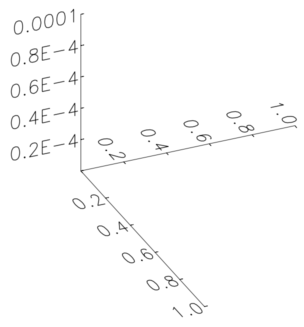
```
gli> gus set space 0 0.0001
```

Draw axes with ***x-tick***, ***y-tick*** and ***z-tick*** parameters less than 1, indicating that the shorter format of floating point or scientific notation labels will be applied to the axis.

```
gli> gus axes_3d 0.2, 0.2, 0.00002, 0, 0, 0, 1, 1, 1, 0.005
```

GUS AXES-3D continued

FIGURE 5-9 3-D Axes with Scientific Notation



GUS BAR_GRAPH

Draws a standard Bar Chart.

FORMAT

GUS BAR_GRAPH *x-data, y-data*

PARAMETERS

x-data, y-data
Data elements, including:

- An array;
- An arithmetic expression;
- The name of a variable(s) or function(s);
- A sub-range or range specification.

DESCRIPTION

GUS BAR_GRAPH draws bars of a height equal to the values of Y at points on an axis specified by the values of X. The labeled axes for this utility are drawn using GUS AXES_2D. Line color and width are controlled using GKS SET PLINE commands.

GUS BAR_GRAPH automatically fills bars using the values contained in the GKS state list. Fill patterns and colors can be specified using GKS SET FILL commands. (Refer to chapter 6 for GKS commands.)

RESTRICTION

This utility assumes X values are evenly spaced.

EXAMPLE

Create sample data items and assign them to variables X and Y.

```
gli> x := 2, 4, 6, 8, 10
gli> y := 0, 2.4, 5.26, 4.4 0
```

GUS AUTOSCALE 2_D scales the window properly for the data.

```
gli> gus autoscale_2d x y
```

Fill the bars with a GLI hatch (index 2) style. **Note**: GLI also allows hardware (output device) patterns to be specified. (Refer to GKS SET FILL_STYLE in chapter 6.)

```
gli> gks set fill int_style hatch
gli> gks set fill style 2
```

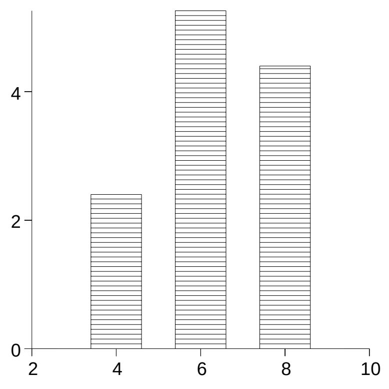
Use the GKS SET FILL INT_STYLE command

Draw default-style axes and plot the data using the GUS BAR_GRAPH utility.

```
gli> gus axes_2d
gli> gus bar_graph x y
```

GUS BAR_GRAPH continued

FIGURE 5-10 Bar Graph



GUS COLORMAP

Draws the current colormap and displays data values corresponding to the colormap shades.

FORMAT

GUS COLORMAP

PARAMETERS

None

DESCRIPTION

GUS COLORMAP displays the current colormap (see GUS SET COLORMAP) in the last opened graphics window. If a window is not opened, a default window and viewport will be created and the colormap will be drawn. The COLORMAP utility controls the shading of surfaces.

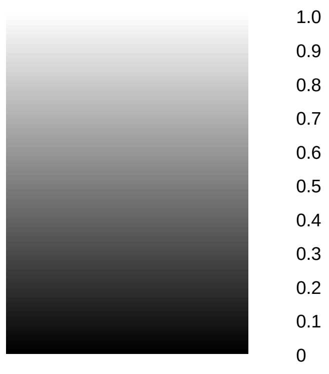
Labels corresponding to the intensity levels of the colormap are displayed to the right of the colormap, provided the viewport is sized to prevent the labels from being clipped. When used in conjunction with GUS SURFACE, the labels correlate to the Z data given.

EXAMPLE

This example changes the default colormap to GRAYSCALE and draws it in a window with the viewport set so labels are displayed.

```
  gli> gus set colormap grayscale
  gli> gks set viewport 0.2, 0.7, 0.2, 0.9
  gli> gus colormap
```

FIGURE 5-11 Colormap



GUS SET COLORMAP

Sets the colors used by the SURFACE utility uses to draw 3-D surfaces or filled contours.

FORMAT

GUS SET COLORMAP *colormap*

PARAMETERS

colormap

TABLE 5-1 Colormap Parameters

PARAMETERS	DESCRIPTION
Uniform	Regularly distributed (3° rotation) colormap from the HLS color-wheel with 120 colors assigned.
Temperature	Represents colors commonly used to display temperatures. Higher data values are assigned red shades and lower values blue shades.
Grayscale	Arbitrarily distributed shades of gray. (Useful for laser printer output.)
Glowing	Arbitrarily distributed colors commonly used for light source shading.
Rainbow	Arbitrarily distributed colors representing colors of a rainbow.
Geologic	Represents colors commonly used to display map data. Higher data values are assigned brown, lower values are blue and intermediate values are assigned green shades to represent vegetation.
Greenscale	Evenly distributed colors from dark (lower values) to light (higher values) green.
Cyanscale	Evenly distributed colors from dark (lower values) to light cyan (higher values).
Bluescale	Evenly distributed colors from dark (lower values) to light blue (higher values).
Magentascale	Evenly distributed colors from dark (lower values) to light magenta (higher values).
Redscale	Evenly distributed colors from dark (lower values) to light red (higher values) .
Flame	Arbitrary distributed colors representing colors of fire.

DESCRIPTION

GUS SET COLORMAP determines the colors from which GUS SURFACE uses to draw shaded 3-D surfaces or filled contours. Color maps can be selected to fit the data types displayed. UNIFORM is the default colormap.

Colors are translated into shades of gray when displayed on grayscale devices. The grayscale translation formula is:

Gray Intensity = 0.3 * Red-intensity + 0.59 * Green-intensity + 0.11 * Blue-intensity

Note: Changing GKS color attributes will not affect colormaps.

GUS SET COLORMAP continued

RESTRICTION

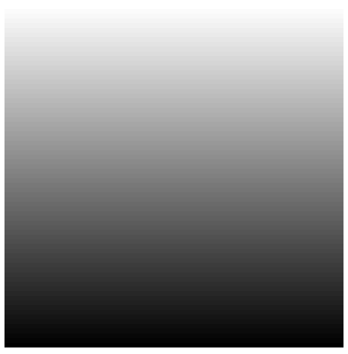
Color maps can not be changed by the user.

EXAMPLE

Change the color map to use the colors assigned to the parameter GRAYSCALE and draw a colormap.

```
gli> gus set colormap grayscale
gli> gus colormap
```

FIGURE 5-12 Grayscale Color Map



GUS CONTOUR

Draws contours of a function of two variables (f(x,y)) whose values are specified over a rectangular mesh. Contour lines may optionally be labeled.

FORMAT

GUS CONTOUR *x-data, y-data, h-data, z-data [, dimension_x, label-count]*

PARAMETERS

x-data, y-data, z-data
X-data and **y-data** specifications defining a grid. **Z-data** is a singly dimensioned array containing at least x-data * y-data points. Z-data describes height [or f(x, y)] at each point on the X, Y grid. Data for GUS CONTOUR is ordered as shown in the figure on the following page.

h-data
Contour height (interval) data specification. If ***h-data*** is equal to one (1), GUS CONTOUR automatically calculates 16 height intervals equally distributed among the range of Z values.

dimension_x
When graphing a subset of the data, this parameter must be set to a value equal to the number of data items in the X direction so the data will be graphed in proper sequence. This option is not required when graphing the entire data set.

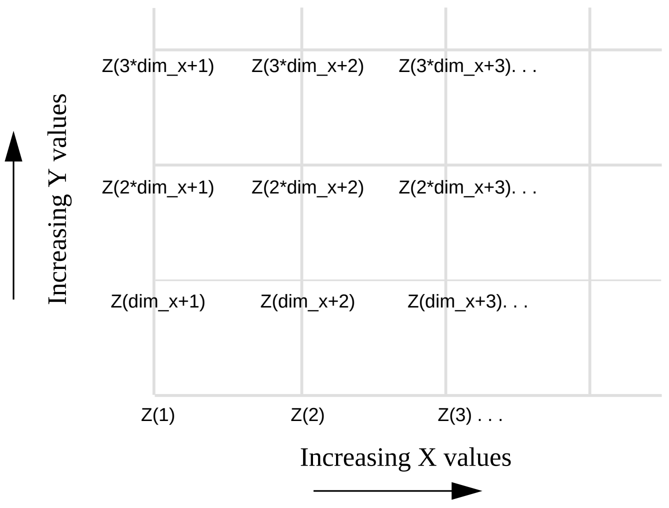
label-count
Directs GLI to label contour lines. For example, a value of 3 would label every third line. A value of 1 will label every line. A value of 0 (default) produces no labels. If labeling is invoked, un-labeled lines will be drawn using a dashed linetype and labeled lines will use a solid linetype.

DESCRIPTION

GUS CONTOUR plots the function of two variables as contour lines. The function of two variables is also referred to as a field or a 2-D function.

GLI supports two types of contour applications: *gridded* and *random* contouring. The difference between these applications is the type of input data. GUS CONTOUR is used with a field that is highly ordered. The field to be contoured is divided into a mesh, or grid, in both the X and Y directions. This grid must be linear (spacing between grid lines is always equal) in each direction, but the spacing on the X and Y axes may be different. The figure below illustrates a sample grid.

FIGURE 5-13 GUS CONTOUR Data Ordering



Random contouring is used with data that is arbitrarily distributed over a 2-D field. Thus, a functional value, Z, is defined for every location provided, and any number of such locations may be defined. Unlike the gridded field where the location of every point is well defined, the location of points in the resultant data sets is arbitrary and is thus defined to be random. Random data must be gridded before being input to GUS CONTOUR. Refer to the GRIDIT utility in chapter 3 for random contouring.

The specific intervals at which the contours are to be drawn can be specified or they can be set to the default interval of 16. GLI supports colored contour lines and filled (color or grayscale) contours.

Contour line style, width, and color are controlled using GKS SET PLINE commands. Labeled and non-labeled contour lines are drawn as solid lines. When labeled and non-labeled contour lines are drawn on the same graph, non-labeled lines are drawn with a dashed line style.

For optimum labeling, use the GLI software fonts with stroke precision (see GKS SET TEXT FONTPREC). The GKS SET TEXT FONTPREC *dimension-x* parameter must be set equal to the size of the GUS CONTOUR *x-data* parameter.

Axes can be added using the GUS AXES_2D instruction. GUS SET SPACE defines the tilt and rotation of the contour graph.

For filled and colored contours, refer to the GUS SURFACE command. To add legends to filled contour plots, see GUS COLORMAP.

RESTRICTION

GUS CONTOUR accepts only regularly-spaced (gridded) data values. For irregularly-spaced data, refer to GRIDIT in Chapter 3. GUS CONTOUR must not exceed a maximum array size of 2×10^3 data points in the X or Y directions.

EXAMPLE

For this example, create data ranges and arithmetic expressions to provide GUS CONTOUR with the sample regularly-spaced data items. The height interval (H) is set to arbitrary intervals of 0.2. **Note:** If H was set equal to 1, 16 intervals would automatically be computed and drawn within the range of Z values.

```
gli> x := -3(0.15)3
gli> y := x
gli> h := 0(0.2)2.8
gli> z = exp(sin(y)*cos(x**2))
```

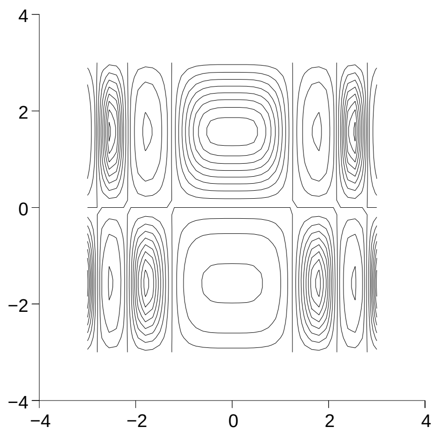
To make sure the drawing transformation is set for world coordinates use the SET XFORM command. The SET WINDOW instruction creates the window for the graph and, therefore, defines the endpoints of the axes. The SET SPACE command scales the window for the Z data and sets the tilt and rotation of the graph. Since the default tilt and rotation is 60° and 60°change to 0° and 90°, respectively.

```
gli> gks set xform wc
gli> gks set viewport 0.15, 0.95, 0.15, 0.95
gli> gks set window -4, 4, -4, 4
gli> gus set space 0 3 0 90
```

GUS CONTOUR draws the graph with an axis.

```
gli> gus axes_2d
gli> gus contour x, y, h, z
```

FIGURE 5-14 Contour Graph



EXAMPLE TWO

In this example, contour data from a file using labeled contour lines. First, define the coordinate transformation, viewport and window.

```
gli> gks set xform wc
gli> gks set viewport 0.05 0.95 0.05 0.95
gli> gks set window 1 100 1 100
```

Read in the data from a file and create x and y coordinates that correspond to the z data. Define the data ranges (2800 and 4400) and the interval (100) at which the contour lines are to be drawn. **Note**: use the PRINT MIN () MAX () command to determine the minimum and maximum values for x and y.

```
gli> read GLI_DEMO/xdemo5.dat
gli> x := 1..100
gli> y := 1..100
gli> h := 2800(100)4400
```

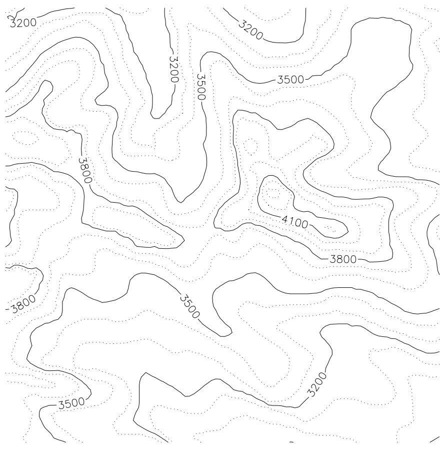
Set the text font (using stroke font -1, Cartographic Roman), text height and the viewing space.

```
gli> gks set text fontprec -1 stroke
gli> gks set text height 0.015
gli> gus set space 2800 4400 0 90
```

Draw the contours, labelling every third contour.

```
gli> gus contour x y h z 100 3
```

FIGURE 5-15 Contour Graph



This page intentionally left blank.

GUS CURVE

Draws a simulated three-dimensional curve from the data given.

FORMAT

GUS CURVE *x-data, y-data, z-data [primitive]*

PARAMETERS

x-data, y-data, z-data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

primitive
Optional output primitive (POLYLINE, POLYMARKER).

DESCRIPTION

GUS CURVE draws the 3-D curve using the current polyline attributes in the GKS state list. Line type, color and width can be modified using GKS SET PLINE commands.

GUS SET SCALE specifies linear or logarithmic transformations. Appropriate axes are provided using GUS AXES_3D. GUS SET SPACE controls the Z values (height), and rotation and tilt of the axes and curve.

EXAMPLE

For this example, data ranges and arithmetic expressions provide sample values to be graphed.

```
gli> t := 1(.1)40
gli> def fun x 40+t*cos(t)
gli> def fun y 40+t*sin(t)
gli> def fun z t
```

The drawing transformation is set for world coordinates and the viewport is described in normalized device coordinates. The SET WINDOW instructions creates the window for the graph from 1 to 40 (world coordinates) in the X and Y directions.

```
gli> gks set xform wc
gli> gks set viewport 0.15, 0.95, 0.15, 0.95
gli> gks set window 1, 100, 1, 100
```

GUS commands define the window according to the Z values (height) and 3-D axes are drawn.

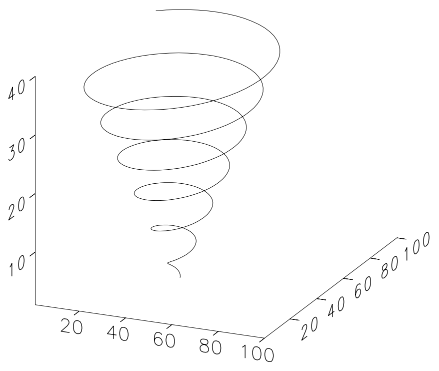
```
gli> gus set space 1,40, 30, 45
gli> gus axes_3d
```

The GUS CURVE command plots the 3-D data on the axes.

```
gli> gus curve x, y, z
```

GUS CURVE continued

FIGURE 5-16 3-D Line Graph



GUS ERROR_BARS

Draws a standard vertical or horizontal Error Bar graph.

FORMAT

GUS ERROR_BARS *x-data, y-data, error-1, error-2, orientation*

PARAMETERS

x-data, y-data
An array of numbers representing the mean of the values to be plotted.

error-1
The absolute value of the negative deviation at the given point.

error-2 CONTOUR
The absolute value of the positive deviation at the given point.

orientation
Using keywords, VERTICAL or HORIZONTAL, this parameter defines the alignment of the error bars at the given points. The default orientation is vertical.

DESCRIPTION

This utility draws a standard error bar graph with either vertical or horizontal error bars representing the range of values at the given points. The default polymarker type and color representing each point can be changed using GKS SET PMARK commands. GUS SET SCALE instructions can be used to define a logarithmic transformation. The GUS POLYLINE utility can be used to connect the X and Y values.

RESTRICTION

The array size for X-DATA, Y-DATA, ERROR-1 and ERROR-2 must be the same size.

EXAMPLE

Define sample data items.

```
gli> x := 1, 2, 3, 4
gli> y := 0.67, 0.81, 0.52, 0.43
gli> e1 := 0.64, 0.8, 0.43, 0.41
gli> e2 := 0.69, 0.88, 0.54, 0.49
```

Set the transformation, window and viewport for the graph.

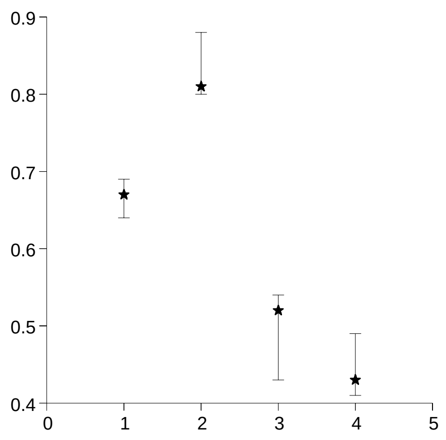
```
gli> gks set xform wc
gli> gks set viewport 0.15, 0.95, 0.15, 0.95
gli> gks set window 0, 5, 0.4, 0.9
```

Draw default 2-D axes and plot the Error Bars.

```
gli> gus axes_2d
gli> gus error_bars x, y, e1, e2, vertical
```

GUS ERROR_BARS continued

FIGURE 5-17 Error Bar Graph



GUS FFT

Computes and draws the Fast Fourier Transformation of a real number sequence.

FORMAT

GUS FFT *data*

PARAMETERS

data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS FFT computes and plots the Fast Fourier Transformation using the current polyline attributes in the GKS state list. The FFT polyline type, color and width can be controlled using GKS SET PLINE commands.

If logging (GUS SET LOG) is enabled, GUS FFT creates a two-column output file named GUS.LOG containing the Fourier coefficients. GUS.LOG is located in the current directory.

Note: Since GUS FFT transforms the data as it is being graphed, invoking AUTOSCALE_2D before this command will not scale the window properly. Therefore, the window should be scaled manually using a GKS SET WINDOW instruction. However, if you elect to use AUTOSCALE_2D, save the transformed data using GUS SET LOG and re-graph the data.

EXAMPLE

This example reads the GLI demonstration file, demo15.dat, into variables X and Y. **Note:** To include this data file, be sure to enter ‘GLI_DEMO’ in uppercase characters enclosed with single quote marks.

```
gli> read ’GLI_DEMO’demo13_1.dat x, y
333 lines read from file demo13_1.dat
```

Use GKS SET WINDOW to set the window to values that better fit the transformed data.

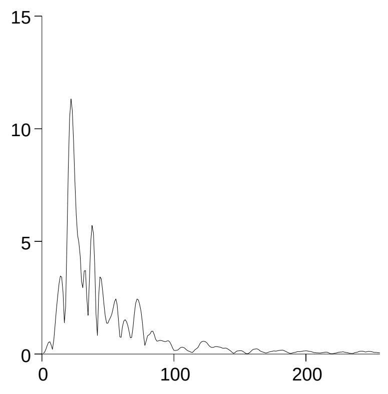
```
gli> gks set window 0 256 0 15
```

GUS FFT continued

Draw the axes, create the LOG file, and plot the Fast Fourier Transform of Y. Then turn off the LOG file saving the transformed data.

```
gli> gus axes_2d
gli> gus set log
gli> gus fft y
gli> gus set nolog
```

FIGURE 5-18 FFT Plot



GUS GRID

Draws a linear and/or logarithmic grid.

FORMAT

GUS GRID *x-tick, y-tick, x-org, y-org, major-x, major-y*

PARAMETERS

x-tick, y-tick
The length in world coordinates of the interval between minor grid lines.

x-org, y-org
The world coordinates of the origin (point of intersection) of the grid.

major-x, major-y
Unitless integer values specifying the number of minor grid lines between major grid lines. Values of 0 or 1 imply no grid lines.

DESCRIPTION

GUS GRID draws grid lines where tick marks are positioned along each axis. Major grid lines correspond to the axes origin and major tick marks whether visible or not. Minor grid lines are drawn at points equal to minor tick marks.

All command parameters may optionally be omitted.

Major grid lines are drawn using solid lines and minor grid lines are drawn using dashed lines. Grid line color and width can be specified using GKS SET PLINE commands. The GUS SET SCALE command is used to set linear and/or logarithmic grids.

RESTRICTION

If a logarithmic grid is specified, the GKS SET WINDOW minimum value must be greater than 0 in the direction of the log grid. Negative values are not allowed as parameters for GUS GRID.

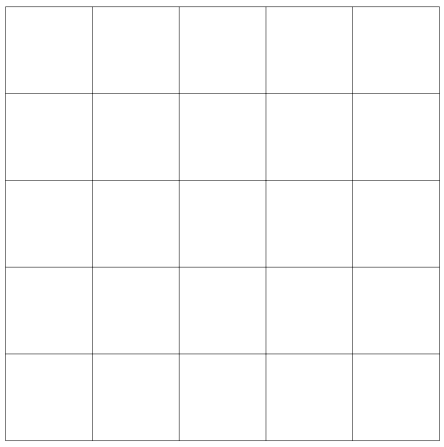
GUS GRID continued

EXAMPLE 1

Draw a standard grid using default values.

```
gli> gus grid
```

FIGURE 5-19 Default Grid Style

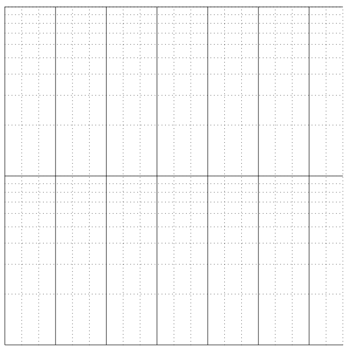


EXAMPLE 2

Draw a log/linear grid with a customized grid. GUS SET SCALE designates a logarithmic Y axis, and GUS SET WINDOW arbitrarily scales the window for the Y log axis.

```
gli> gks set window 0 20 1 100
gli> gus set scale y_log
gli> gus grid 1 10 0 1 3 3
```

FIGURE 5-20 Customized Grid Style



GUS HISTOGRAM

Draws a histogram with up to forty cells.

FORMAT

GUS HISTOGRAM *x-data*

PARAMETERS

x-data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS HISTOGRAM generates a bar diagram using the current attributes in the GKS state list in which the height of each bar represents the number of occurrences of a data set within a range of values.

A number of options are available to modify the appearance of the histogram, thereby increasing the effectiveness of the data presentation. Use the GUS GRID utility to specify a background grid. Line width, color and type are controlled using GKS SET PLINE instructions. The fill pattern or color of each cell is controlled by GKS SET FILL commands. Text is added using GKS TEXT instructions. (Refer to chapter 6 for GKS commands.)

Axes are drawn for HISTOGRAM using the GUS AXES_2D utility.

If logging (SET LOG) is enabled, this utility creates a two-column output file (GUS.-LOG) in the current directory containing the sample mean, the sample variance, the number of cases used to calculate them, and the cell statistics for every data set.

EXAMPLE

For this example, sample data items are created and assigned to variables X and Y.

```
gli> x := -1(.1) 1
gli> y = x
```

Scale the window according to the data and draw default-style axes.

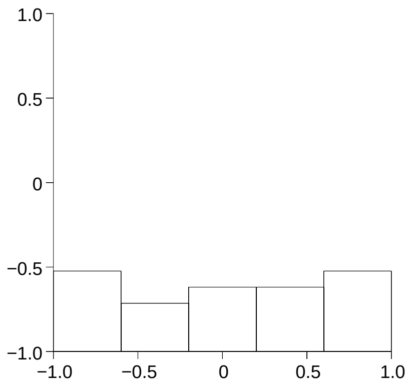
```
gli> gus autoscale_2d x y
gli> gus axes_2d
```

GUS HISTOGRAM continued

Plot the data using the GUS HISTOGRAM utility.

```
gli> gus histogram x
```

FIGURE 5-21 Histogram



GUS INVERSE_FFT

Computes and draws the inverse Fast Fourier Transformation of a real valued sequence.

FORMAT

GUS INVERSE_FFT *x-data y-data*

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a dynamic variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS INVERSE_FFT computes and plots the Inverse Fourier transformation using the current polyline attributes in the GKS state list. Line width, color and style can be controlled using GKS SET PLINE commands.

If logging (SET LOG) is enabled, GUS INVERSE_FFT creates a two-column output file named GUS.LOG containing the Inverse Fourier coefficients. GUS.LOG is located in the current directory.

EXAMPLE

This example reads the GLI demonstration file (demo13_1) into variables X and Y. **Note:** To include this data file, be sure to enter 'GLI_DEMO' in uppercase characters enclosed with single quote marks.

```
gli> read 'gli_demo'demo13_1.dat x, y
21 lines read from file demo13_1.dat
```

Use GUS AUTOPLOT (or its abbreviation, GUS PLOT) utility to scale the window, draw the axes and plot the data.

```
gli> gus plot x y
```

Plot the Inverse FFT of the data as a second curve and save the coefficients to the GUS.-LOG file.

```
gli> gus set log
gli> gus inverse_fft x, y
```

GUS LINFIT

Generates a robust Straight-Line fit, starting from the first data point and ending at the last data point.

FORMAT

GUS LINFIT *x-data y-data*

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS LINFIT uses a robust statistical estimator to avoid undesired sensitivity to outlying data points. The line is fitted through the given world coordinate points, in the order specified, using the current polyline attributes in the GKS state list. Use GKS SET PLINE commands to control the line style, width and color.

All points are transformed and/or smoothed using the current values for GUS SET SCALE and GUS SET SMOOTHING.

EXAMPLE

Create sample data items and assign them to variables X and Y.

```
gli> x := .1, .2, .3, .6, .8
gli> y := .5, .1, .8, .2, .4
```

Use GUS LINFIT to plot the data and apply 2-D axes.

```
gli> gus linfit x, y
gli> gus axes_2d
```

GUS LINREG

Generates a Linear Least-Square fit, starting from the first data point and ending at the last data point.

FORMAT

GUS LINREG *x-data y-data*

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression.
- The name of a dynamic variable or function.
- A sub-range or range specification.

DESCRIPTION

GUS LINREG fits a line through the given world coordinate points in the order specified, using the current polyline attributes in the GKS state list. Line style, color and width are controlled by GKS SET PLINE commands.

All points are transformed and/or smoothed using the current values for GUS SET SCALE and GUS SET SMOOTHING.

EXAMPLE

For this example create same data items and assign them to variables X and Y.

```
gli> X := .1, .2, .3, .6, .8
gli> Y := .5, .1, .8, .2, .4
```

Use GUS LINREG to plot the data and add 2-D axes.

```
gli> gus linreg x, y
gli> gus axes_2d
```

GUS SET LOG

Creates a file containing results from the GUS utilities for smoothing, spline, linear regression, inverse FFT or FFT.

FORMAT

GUS SET *[NO]LOG*

PARAMETERS

None

DESCRIPTION

The mathematically transformed data from GUS utilities for smoothing, spline, least square fit, linear regression, FFT, Inverse FFT is saved to a file named GUS.LOG.

EXAMPLE

```
gli> gus set log
.
. ! gus fft
.
gli> gus set nolog
```

GUS PIE_CHART

Draws a pie chart on the screen for the user.

FORMAT

GUS PIE_CHART *data*

PARAMETER

data
the data items for chart slices

DESCRIPTION

GUS PIE_CHART draws a pie chart varying the hatch pattern, style or color automatically for each data value. All data is added together internally in GLI, and then each individual element is converted to a percentage of the total for the drawing of slices. Each slice is labeled according to the data plotted. The chart can be labeled further using GKS TEXT or GUS TEXT commands.

The first slice will be drawn using the default style hatch, pattern or solid specified in the GLI STARTUP file or by the style currently selected by the GKS SET FILL commands for color and style. Additional slices will be filled using the subsequent style and color according to the tables shown in Chapter 6 for the GKS SET FILL commands for AREA, INTERIOR_STYLE, STYLE and COLOR_INDEX. (See chapter 6 for GKS commands)

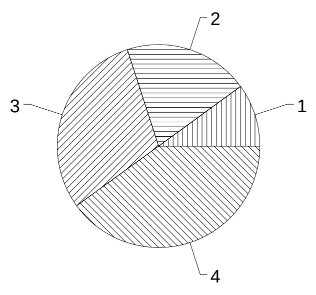
For instance, if you set the fill interior style to solid, and the current fill color is default color 1 (black), the slice representing the second data item will be drawn with red fill (default color 2) or the color defined as color 2 by the GKS SET COLOR index. If the number of data items exceeds the number of colors or pattern or hatch styles, GLI will begin repeating the fill style.

EXAMPLE

Draw a simple pie chart, after creating data items for the chart and setting GLI to use the PATTERN style of fill.

```
gli> x := 1 2 3 4
gli> gks set fill int_style hatch
gli> gus pie_chart x
```

FIGURE 5-22 Pie Chart



GUS PLOT

The PLOT utility produces an XY graph from two-dimensional data.

FORMAT

GUS PLOT *x-data, y-data [, y2-data, y3-data . . .]*

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS PLOT produces a graph with labeled axes, starting from the first data point and ending at the last data point provided. GUS PLOT automatically sets a default window and viewport and invokes GUS AUTOSCALE_2D and GUS AXES_2D to create complete line graphs with a single command.

GUS PLOT uses the current polyline attributes in the GKS state list. To modify line color, style, or width use GKS SET PLINE commands. Linear or logarithmic scales are invoked using GUS SET SCALE. GUS SET SMOOTHING applies a degree (from 1 to 20) of smoothing to the curve. (Refer to chapter 6 for GKS commands.)

Note: Axes are scaled to the limits of the first y-data specification given. Subsequent y-data values are ignored in scaling the axes, which may cause clipping of output.

By first converting polar coordinates to cartesian coordinates, this utility can graph equations described in polar coordinates. The conversion equations are X=R*COSq and Y=R*SINq. (See Example 2).

EXAMPLE 1

Use the GUS PLOT command to automatically generate a complete graph from data created and assigned to variables X, Y and Y2.

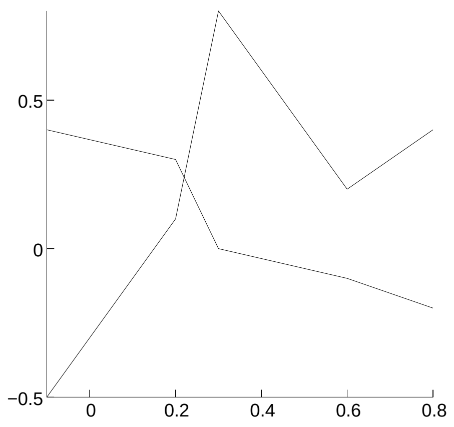
```
gli> x := -.1, .2, .3, .6, .8
gli> y := -.5, .1, .8, .2, .4
gli> y2 := .4, .3, 0, -.1, -.2
```

Produce a line graph with axes and labels using GUS PLOT.

```
gli> gus plot x, y, y2
```

GUS PLOT continued

FIGURE 5-23 Auto Line Graph



EXAMPLE 2

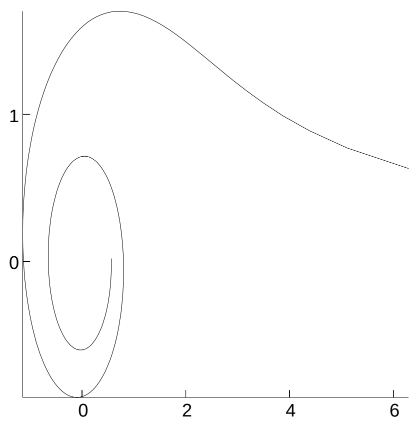
To create a polar graph, first take the data in polar coordinates, convert them to cartesian and assign them to variables, in this case, X and Y.

```
gli> theta = 0.1(0.05)12.6
gli> r = sqrt(4/theta)
gli> x = r*cos(theta)
gli> y = r*sin(theta)
```

Plot using the GUS PLOT command.

```
gli> gus plot x y
```

FIGURE 5-24 Polar Coordinate Graph



GUS POLYLINE

The POLYLINE utility generates a sequence of connected straight lines, starting from the first data point and ending at the last data point.

FORMAT

GUS POLYLINE *x-data, y-data [, y2-data, y3-data . . .]*

PARAMETERS

x-data, y-data [, y2-data, y3-data . . .]
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS POLYLINE draws one or more continuous straight line segments, called a polyline, connecting the world coordinate points in the order specified. GUS POLYLINE uses the current polyline attributes in the GKS state list. Line width, type and color are controlled using GKS SET PLINE commands.

If more than one y-data specification is given, the initial curve is drawn using the current line type in the GKS state list. The line types applied to subsequent curves automatically vary to a maximum of four.

GUS AXES_2D applies axes scaled to the values of the initial y-data specification. Linear and Logarithmic transformations are specified using the GUS SET SCALE command, and smoothing can be applied to the polyline with the GUS SET SMOOTHING instruction.

EXAMPLE

For this example, create sample data items and assign them to variables X and Y.

```
gli> x := .1, .2, .3, .6, .8
gli> y := .5, .1, .8, .2, .4
```

Scale the window according to the data and draw the axes.

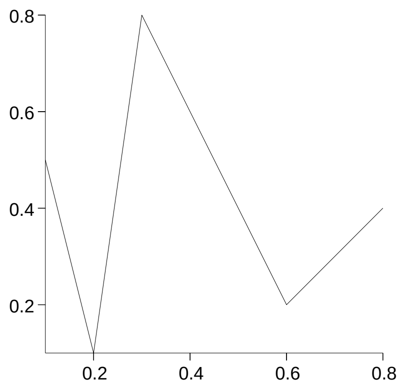
```
gli> gus autoscale_2d x y
gli> gus axes_2d
```

Use GUS POLYLINE to graph the X and Y values using the default polyline attributes.

```
gli> gus polyline x, y
```

GUS POLYLINE continued

FIGURE 5-25 Line Graph



GUS POLYMARKER

The POLYMARKER utility generates a sequence of markers (polymarkers) to identify the given data points.

FORMAT

GUS POLYMARKER *x-data, y-data [, y1-data, y2-data . . .]*

PARAMETERS

x-data, y-data [, y2-data, y3-data . . .]
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GUS POLYMARKER places one or more special symbols, called polymarkers, at the specified world coordinates, using the current polymarker attributes in the GKS state list. Polymarker size, style and color are controlled using GKS SET PMARK commands.

If more than one y-data specification is given, the initial data set is drawn using the default marker type. The marker types applied to subsequent data sets automatically vary to a maximum of four different types.

GUS AXES_2D applies axes to graphs created with this command. Linear and Logarithmic transformations are specified using the GUS SET SCALE command, and smoothing can be applied to the polyline with the GUS SET SMOOTHING instruction.

EXAMPLE

For this example create sample data items and assign them to variables X and Y.

```
gli> x := .1, .2, .3, .6, .8
gli> y := .5, .1, .3, .2, .4
```

Scale the window according to the data and draw the axes.

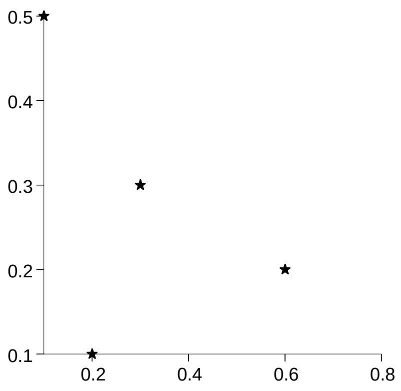
```
gli> gus autoscale_2d x y
gli> gus axes_2d
```

Invoke GUS POLYLINE to graph the X and Y values using the asterisk polymarker type.

```
gli> gks set pmark type asterisk
gli> gus polymark x, y
```

GUS POLYMARKER continued

FIGURE 5-26 Scatter Diagram



GUS SET SCALE

Sets the type of transformation to be used for subsequent GUS output primitives.

FORMAT

GUS SET SCALE *scale [, flip]*

PARAMETERS

scale
Scale specification (see Table 5-2, below).

TABLE 5-2 Log Linear Scale options

Scale	**Description**
LINEAR	Linear axes
X_LOG	Logarithmic X-axis
Y_LOG	Logarithmic Y-axis
XY_LOG	Logarithmic X-, Y-axis
Z_LOG	Logarithmic Z-axis
XZ_LOG	Logarithmic X-, Z-axis
YZ_LOG	Logarithmic Y-, Z-axis
XYZ_LOG	Logarithmic X-, Y-, Z-axes

TABLE 5-3 Flip options

Filp	**Description**
NONE	Normal axes
FLIP_X	Filp X-axis
FLIP_Y	Flip Y-axis
FLIP_XY	Flip X-, Y-axis
FLIP_Z	Flip Z-axis
FLIP_XZ	Flip X-, Z-axis
FLIP_YZ	Flip Y-, Z-axis
FLIP_XYZ	Flip X-, Y-, Z-axes

DESCRIPTION

GUS SET SCALE defines the current transformation type to the value specified by the parameter. GUS uses this value for all subsequent GUS output primitives until another value us provided.

The transformation scale is used to transform points from an abstract logarithmic or semi-logarithmic coordinate system into the GKS world coordinate system. Changing the transformation type does not affect the appearance of any GKS output primitive.

Note: When applying a logarithmic transformation to a specific axis, the system assumes that the axes limits are greater than zero.

EXAMPLE

```
gli> y:= 1, 2, 33, 44, 55, 10000
gli> X:= 1, 2, 33, 44, 55, 10000
gli> gus plot x, y
gli> gus set scale xy_log
gli> gus plot x, y
```

GUS SET SPACE

Sets the abstract Z-space used for mapping three-dimensional GUS output primitives into the current GKS world coordinate space.

FORMAT

GUS SET SPACE ***[z-min [, z-max [, rotation [, tilt]]]]***

PARAMETERS

z-min, z-max
Minimum and maximum values for the Z-axis.

rotation
Angle for the rotation of the X axis, in degrees.

tilt
Viewing angle of the Z axis in degrees.

DESCRIPTION

GUS SET SPACE establishes the limits of an abstract Z-axis and defines the angles for rotation and for the viewing angle (tilt) of a simulated three-dimensional graph, used for mapping corresponding GUS output primitives into the current GKS window. GUS uses these settings for all subsequent three-dimensional output primitives until other values are specified.

RESTRICTION

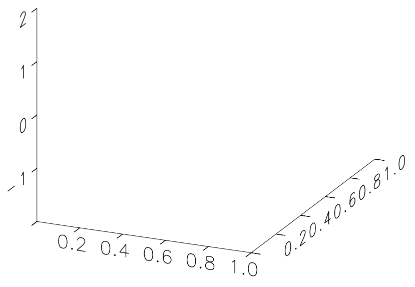
Angles of rotation and viewing angle must be specified between 0° and 90°.

EXAMPLE

This example sets the minimum value of the Z axis to -2 and the maximum value to 2 The rotation of the Z axis is set to 30° and the tilt is set to 45°.

```
gli> gus set space -2, 2, 30, 45
gli> gus axes_3d
```

FIGURE 5-27 Set 3-D Space



GUS SET SMOOTHING

Sets the smoothing level used for subsequent GUS output primitives.

FORMAT

GUS SET SMOOTHING *smoothing*

PARAMETERS

smoothing
Specifies a smoothing level in the range 1 to 20.

DESCRIPTION

GUS SET SMOOTHING sets the current smoothing level to the value specified by the parameter. GUS uses this value for all subsequent GUS output primitives (expect GUS POLYMARKER) until you specify another value.

The Y coordinates should contain a sequence of observed values of a function at equally spaced intervals to be smoothed.

The smoothing procedure is non-linear. The algorithm was originally developed by J.W. Tukey and Alberto Tubilio.

EXAMPLE

For this example, plot a data file then apply a smoothed polyline to the graph as a second curve.

Create sample data items and assign them to variables X and Y.

```
gli> x := -10(2)10
gli> y := cos(x)
```

Scale the window according to the data, draw the axes, and use GUS POLYLINE to plot the X and Y variables

```
gli> gus autoscale_2d x y
gli> gus axes_2d
gli> gus polyline x y
```

Use a GKS command to set the polyline type to dashed to represent the smoothed curve and set the smoothing level to 20 with GUS SET SMOOTHING.

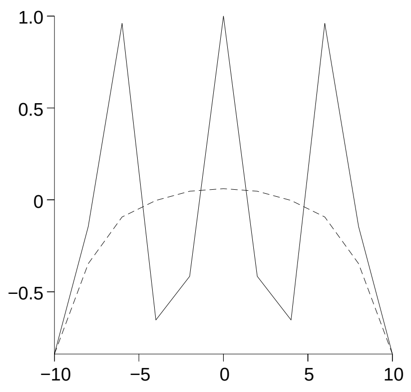
```
gli> gks set pline linetype dashed
gli> gus set smoothing 20
```

Using GUS POLYLINE plot the smoothed data using a dashed curve.

```
gli> gus polyline x y
```

GUS SMOOTHING continued

FIGURE 5-28 Smoothed Line Graph



GUS SET WS_VIEWPORT

Sets the orientation and the size of the workstation viewport.

FORMAT

GUS SET WS_VIEWPORT *window, size*

PARAMETERS

window
Specifies the workstation window (see Table 5-3, below).

TABLE 5-4 Window Options

Window	Description
FULL	Full viewport.
UPPER_HALF	Upper half.
LOWER_HALF	Lower half.
UPPER_LEFT	Upper left quarter.
UPPER_RIGHT	Upper right quarter.
LOWER_LEFT	Lower left quarter.
LOWER_RIGHT	Lower right quarter.

size
Specifies the workstation viewport size (see Table 5-4, below).

TABLE 5-5 Viewport Size

Viewport Size	Dimension
A3	0.42 x 0.297 meters (16.54 x 11.69 inches)
A4	0.297 x 0.21 meters (11.69 x 8.27 inches)
A5	0.21 x 0.1985 meters (8.27 inches x 7.81 inches)

DESCRIPTION

GUS SET WS_VIEWPORT automatically establishes the portion of the workstation display surface (workstation viewport) on which GKS maps the workstation window. AUTOSCALE can be used to scale the window to the data.

GUS SET WS_VIEWPORT continued

RESTRICTION

If the viewport size specified exceeds the display limits of the output device, clipping will occur.

EXAMPLE

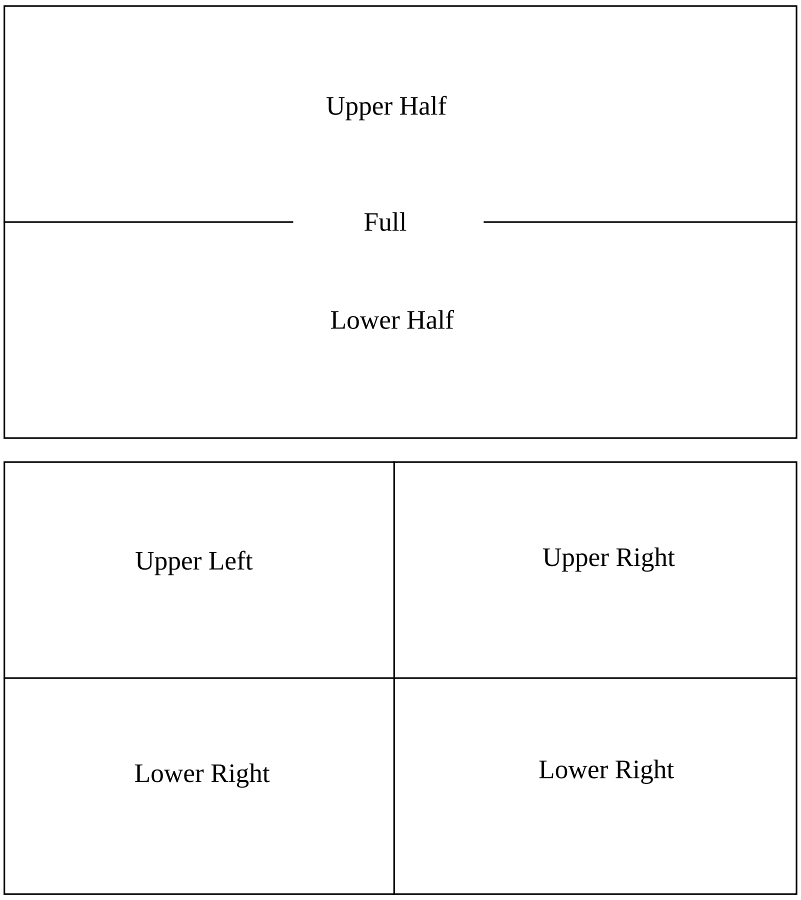
This command automatically draws graphics in a full-sized workstation window with viewport dimensions equivalent to DIN A4.

```
gli> gus set ws_viewport full a4
```

This command automatically draws graphics in the upper left portion of a DIN A3 viewport.

```
gli> gus set ws_viewport upper_left a3
```

FIGURE 5-29 GUS Window and Viewport Sizes



GUS SPLINE

Generates a cubic spline-fit, starting from the first data point and ending at the last data point.

FORMAT

GUS SPLINE *x-data, y-data*

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression.
- A constant expression.
- The name of a dynamic variable or function.
- A sub-range or range specification.

DESCRIPTION

GUS SPLINE fits a smooth curve through the given world coordinate points, in the order specified, using the current polyline attributes in the GKS state list.

All points are transformed and/or smoothed using the current values for GUS SET SCALE and GUS SET SMOOTHING.

The algorithm requires that the X coordinates are sorted in ascending order.

EXAMPLE

```
gli> x := .1, .2, .3, .6, .8
gli> y := .5, .1, .8, .2, .4
gli> gus spline x, y
gli> gus axes_2d
```

GUS SURFACE

Draws a three-dimensional surface plot for the given data points. Also is used with GUS CONTOUR to produce shaded contour plots.

FORMAT

GUS SURFACE *x-data, y-data, z-data [option [dimension_x]]*

PARAMETERS

x-data, y-data, z-data
X-data and Y-data define a grid. Z-data is a singly dimensioned array containing at least X-data * Y-data points. Z-data describes the surface height (or f(X,Y))at each point on the X,Y grid. Data for GUS SURFACE is ordered as shown in the figure on the following page.

option
Surface display options (see Table 5-5, below). Default option is LINES.

TABLE 5-6 Surface Options

Option	Description
LINES	Use X Y polylines to denote surface.
MESH	Use wire grid to denote surface.
FILLED_MESH	Applies opaque grid to surface.
COLORED_MESH	Applies colored grid to surface.
CELL_ARRAY	Applies a grid of individually-colored cells to the surface
SHADED_MESH	Applies light source shading to 3-D surface
Z_SHADED_MESH	Applies Z-value shading to surface.

dimension_x
When graphing a subset of the data, this parameter must be set to a value equal to the X dimension of the full data set. This option is not required when graphing the entire data set.

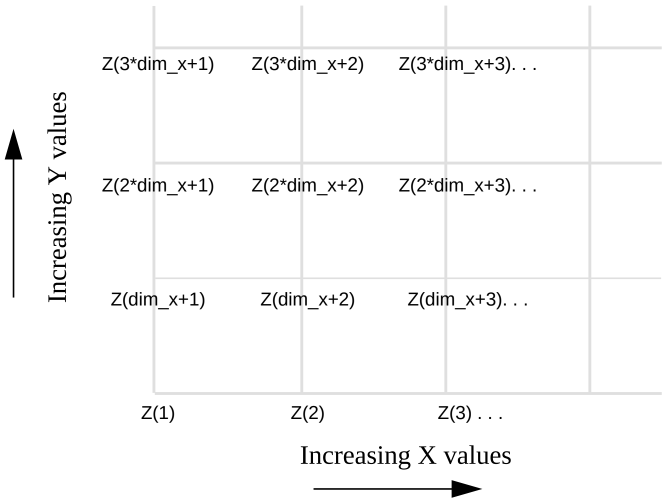
DESCRIPTION

GUS SURFACE constructs a three-dimensional surface plot for the given data points using the current polyline attributes and the current colormap in the GKS state list. A surface is defined as a graphical representation of a function of two variables (a field or 2-D function), where functional values are connected with lines of constant X and constant Y.

The surface is generated by projecting the 3-D coordinates that is (X, Y, FIELD(X,Y)), onto a 2-D plane. These projected points are then used to connect lines of constant X, Y and/or FIELD(X,Y), removing hidden lines in the process. The figure below illustrates a sample input field.

GUS SURFACE continued

FIGURE 5-30 GUS SURFACE Data Ordering



Surface representations are useful in numerous applications. For example, cartography and demography frequently use surfaces to represent functions defining elevation, average rainfall, population density, or other spatial data for a geographic region. A surface graphic may also be used simply to represent the range of a 2-D function over a specific domain of input values. For a surface graphic of elevation, the elevation, Z, at a given location would be represented by the function Z=ELEVATION(LAT, LONG).

GUS SURFACE generates a 2-D graphic from input data which is intrinsically 3-D. This process requires a projection of the 3-D coordinates onto a 2-D plane. To define such a projection, GUS SET SPACE is used to set the position in three-space indicating the viewing angle, which is determined by the tilt and rotation using GUS SET SPACE. Default values for rotation and tilt are 60° and 60°, respectively.

GUS SET SPACE is also used to define the minimum and maximum Z values to be graphed. If the Z values from the user's data are unknown, PRINT MIN(Z) and MAX(Z) can be invoked to determine the extreme range of Z for input to GUS SET SPACE.

Refer to GKS SET PLINE and GUS SET COLORMAP for information on changing SURFACE attributes.

Note: Window and viewport coordinates should be set according to the limits of X-data and Y-data. (Refer to GKS SET WINDOW and VIEWPORT, chapter 6.)

RESTRICTION

GUS SURFACE accepts array sizes that do not exceed 2×10^3 data elements in the X or Y directions. GUS SURFACE accepts only gridded X and Y values. Use the GRIDIT routine for randomly-spaced X and Y data.

GUS SURFACE continued

EXAMPLE 1

Create a surface graph using sample data defined as variables X, Y, and Z.

```
gli> x := -3(.15)3
gli> y := x
gli> z = exp(sin(y)*cos(x**2))
```

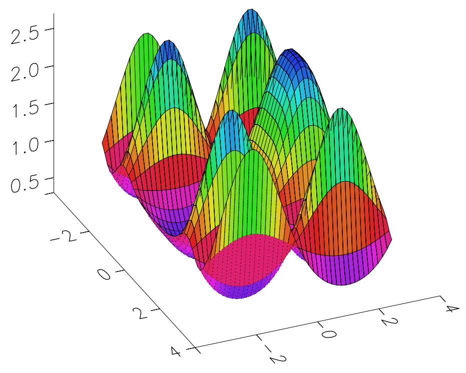
To avoid clipping portions of the graph, use GKS commands to set the window and viewport according to the X and Y data limits. The PRINT MIN(X) MAX(X) instruction will return the data limits if needed.

```
gli> gks set xform wc
gli> gks set viewport 0.15, 0.95, 0.15, 0.95
gli> gks set window -4, 4, -4, 4
gli> gus set space 0.3 2.7
```

Draw a 3-D shaded surface and overlay a wireframe mesh and add 3-D axes.

```
gli> gus surface x, y, z colored_mesh
gli> gus surface x y z mesh
gli> gus axes_3d
```

FIGURE 5-31 3-D Shaded Wireframe Surface



EXAMPLE 2

The `GUS SURFACE` utility can be used to create a shaded contour plot by tilting the surface and compressing or the Z values (height). `GUS CONTOUR` will overlay contour lines.

Using the same X, Y and Z data items defined in EXAMPLE 1 produce a shaded contour plot.

```
gli> x := -3(.15)3
gli> y := x
gli> z = exp(sin(y)*cos(x**2))
```

To avoid clipping portions of the graph, set the window and viewport using GKS commands.

```
gli> gks set xform wc
gli> gks set viewport 0.15, 0.95, 0.15, 0.95
gli> gks set window -4, 4, -4, 4
```

Use the `PRINT` command to determine the minimum and maximum Z values. Since `MAX` and `MIN` only accept variables, the Z function can be assigned a variable name, in this case TEMP.

```
gli> temp := z
gli> print min(temp) max(temp)
0.379411 2.63567
```

Use `GUS SET SPACE` to scale the window according to the Z values and to rotate (0°) and tilt (90°) the graph.

```
gli> gus set space 0.36 2.7 0 90
```

H can be defined to specify the contour intervals, or it can be set to 1 to produce 16 contour intervals equally distributed according to the Z values.

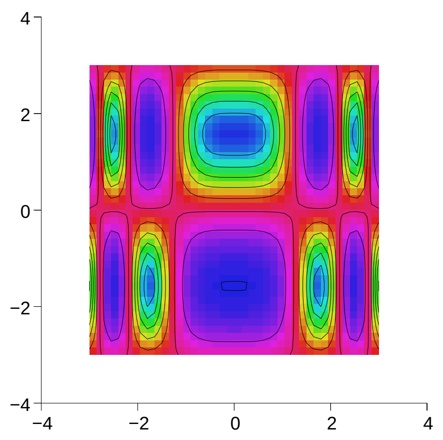
```
gli> h := .37(.3)2.6
```

Draw a colored contour and add axes.

```
gli> gus surface x, y, z z_shaded_mesh
gli> gus contour x y h z
gli> gus axes_2d
```

GUS SURFACE continued

FIGURE 5-32 Shaded Contour



GUS TEXT

Generates a character string starting at the given location. Strings can be defined to create mathematical symbols and Greek letters.

FORMAT

GUS TEXT *x-pos, y-pos, string*

PARAMETERS

x-pos, y-pos
Position of the text string specified in world coordinates.

string
Specifies a 1 to 80 character ASCII text string.

DESCRIPTION

`GUS TEXT` automatically draws and positions an ASCII text string according to the specified world coordinates and according to the current text attributes given by the GKS state list. The shape of the characters within the text string may vary depending on the current text attributes, the current normalization transformation, and the particular workstation capabilities. For information on changing the text style, color and size, refer to the GKS SET TEXT commands in chapter 6.

Note: GLI supports 22 software fonts and, if available, PostScript and X11 hardware fonts. See GKS SET TEXT commands in chapter 6 and the fonts shown in Appendix B for complete information on GLI software and hardware fonts.

The character string is interpreted to be a mathematical formula. The following notations apply:

- Terms containing a division operator (/) will be separated by a horizontal fraction line.
- Terms preceded by an exponential operator (** or ^), will be superscripted (raised and displayed in a slightly reduced character height).
- Alphanumeric characters (ABC…etc.) and characters enclosed in braces ({}) are interpreted as one term.
- Alphabetical characters contained within single quotes (‘ ‘) are evaluated as the value of the defined variable.

Special Expressions

Character strings may contain special expressions. A special expression must be enclosed between less than and greater than signs. The general form of a special expression is:

<argument [\ index [\ exponent [\ subscript [\ superscript]]]]>

Roots

A root must be enclosed between less than and greater signs (< >) and specified by the keyword root. The general form of a special expression is:

<ROOT| argument [| degree]>

argument
Applies an argument term to the root.

degree

Applies a degree term to the root.

Mathematical Sums

A mathematical sum must be enclosed between less than and greater than signs and specified by a keyword. The general form of a mathematical sum is:

<keyword| argument [| from [| to]]>

TABLE 5-7 Keywords for Mathematical Sums

Keyword	Symbol	Keyword	Symbol
INTEGRAL	$\int$	C_INTEGRAL	$\oint$
PRODUCT	$\prod$	SUM	$\sum$
CAP	$\cap$	CUP	$\cup$

argument
Applies an argument term to the sum.

from, to
Applies limits to the sum.

GUS TEXT continued

Greek Letters

To include a Greek letter you must specify the corresponding keyword between less than and greater signs (< >). The text translator produces uppercase or lowercase Greek letters depending on the case of the keyword.

<keyword>

TABLE 5-8 Keywords for Greek Letters

Keyword	Symbol	Keyword	Symbol
ALPHA	α, Α	OMICRON	ο, Ο
BETA	β, Β	PI	π, Π
CHI	χ, Χ	VARTHETA	ϑ
DELTA	δ, Δ	RHO	ρ, Ρ
EPSILON	ε, Ε	SIGMA	σ, Σ
PHI	ϕ, Φ	TAU	τ, Τ
GAMMA	γ, Γ	UPSILON	υ, ϒ
ETA	η, Η	PSI	ψ, Ψ
IOTA	ι, Ι	OMEGA	ω, Ω
KAPPA	κ, Κ	XI	ξ, Ξ
LAMBDA	λ, Λ	ZETA	ζ, Ζ
MU	μ, Μ	THETA	θ, Θ
NU	ν, Ν	VARPHI	φ

Parenthesis

Parenthesis must be enclosed between less than and greater signs and specified by a keyword. The general form of a special expression is:

<keyword| argument>

TABLE 5-9 Keywords for Parenthesis

Keyword	Symbol	Keyword	Symbol
PAR	( )	BRACKET	[ ]
BRACE	{ }	ABS	| |
NORM	|| ||		

argument
Applies an argument term to the parenthesis.

GUS TEXT continued

Mathematical Symbols

To include a mathematical symbol you must specify the corresponding keyword between less than and greater signs (< >).

<keyword>

TABLE 5-10 Keywords for Mathematical Symbols

Keyword	Symbol	Keyword	Symbol
DEGREE	°	RIGHT_ARROW	→
EPSILON	ε	UP_ARROW	↑
PM	±	LEFT_ARROW	←
DOWN_ARROW	↓	CROSS	×
MULT	·	NABLA	∇
DIV	÷	SURD	√
NEQ	≠	LWR	~
EQUIV	≡	LEQ	≤
INFTY	∞	EXISTS	∃
PROPTO	∝	LPAR	(
SUBSET	⊂	RPAR	)
CAP	∩	LBRACKET	[
SUPSET	⊃	RBRACKET	]
CUP	∪	LBRACE	{
IN	∈	RBRACE	}
EMPTY_SET	∅		

German Umlaut

To include the German Umlaut, the corresponding keyword must be specified between less than and greater signs. The text translator produces an uppercase or lowercase German Umlaut, depending on the case of the keyword.

<keyword>

TABLE 5-11 Keywords for German Umlaut

Keyword	Symbol	Keyword	Symbol
ae, AE	ä,Ä	oe, OE	ö,Ö
ue, UE	ü,Ü	ss, SS	ß

Special Characters

To include a special character, you must specify the corresponding keyword between less than and greater than (< >) signs.

<keyword>

TABLE 5-12 Keywords for Special Characters

Keyword	Symbol	Keyword	Symbol
LT (Less than)	<	GT (Greater than)	>
BCKSL (Back slash)	\	* (Asterisk)	*
/ (Forward slash)	/		

TABLE 5-13 Symbols for Superscripts and Subscripts

Symbol	Description
** (Double asterisk)	Subsequent text is superscript.
^ (Caret)	Subsequent text is superscript.
_ (Underscore)	Subsequent text is subscript.
\ (Back slash)	Subsequent text is subscript.
\\ (Double back slash)	Subsequent text is superscript.
\\\ (Triple back slash)	Subsequent text is placed directly beneath the character.
\\\\ (Quadruple back slash)	Subsequent text is placed directly on top of the character.

Note: The symbols listed in the above table should be enclosed in greater than or less than signs (< >) when an entire mathematical expression is to be subscripted or subscripted.

EXAMPLES

Example 1 outputs an integral expression:

```
gli> gus text 0.4 0.95 <intergral\1/x dx\x_0\x_1>
```

Example 2 outputs a sum:

```
gli> gus text 0.4 0.75 <sum\<x\i\2>\i=1\n>
```

Example 3 outputs an expression within braces:

```
gli> gus text 0.4 0.55 <brace\1, 1/2, 1/4, 1/8, 1/16, ...>
```

GUS TEXT continued

Example 4 outputs a root expression:

```
gli> gus text 0.4 0.35 <root\x_old\3>
```

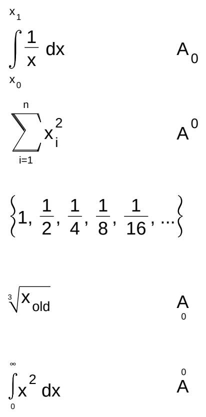
Example 5 demonstrates superscripts using the caret exponential operator:

```
gli> gus text 0.4, 0.15, <integral\x^2 dx\0\<infty>>
```

Examples 6, **7**, **8** and **9** show the results of using the back slash symbol to position the argument:

```
gli> gus text 0.7, 0.95 <A\0>
gli> gus text 0.7, 0.75 <A\\0>
gli> gus text 0.7, 0.35 <A\\\0>
gli> gus text 0.7, 0.15 <A\\\\0>
```

FIGURE 5-33 Mathematical Symbol Examples


$\int_{x_0}^{x_1} \frac{1}{x}\, dx$
A_0
$\sum_{i=1}^{n} x_i^2$
A^0
$\left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots\right\}$
$\sqrt[3]{x_{old}}$
$\underset{0}{A}$
$\int_0^\infty x^2\, dx$
$\overset{0}{A}$

CHAPTER 6

Graphical Kernel System Command Language

Table of Contents

Table of Contents continued

GKS Overview

A complete graphic, such as a simple X versus Y graph or contour plot, is composed of a large set of individual atomic graphical elements, called graphics primitives. Examples of these primitives would include polylines, polygons, text and symbols. Each primitive can have its own characteristics or attributes such as color, line type, etc. These graphic primitives have been assembled into a low-level graphics standard by the American National Standards Institute (ANSI) and the International Standards Organization (ISO). The standard is named the Graphical Kernel System (GKS).

In 1985, GKS became the first internationally approved graphics standard. As such, it represents a milestone in establishing machine-, language- and device-independence, thereby supporting an *open systems* concept. Prior to its approval (and with many graphics programs still today) anyone needing to create graphical representations of their data was relegated to writing device- and machine-dependent software. When faster or better computers became available, a substantial expenditure of resources was required to make previously written graphics applications compatible with the new hardware.

GKS standardizes and abstracts the creation of graphics primitives, providing a standard interface on which powerful, complex software can be constructed, while at the same time protecting investments made in previous software development. Applications written with GKS will transparently port to most computer systems and will function properly for a variety of graphics printers, plotters, terminals, and other devices.

GLI includes an implementation of the Graphical Kernel Standard (GKS). This implementation conforms to the ANSI/ISO GKS standard and includes the portable subroutine library of drawing tools (input/output primitives).

GKS allows the user to control the basic elements of a graph. GLI also includes the Graphics Utility System (see GUS, chapter 7) which, in many cases, preforms several GKS functions automatically, thereby giving GLI users sophisticated but easy-to-use graphics capability. In addition to the 2-D functions of GKS, GLI contains a number of tools and utilities to produce high quality, 3-D representations.

A basic understanding of the concepts and principals of GKS will assist the GLI user. The advanced user may want to review GKS in detail using a published text such as *Computer Graphics Programming: GKS - The Graphics Standard* by Enderle, Kansy and Pfaff, published by Springer-Verlag.

Although GLI includes its own implementation of GKS, it is also compatible with GKS software from other vendors.

Although the GKS standard includes components not discussed here, a fundamental understanding of the following will help GLI users.

- Operating States
- Transformations
- Workstations
- Output primitives
- Output primitives attributes

- Error Conditions

GLI is compatible with the GKS software available from most major vendors.

Operating States

Operating states ensure a well defined and consistent programming structure. All GKS functions require a specific state in which to properly execute. GLI handles initial operating states automatically, therefore you need very few control commands.

Transformations

GKS transforms the data in a two-stage process to produce the output primitive's drawing coordinates. The transformations occur between three coordinate systems:

- World Coordinates (WC) are defined by the user.
- Normalized Device Coordinates (NDC) specify a workstation independent abstract display surface and are used to define a uniform coordinate system for all workstations.
- Device Coordinates (DC) represent the workstation's display space coordinates.

The first stage transforms the WC data into device-independent coordinates, using the NDC space. Normalized device coordinates are "normalized" in that they are mapped onto the unit square (in the range from zero to one [0.0, 1.0] in both the X and Y directions).

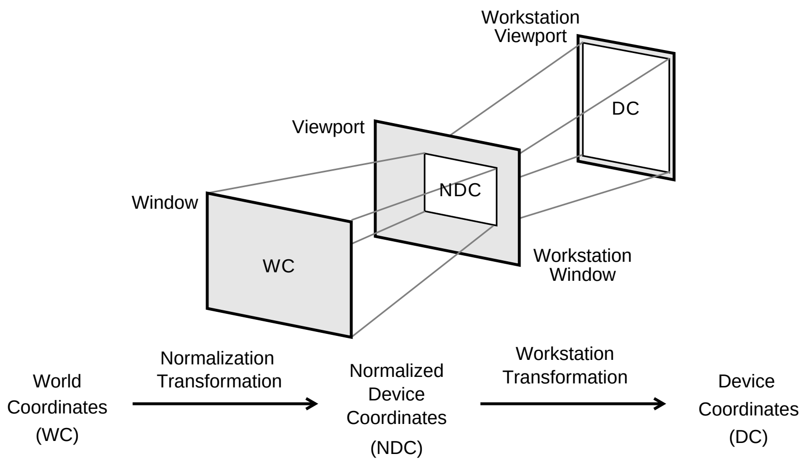
The transformation from WC to NDC is called the normalization transformation and is defined by the rectangle in the WC space (a window), and the rectangle in the NDC space (a viewport). Together, the window and the viewport define the normalization transformation which maps the contents of the window onto the viewport. The normalization transformations may be changed at any time during processing simply by redefining either the window or the viewport, or both.

The second stage transforms the device-independent coordinates so they apply to a specific display device. Device coordinates are stated in terms of units of length (meters by default), expressing the span in the X and Y directions of the display surface of output devices. This is done so that exact dimensional scaling can be accomplished.

The transformation from NDC to DC is called the workstation transformation and is defined by the rectangle in the NDC space (workstation window), and the rectangle in the DC space (workstation viewport).

The figure below illustrates the two-stage transformation process.

FIGURE 6-1 Transformation Process



Further, the viewport associated with a particular normalization transformation can define a “clipping rectangle” that, in addition to the window, specifies the normalization transformation. Graphics drawn within the window appear on the display device, and graphics drawn outside the clipping rectangle do not. Clipping on the viewport can be enabled or disabled as needed.

Workstations

In GKS, a workstation is an abstract graphical device, which usually is a representation of a physical device. GKS workstations fall into one of six workstation types described in the table below.

TABLE 6-1 GKS Workstations

Type	Description
Output	Has a display surface for the drawing of output primitives. Examples include pen plotters and laser printers.
Input	Includes at least one physical input device, such as digitizers, toggle switches or potentiometers.
Output/Input	Has both a display surface and at least one input device and is often called an interactive graphical workstation. Examples would include a terminal or microcomputer with a keyboard, mouse, joystick, trackball, lightpen or other input device.
Metafile Output	Allows the streaming of GKS graphics instructions to a data structure in a file which may be accumulated to mass-storage media, such as a disk or tape for archival, subsequent retrieval, editing, drawing or deletion. This allows a session to be recorded in detail and replayed later.
Workstation-Independent Segment Storage (WISS)	This workstation provides the ability to specify sub-portions of a picture for further processing, including the grouping of one or more graphical elements. This workstation has no corresponding

mapping onto a physical device and is used only as an internal workstation concept in the GKS model.

Output Primitives

GKS output primitives are abstractions of the basic capabilities of any display device. GKS provides for six (6) output primitives shown below.

TABLE 6-2 GKS Output Primitives

Primitive	Description
Polylines	Provides for the drawing of multi-point lines or curves.
Polymarkers	Draws selected symbols at given points and is used, for example, in a scatter diagram.
Text	A character string drawn at a given position.
Fill Areas	A polygon which may be hollow or filled with a color, pattern or hatch style.
Cell Array	Provides for the definition of individual *cells* in a grid which may be independently filled with a color.
Generalized Drawing Primitive	A software or hardware generated primitive, such as a circle or ellipse.

Output Primitive Attributes

Associated with each output primitive is information further defining its appearance such as color, line type, orientation, etc. Attribute information is both geometric (e.g., Text Height) and non-geometric (e.g., Text Color Index).

Error Conditions

GKS deals with error conditions by calling a standard error handling routine which in turn calls a standard error logging routine. GKS also provides a method of closing the GKS environment when an unpredictable error occurs.

Inquiry

The complete GKS state is recorded in a number of state lists and description tables. GKS provides inquiry functions allowing your application program to retrieve information about the current state of GKS.

ACTIVATE_WS

Starts activity in any previously deactivated workstation.

FORMAT

GKS ACTIVATE_WS name

PARAMETERS

name
name of the workstation to be activated

DESCRIPTION

GKS ACTIVATE_WS can be used to make active any workstation that has been made inactivate with the GKS DEACTIVATE_WS command. An inactive window can have no output directed to it from any GLI command, yet remains visible to the user. Using the GKS ACTIVATE_WS command in conjunction with the GKS DEACTIVATE_WS command, up to 13 X-windows (workstation type 211) can be displayed simultaneously on the screen. Any other type of workstation that has been deactivated with the GKS DEACTIVATE_WS can be reactivated using GKS ACTIVATE_WS.

RESTRICTIONS

Memory resources on a computer system can be exhausted if too many workstations are open at the same time. This depends upon the available memory and the number of programs running concurrently on your computer. If too much memory is being used at one time your system will crash.

GLI is capable of simultaneously displaying three different colormaps in three different windows on an 8 plane color system. When a fourth window is displayed with an additional color map, unpredictable results may occur. X Windows does not have the color resources to be able to display more than three GLI color maps at the same time on an 8 plane color system. 8 plane systems can display up to 13 windows using three or less color maps with no adverse effects.

EXAMPLE

Activate a previously inactive workstation.

```
gli> gks activate_ws wk1
```

Tailoring Commands

GLI commands may be abbreviated. For example:

```
gli> h
gli> he
gli> hel
gli> help
```

…all invoke the GLI Help facility.

You can substitute your own meaningful symbols for command names or parameters. Use assignment statements (:= and =) or the DEFINE SYMBOL command.

```
gli> define symbol eraseall = ”gks clear_ws”
```

You can define and use abbreviated forms of symbols using the abbreviation character, an asterisk (*). For example:

```
gli> define symbol erase*all = ”gks clear_ws”
```

…allows you to execute the GKS CLEAR_WS command using:

```
gli> erase
gli> eraseall
```

You can use symbols in places other than at the beginning of a command line by using the symbol substitution operator, an apostrophe (’). For example:

```
gli> filename = ”spectrum.dat”
gli> read ’filename’ x,y
```

…tells GLI that FILENAME is a symbol name and not a literal string. If you had not used apostrophes, the command would have looked for a file called FILENAME.DAT.

CELL_ARRAY

Defines individual rectangular cells which may be independently filled with color.

FORMAT

GKS CELL_ARRAY array [dimension_x]

PARAMETERS

array
An array of data values equal in size to (dimension_x)(dimension_x).

dimension_x
The number of cells to be drawn in one direction.

DESCRIPTION

A cell array is generated using the current world coordinate rectangle, the color index array, and the dimensions of the color index array. A virtual square is created on the screen which is divided into dimension_x by (array size/dimension_x) cells. The cells are filled with color in accordance with the value in the array.

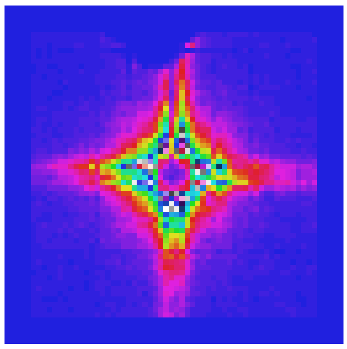
The CELL_ARRAY command does not support 3-D transformations or interpolation. However the CELL_ARRAY option to the GUS SURFACE command does support 3-D transformations and interpolation.

EXAMPLE

Read sample data from a GLI example program, and scale the data so it can be drawn inside a default window with the GKS CELL_ARRAY command. **Note**: Be sure to enter ‘GLI_DEMO’ in uppercase characters with single quotes.

```
gli> read ‘GLI_DEMO’xdemo2 z
 342 lines read from file xdemo2.dat
gli> ci := z/1790*119+8
gli> gks cell_array ci 64
```

FIGURE 6-2 GKS CELL_ARRAY



CLOSE_SG

Closes an open GKS segment.

FORMAT

GKS CLOSE_SG

PARAMETERS

None

DESCRIPTION

The GKS CLOSE_SG command closes the GKS open segment. Once a segment is closed, additional primitives can not be added or deleted from it.

In GLI, clearing all open workstations with the GKS CLEAR_WS command will implicitly close an open segment. The INITIALIZE instruction (see GLI Commands) also implicitly issues a GKS CLOSE_SG command.

EXAMPLE

```
gli> gks close_sg
```

CLEAR_WS

Clears the display surface.

FORMAT

GKS CLEAR_WS

PARAMETERS

None

DESCRIPTION

GKS CLEAR_WS erases previously generated primitives (graphics), allowing you to plot new images on the display surface.

EXAMPLE

```
gli> gks clear_ws
```

CLOSE_GKS

Closes the Graphical Kernel System (GKS).

FORMAT

GKS CLOSE_GKS

PARAMETERS

None.

DESCRIPTION

GKS CLOSE_GKS resets all GKS parameters to their default values and clears all buffers. GKS CLOSE_GKS releases all GKS internal data structures; the GKS state list and the workstation description tables become unavailable. You must close each workstation before you close GKS.

GKS CLOSE_GKS is part of the GLI exit procedure and is used when leaving GLI.

EXAMPLE

```
gli> gks close_gks
```

CLOSE_WS

Deactivates and closes a graphical workstation.

FORMAT

GKS CLOSE_WS name

PARAMETERS

name
Graphics device name (see the table below).

TABLE 6-3 Available Graphical Devices

Workstation Class	Environment Variable or VMS Logical Name	Description
FIGURE_FILE	GLI_FIG	Figure File (VAX Document, TeX, DECwrite, FrameMaker)
LASER_PRINTER	GLI_LW	Graphics Printer
PLOTTER	GLI_PL	Plotter
WISS		Workstation Independent Segment Storage
CGM	GLI_CGM	Computer Graphics Metafile
TERMINAL	GLI_CONID	Command Terminal, X display
WK1… WK12		Multiple workstations of the same type (the second workstation is called WK1). Useful for X displays.

Note: GLI documentation refers to generic input/output logical devices as *workstations.*

DESCRIPTION

GKS CLOSE_WS updates, deactivates, and closes a workstation previously opened by GKS OPEN_WS. The connection between GKS and the workstation is broken: the workstation's state list is de-allocated and the specified workstation identifier is deleted from the set of open workstations in the GKS state list. A primary use for this command is to close a workstation or file for archival purposes or for subsequent printing.

GKS CLOSE_WS does not clear the display surface. GKS CLEAR_WS implicitly issues a CLOSE_WS instruction and clears the display surface.

EXAMPLE

This example opens a workstation (HPGL plotter) and GLI commands create a graphic application. The GKS CLOSE_WS command is used to save the plotter GLI instructions.

```
gli> gks open_ws plotter 53
.
. ! a GLI graphics application is completed here
.
gli> gks close_ws plotter
```

DEACTIVATE_WS

Suspends activity in an active GLI workstation type.

FORMAT

GKS DEACTIVATE_WS name

PARAMETER

name

name of the workstation to be deactivated

DESCRIPTION

GKS DEACTIVATE_WS suspends activity in any active GLI workstation type.An inactive window can have no output directed to it from any GLI command, yet remains visible to the user. Using the GKS DEACTIVATE_WS command in conjunction with the GKS ACTIVATE_WS command up to 13 X-windows (workstation type 211) can be displayed simultaneously on the screen. Any other type of workstation that has been deactivated with the GKS DEACTIVATE_WS can be reactivated using GKS ACTIVATE_WS.

RESTRICTIONS

Memory resources on a computer system can be exhausted if too many workstations are open at the same time. This depends upon the available memory and the number of programs running concurrently on your computer. If too much memory is being used at one time your system will crash.

GLI is capable of simulteanously displaying three different colormaps in three different windows on an 8 plane color system. When a fourth window is displayed with an additional color map, unpredictable results may occur. X Windows does not have the color resources to be able to display more than three GLI color maps at the same time on an 8 plane color system. 8 plane systems can display up to 13 windows using three or less color maps with no adverse effects.

EXAMPLE

Deactivate a previously inactive workstation.

```
gli> gks deactivate_ws wk1
```

COPY_SG

Copies a GKS segment to all open workstations.

FORMAT

GKS COPY_SG

PARAMETERS

none

DESCRIPTION

The GKS COPY_SG command sends GKS primitives to all open GKS workstations.

EXAMPLE

GKS COPY_SG is useful when sending output simultaneously to several open GKS workstations. For instance, the following example opens a Workstation-Independent Segment Storage (WISS), creates a graph and directs the output to a laser printer and plotter using the GKS COPY_SG instruction. (See Appendix G for complete printing information.)

```
gli> gks open_ws wiss 5
.
. !GLI commands generating a graph
.
gli> gks clear_ws
gli> gks open_ws laser_printer 61
gli> gks open_ws plotter 51
gli> gks copy_sg
```

CREATE_SG

Creates a GKS segment, which is a collection of output primitives and attribute settings.

FORMAT

GKS CREATE_SG

PARAMETERS

None

DESCRIPTION

GKS CREATE_SG puts GKS into an operating state in which all subsequent output primitives (polylines, polymarkers, text) and their attributes are collected to form a segment. A segment is created until the next GKS CLOSE_SG is issued.

Note: In GLI, opening the Workstation Independent Segment Storage (WISS) with the GKS OPEN_WS command will implicitly create a segment.

GKS CREATE_SG is useful when creating device independent output for compute intensive applications.

EXAMPLE

```
gli> gks open_ws wiss 5
gli> gks create_sg
```

EMERGENCY_CLOSE

Closes GKS in a rapid and orderly fashion.

FORMAT

GKS EMERGENCY_CLOSE

PARAMETERS

None.

DESCRIPTION

GKS EMERGENCY_CLOSE closes the Graphical Kernel System in as rapid and orderly fashion as possible.

Usually, you will use GKS EMERGENCY_CLOSE when an unpredictable error condition occurs. GKS EMERGENCY_CLOSE saves as much graphical information as possible by following the actions outlined below:

1. Closes a GKS segment (if open);
2. Updates all active workstations;
3. Deactivates all active workstations;
4. Closes all open workstations;
5. Closes GKS.

EXAMPLE

```
gli> gks emergency_close
```

FILL_AREA

Generates a closed polygon filled according to the current color or interior style.

FORMAT

GKS FILL_AREA x-data, y-data

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GKS FILL_AREA draws a polygon as defined by the parameters and fills it with a uniform color, pattern, or hatch style. For information on changing the fill area attributes (color, pattern or style) refer to the GKS SET FILL command in this chapter.

GKS does not place any restrictions on the shape of the polygon. GKS determines which areas to fill by using the even/odd intersections rules. That is, an area is to be filled if a vector starting from a point within the area intersects the bounding polygon an odd number of times.

GKS FILL_AREA is clipped precisely around the clipping rectangle. As a result, clipping may change the boundaries of the bounding polygon and even generated multiple sub-areas from one original area.

EXAMPLE

This example draws a pattern-filled polygon using the FILL_AREA command.Using GKS instructions change the default fill color and interior style.

```
gli> gks set fill color_index red
gli> gks set fill int_style hatch
```

Create sample data items to plot and assign them to variables x and y.

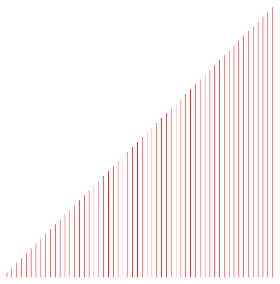
```
gli> x := 0.1,0.9,0.9
gli> y := 0.1,0.1,0.9
```

FILL_AREA continued

Plot the filled polygon using the GKS FILL_AREA command.

```
gli> gks fill_area x,y
```

FIGURE 6-3 Filled Polygon



INQUIRE GKS_STATE

Displays the GKS state list information.

FORMAT

GKS INQUIRE GKS_STATE

PARAMETERS

None

DESCRIPTION

The GKS state list contains information referring to the global state of GKS. GKS INQUIRE GKS_STATE reveals the settings of the current normalization transformations and the current settings of all workstation-independent primitive attributes.

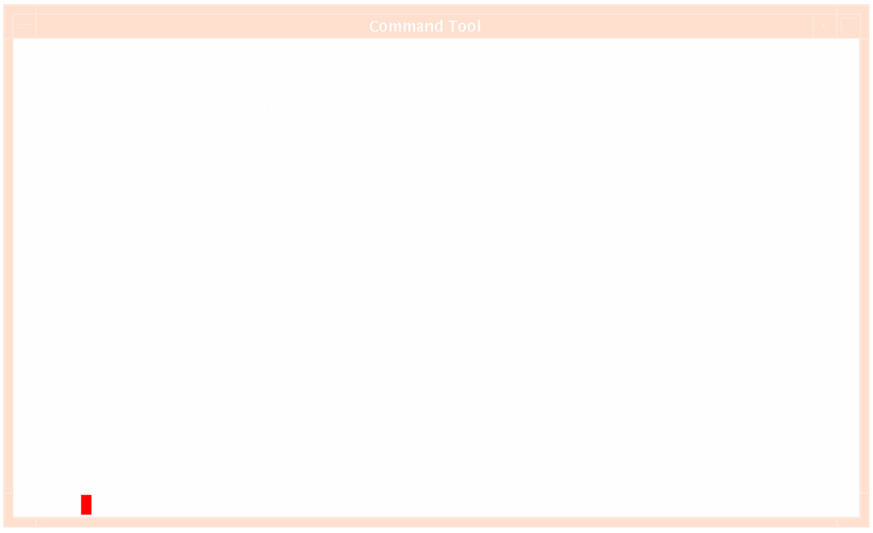
This command can be used to display all current attribute settings and transformations (color, size, line type, line style, log, linear, etc.) for polylines, polymarkers, text, fill areas.

EXAMPLE

Using the abbreviated INQUIRE keyword, the following command reveals the current GKS state list.

```
gli> gks inquire gks_state
```

FIGURE 6-4 Default GKS State List



INQUIRE WS_CONNTYPE

Queries connection and type of all open GKS workstations.

FORMAT

GKS INQUIRE WS_CONNTYPE

PARAMETERS

None

DESCRIPTION

GKS INQUIRE WS_CONNTYPE displays a list of and information concerning all currently open (active) workstations. For each workstation, the following information is displayed:

- Workstation Identifier
- Connection Identifier
- Workstation Type (see GKS OPEN_WS)

EXAMPLE

In this case, the command identifies the data stream being sent to one active X Window workstation.

```
gli> gks inquire ws_conntype
gks workstation connections on 17-FEB-1994 12:55:10.59
workstation id        stream id  type
     1                    0      211
```

INQUIRE WS_TYPE

Queries all GKS workstation types available to the system.

FORMAT

GKS INQUIRE WS_TYPE

PARAMETERS

None.

DESCRIPTION

GKS INQUIRE WS_TYPE displays a list of all available GKS workstation types. For each workstation type, the following information is displayed:

- Workstation Type (see GKS OPEN_WS);
- Maximum display surface size in device coordinates;
- Device coordinate units;
- Maximum display surface size in raster units.

Note: The GKS workstation types available will vary depending upon the implementation of GKS installed. Refer to GKS OPEN_WS for a complete list of the workstations available with GLI's GKS.

EXAMPLE

The table below shows workstation information for GLI's GKS.

```
gli> gks inquire ws_type
gli>GKS available workstation types on Mon Nov 20 10:23:34 1995


 Type      Category   Size       Unit       Device Units

 200       0          100.000, 100.000 cm   1012, 835
 201       2          25.600, 19.200 cm     1024, 766
 204       2          25.600, 19.200 cm     1024, 768
 207       2          25.600, 19.200 cm     1024, 768
 82        2          25.600, 19.200 cm     1024, 768
 51        0          27.200, 19.000 cm     10870, 7600
 53        0          27.200, 19.000 cm     10870, 7600
 72        2          25.600, 19.200 cm     1024, 768
 16        2          24.000, 14.400 cm     800, 480
 17        2          24.000, 14.400 cm     800, 480
 61        0          28.575, 19.685 cm     6750, 4650
 62        0          28.575, 19.685 cm     6750, 4650
 63        0          28.575, 19.685 cm     6750, 4650
 64        0          28.575, 19.685 cm     6750, 4650
 210       0          33.300, 28.100 cm     1024, 864
 211       2          34.200, 27.400 cm     1280, 1024
 212       2          33.300, 28.100 cm     1024, 864
 213       2          33.300, 28.100 cm     1024, 864
 214       0          33.300, 28.100 cm     1024, 864
 215       0          33.300, 28.100 cm     1024, 864
 216       0          33.300, 28.100 cm     1024, 864
 217       0          33.300, 28.100 cm     1024, 864
 230       0          33.300, 28.100 cm     1024, 864
 231       2          33.300, 28.100 cm     1024, 864
 232       2          33.300, 28.100 cm     1024, 864
 233       2          33.300, 28.100 cm     1024, 864
 7          4          100.000, 100.000 cm   65536, 65536
 8          4          100.000, 100.000 cm   65536, 65536
 5          3          100.000, 100.000 cm   32767, 32767
 41         2          33.300, 28.100 cm     1024, 864
 38         0          25.600, 19.200 cm     1024, 768
 103        0          25.400, 20.320 cm     720, 576
 104        0          25.400, 20.320 cm     750, 600
 92         0          27.940, 20.320 cm     1980, 1440
 2          4          100.000, 100.000 cm   65536, 65536

           Total of35 workstations.

gli>
```

OPEN_GKS

Opens the Graphical Kernel System (GKS).

FORMAT

GKS OPEN_GKS

PARAMETERS

None.

DESCRIPTION

GKS OPEN_GKS allocates, initializes, and allows access to the GKS state list, GKS description table, and the workstation description tables. The GKS operating state is changed from GKCL (GKS closed) to GKOP (GKS open).

This command must be invoked before using GKS, therefore, it is included as part of the GLI start-up procedure and is automatically executed when entering GLI.

EXAMPLE

```
gli> gks open_gks
```

OPEN_WS

Opens and activates a graphical workstation.

FORMAT

GKS OPEN_WS [name [ws-number]]

PARAMETERS

name
Graphics device. Variables are found in the GLISETUP file (see the table below).

TABLE 6-4 Available Workstations

Workstation Class	Environment Variable or VMS Logical Name	Description
FIGURE_FILE	GLI_FIG	Figure File (VAX Document, TeX, DECwrite, FrameMaker)
LASER_PRINTER	GLI_LW	Graphics Printer
PLOTTER	GLI_PL	Plotter
WISS		Workstation Independent Segment Storage
METAFILE	GLI_GKSM	GKSM Output Metafile
CGM_FILE	GLI_CGM	Computer Graphics Metafile
TERMINAL	GLI_CONID	Command Terminal, X Window
WK1… WK12		Multiple workstations of the same type (the second workstation is called WK1). Useful for X displays.

ws-type
Workstation number (see the table below).

TABLE 6-5 Available Workstation Number

Workstation Numbers	Devices
Graphics Files (FIGURE FILE)	
63, 64	Display PostScript with Compuserve GIF dump (b/w, color)
103, 104	Portable BitMap (72, 75 dpi)
Printers (LASER_PRINTER)	
38	DIGITAL LN03 PLUS
61, 62	PostScript, Color PostScript
92	DEC LJ250 Companion Color Printer
Plotters (PLOTTER)	
51, 53	HP-GL Graphics Plotter (HP 74xx, HP 75xx)
Terminals (TERMINAL)	
16, 17	Digital VT330, VT340 Video Terminal
72, 82	Tektronix 401x, 42xx Series Terminal
201	TAB 132/15-G Terminal

OPEN_WS continued

Terminals (continued)	
204	MONTEREY MG200 Display Terminal
207	IBM Personal Computer
Workstations (TERMINAL, WK1..WK12)	
41	VAX UIS
210	X Display (output only)
211	X Display
214	X Display with Sun rle rasterfile dump
215	X Display with Compuserve GIF dump
217	X Display with frame buffer
Segment Storage (WISS)	
5	Workstation Independent Segment Storage (WISS)
Metafiles (METAFILE)	
2	GKSM Output Metafile
CGM Files (CGM)	
7, 0x30007	Computer Graphics Metafile (Binary CGM, default for GLI)
8, 0x40007	Computer Graphics Metafile (Clear Text CGM)

DESCRIPTION

GKS OPEN_WS opens and activates a workstation class (i.e., Figure File, Laser Printer, CGM, etc.) for use by GLI. You must invoke GKS OPEN_WS before generating output for a workstation. Many GUS and GKS commands, such as GUS PLOT, GUS CONTOUR, GKS POLYLINE, GKS POLYMARKER, etc., automatically invoke GKS OPEN_WS.

Note: For the purposes of this section, the term *workstation* refers to the Workstation Class in Table 6-4. The Workstation Numbers designate device-specific output.

If establishing the first open workstation, the GKS operating state changes from GKOP (GKS open) to WSAC (at least one workstation active).

GKS OPEN_WS may be used to create and open a file, allowing you to save device specific output. The output may be PostScript, HPGL, or any of the other devices supported by GLI's GKS (or the GKS you are using). By default, graphics output is written to the output file GLI.EPS. Close the file with the GKS CLOSE_WS command.

The default output file type is .EPS (Encapsulated PostScript). You may select a different output file type by changing the GLI_FIG variable in the GLISETUP file, or by using a DEFINE LOGICAL command (See GLI Commands).

User Interface Language

Workstation type 212, the User Interface Language (UIL) generator, can be used to create Motif icons without the use of a icon editor. These icons can then be added to a user's own Motif applications.The system logicals GLI_ICON and GLI_UIL must be defined in order to use this workstation type. In this case only the GKS SET WS_-VIEWPORT command measures in pixels times 100, not meters. The command GKS SET WS_VIEWPORT 0 0.5 0 0.5 would, in this case, generate a bitmap 50 by 50 pixels in size (100 x 0.5 = 50). The .uil files generated with this tool can be then compiled along with the users’ Motif code.

Animation

Workstation Number 217, the X window frame buffer, is reserved by GLI’s implementation of GKS for animation. GLI graphical ouput, either plots or images, can be animated by the user by storing the output in the X window frame buffer prior to display. To animate a plot or image, open workstation number 217 using the command GKS OPEN_WS, then display the ouput to be animated. Next, use the GKS CLEAR_WS command to store the frame in the X Window buffer. Repeat this process for each frame. The command GKS CLOSE_WS is then used to close the workstation and animation will begin.

When the workstation is closed, control of the animated frames goes to the mouse not to the GLI command line. Mouse button 1 starts and stops the animation and mouse button 2 steps through individual frames of the animated images after the animation has been stopped with mouse button 1. Mouse button 3 exits the animation sequence entirely, returning control to the GLI command line. After you press mouse button 3 to exit an animation sequence, all frames stored in the X window buffer will be deallocated, and the memory will be freed and returned to the system for use by other applications.

Restrictions for Animation

Memory resources on a computer system can be exhausted if too many windows are open at the same time. This depends upon the available system memory and the number of programs running concurrently on your computer. If too much memory is being used simulteanously your system will crash. For reference, four (4) megabytes of system memory are required to display an animation sequence consisting of sixty-four images each containing a 256 x 256 pixel array.

Restrictions for Color Usage

GLI is capable of displaying three different color maps in three different windows at the same time on computers with 8-bit plane color support. When a fourth window is displayed with an additional color map, unpredictable results may occur. X windows does not have sufficient color resources to display more than three GLI color maps simultaneously on an 8 plane color system. 8 plane systems can display up to 13 windows using a total of three or less GLI color maps with no adverse effects.

Note: The GKS workstation types available may vary depending upon the implementation of GKS installed. The workstation information displayed above refers to GLI’s GKS, V4.4.

OPEN_WS continued

EXAMPLE 1

In this example, a PostScript (61) figure file is opened, and an imaginary command procedure (PLOT.GLI) directs output to the default GLI.EPS file.

```
gli> gks open_ws figure 61
gli> @plot temp.dat
gli> gks close_ws figure
```

EXAMPLE 2

This example displays an axes in one window, and then another window is opened and an axes displayed in it.

```
gli> gus axes_2d
gli> gks open_ws wk1 211
gli> gus axes_2d
```

POLYLINE

Generates a polyline, starting from the first data point and ending at the last data point.

FORMAT

GKS POLYLINE x-data, y-data

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GKS POLYLINE draws one or more straight line segments, called a polyline, connecting the world coordinate points in the order specified, using the current polyline attributes in the GKS state list. By default, GKS draws line segments as solid lines, at the nominal width and in the foreground color, at each data point.

For information on changing polyline attributes, refer to GKS SET PLINE commands.

Note: The GUS POLYLINE utility supports GUS commands for smoothing and log/linear transformations which are not supported by GKS POLYLINE.

EXAMPLE

This example plots a curve using GKS POLYLINE.

Sample data is created and assigned to variables X and Y.

```
gli> x := 0.1, 0.2, 0.3, 0.4, 0.5
gli> y := 0.5, 0.5, 0.6, 0.4, 0.5
```

The window is scaled to fit the data and default-style 2-D axes are drawn.

```
gli> gus autoscale_2d x y
gli> gus axes_2d
```

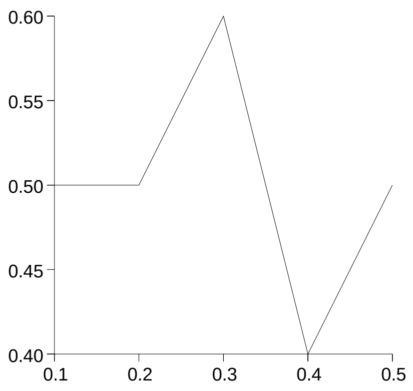
The X and Y data items are plotted using GKS POLYLINE.

```
gli> gks polyline x,y
```

Note: Additional curves can be applied to the graph using GKS POLYLINE.

POLYLINE continued

FIGURE 6-5 Line Graph



POLYMARKER

Generates a sequence of polymarkers starting from the first data point and ending at the last data point.

FORMAT

GKS POLYMARKER x-data, y-data

PARAMETERS

x-data, y-data
Data specification. A data specification may be:

- An arithmetic expression;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GKS POLYMARKER places one or more symbols, called polymarkers, at the specified world coordinates, using the current polymarker attributes in the GKS state list. By default, GKS produces an asterisk polymarker, at the nominal size and in the foreground color, at each data point. For information on changing polymarker attributes, refer to GKS SET PMARK instructions.

Note: The GUS POLYMARKER utility provides capabilities such as smoothing and log/linear transformations which are not supported by GKS POLYLINE.

EXAMPLE

This example graphs data points using GKS POLYMARKER.

Sample data is created and assigned to variables X and Y.

```
gli> x := 0.1,0.2,0.3,0.4,0.5
gli> y := 0.5,0.5,0.5,0.5,0.5
```

The window is scaled to fit the data and default-style 2-D axes are drawn.

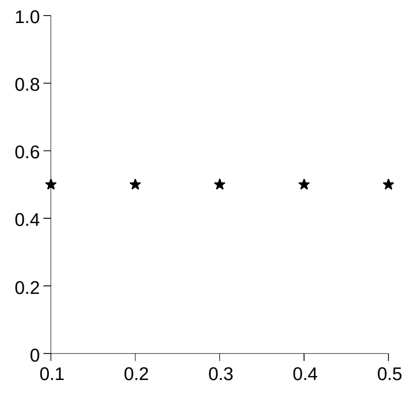
```
gli> gus autoscale_2d x y
gli> gus axes_2d
```

The X and Y data items are plotted using GKS POLYMARKER.

```
gli> gks polymarker x,y
```

POLYMARKER continued

FIGURE 6-6 Scatter Diagram



Note: Additional data sets can be plotted on the graph using GKS POLYMARKER.

SET FILL STYLE

Sets the current fill area style index entry in the GKS state list.

FORMAT

GKS SET FILL_STYLE index

PARAMETERS

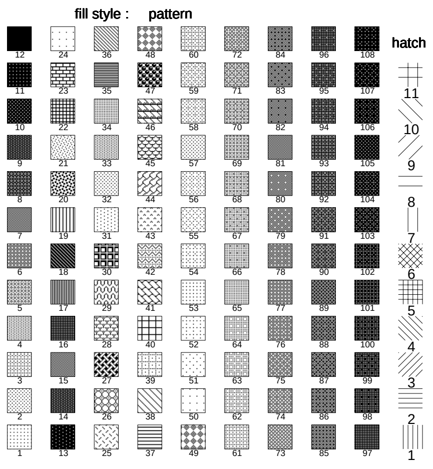
index

DESCRIPTION

GKS SET FILL STYLE specifies an index when PATTERN fill or HATCH fill is requested by the GKS SET FILL INT_STYLE command. If SET FILL INT_STYLE PATTERN is issued, the command SET FILL STYLE *INDEX* points to the device-specific pattern table of the workstation. If SET FILL INT_STYLE HATCH is invoked, the command SET FILL STYLE *INDEX* indicates the GLI fill style from the table below with index 1 being the default value.

If HOLLOW and SOLID is specified by SET FILL INT_STYLE, the style index is unused, and the SET FILL STYLE command is not needed.

TABLE 6-6 GLI Pattern/Hatch Fill Style Index



SET FILL STYLE continued

EXAMPLE 1

These commands set the fill type to PATTERN and the index is set to 2, which requests the corresponding device-specific fill type.

```
gli> gks set fill int_style pattern
gli> gks set fill style 2
```

EXAMPLE 2

These commands set the fill type to HATCH and the index is set to 2, which specifies the GLI fill as indicated in the above table.

```
gli> gks set fill int_style hatch
gli> gks set fill style 2
```

REQUEST LOCATOR

Provides a position in world coordinates when the left mouse button is pressed.

FORMAT

GKS REQUEST LOCATOR xcoord, ycoord

PARAMETERS

x-coord
The X coordinate value (in world coordinates) describing the location of the mouse pointer on the display when the left mouse button is pressed.

y-coord
The Y coordinate value (in world coordinates) describing the location of the mouse pointer on the display when the left mouse button is pressed.

DESCRIPTION

GKS REQUEST LOCATOR attempts to read the logical input values from your system’s default locator input device (e.g. a mouse) until the input is made (e.g. the left mouse button is clicked) or the request is cancelled (CTRL/Z or the middle mouse button is pressed).

The logical input values are the X and Y coordinates (in world coordinates) of the input device (e.g. the mouse pointer) when it is activated. If selected, the values are recorded as the specified parameters and may be used with other GLI instructions. If cancelled, the command is aborted.

Refer to your system documentation for details of how to select, confirm, and cancel input using your particular locator device.

EXAMPLE

In this command procedure, GKS REQUEST LOCATOR is used to determine which action is performed next, depending on where the mouse pointer is on the graph when the mouse button is pressed.

```
continue:
           gks request locator xcoord, ycoord
           if xcoord < 0.7 or xcoord > 0.9 then goto halfway
           if ycoord > 0.6 and ycoord < 0.7 then gosub done
halfway:
           if ycoord < 0.38 or ycoord > 0.82 then goto continue
           if xcoord > 0.08 and xcoord < 0.57 then gosub check
goto continue
.
.
.
gosub check
.
.
.
gosub done
```

REQUEST STROKE

Provides a sequence of points in world coordinates.

FORMAT

GKS REQUEST STROKE xcoord, ycoord

PARAMETERS

xcoord
An arbitrary name of a world coordinate array, indicating where the left mouse button is pressed.

ycoord
An arbitrary name of a world coordinate array, indicating the position of the pointing device where the left mouse button is pressed.

Note: In GLI, the maximum number of points is limited to 2048.

DESCRIPTION

GKS REQUEST STROKE attempts to read a sequence of logical input values from your system's default locator input device (e.g. a mouse) until:

- the input is made (e.g. the left mouse button is clicked),
- the request is cancelled (CTRL/Z or the middle mouse button is pressed), or
- the maximum number of points is input.

The logical input values are the X and Y points (in world coordinates) of the input device (e.g. the mouse pointer) when it is activated. When selected, coordinates are put in the arrays XCOORD and YCOORD until the maximum number of points is input or the middle mouse button is pressed. If cancelled, the command is aborted. The returned values can be used with other GLI instructions.

Refer to your system documentation for details of how to select, confirm, and cancel input using your particular locator device.

This command is useful when writing interactive applications with GLI.

EXAMPLE

In this simple example, points are input through the left mouse button and displayed on an axes.

To contain the graph, a window and viewport are created and defined in world coordinates (wc) using GKS commands.

```
gli> gks set xform wc
gli> gks set viewport 0.15 0.95 0.15 0.95
gli> gks set window 1 100 1 100
```

Using a GUS utility, axes are displayed as an area in which the points are to be added and identified as default-style polymarkers (asterisks).

```
gli> gus axes_2d
```

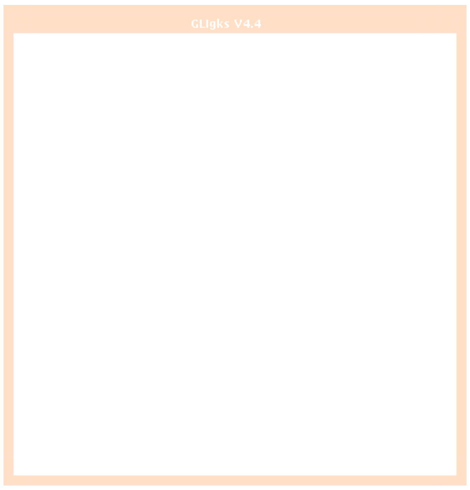
REQUEST STROKE continued

The command is invoked.

```
gli> gks request stroke xcoord, ycoord
gli> gks polymarker xcoord ycoord
```

Now, the when the left mouse button is pressed, the corresponding X and Y locations are identified as coordinates and added to the arrays arbitrarily named XCOORD and YCOORD. Pressing the middle mouse button stops the sequence and subsequent coordinates will not be added to the arrays.

FIGURE 6-7 Interactive Scatter Diagram



SET ASF

Determines how output primitive attributes are to be applied to output primitives.

FORMAT

GKS SET ASF flag

PARAMETERS

flag
Aspect Source Flag (see the table below).

TABLE 6-7 ASF Switch Settings

Switch	Description
BUNDLED	Addressed through indices into bundled tables.
INDIVIDUAL	Applied directly to the output primitive.

DESCRIPTION

GKS SET ASF determines how output primitive attributes (color, line type, size, etc.) will be changed. BUNDLED allows multiple attributes of an output primitive to be changed with a single command. INDIVIDUAL requires each attribute to be set independently. By default, Aspect Source Flag is set to INDIVIDUAL.

EXAMPLE

```
gli> gks set asf individual
```

SET CLIPPING

Sets the clipping indicator in the GKS state list.

FORMAT

GKS SET CLIPPING switch

PARAMETERS

switch
Clipping indicator (see the table below).

TABLE 6-8 Clipping Switch Settings

Switch	Description
OFF	Disable clipping in the current window.
ON	Enable clipping in the current window.

DESCRIPTION

GKS SET CLIPPING enables or disables clipping of the image drawn in the current window. Clipping is defined as the removal of those portions of the graph that lie outside of the defined viewport.

If clipping is on, GKS does not draw generated output primitives past the viewport boundaries. If clipping is off, primitives may exceed the viewport boundaries, and they will be drawn to the edge of the workstation window.

By default, clipping is on.

EXAMPLE

```
gli> gks set clipping off
```

SET COLOR

Defines the current color index entries in the GKS state list.

FORMAT

GKS SET COLOR index red green blue

PARAMETERS

index
Entry number for the GKS color index (from 0 to 7).

color
Intensities of red, green, and blue defining color (each in range 0 to 1).

DESCRIPTION

GKS SET COLOR defines the colors used in the color index for subsequent output primitives. Colors for polyline, polymarker, text, and fill area are controlled by the color index. Using this command, specific colors can be specified by modifying the RGB values.

By default, GLI uses the colors listed in the table below.

TABLE 6-9 Default GLI GKS Color Index

Index	Color	Red	Green	Blue
0	WHITE	1.000	1.000	1.000
1	BLACK	0.000	0.000	0.000
2	RED	1.000	0.000	0.000
3	GREEN	0.000	1.000	0.000
4	BLUE	0.000	0.000	1.000
5	CYAN	0.000	1.000	1.000
6	YELLOW	1.000	1.000	0.000
7	MAGENTA	1.000	0.000	1.000

GKS uses Index 0 as the default background color and Index 1 as the default foreground color.

The GKS SET COLOR command may also be used to define a colormap for use with surfaces in GLI. The user must define colors 8 through 80 which are reserved for use by the GLI colormaps. At least one workstation must be open and active to allow the redefinition of the colormap. All index values from 8 to 80 must have their red, green, and blue values defined in order to define a colormap.

Note: On machines with limited color capability, red, green, blue values will be mapped to their closest approximations. Colors may vary slightly from machine to machine.

SET COLOR continued

Note Also: GLI’s COLORMIX demonstration program can help you determine the exact proportions of RGB for a required color on systems capable of supporting X Windows. To use this tool, enter the following at the GLI prompt:

```
gli> @’GLI-DEMO’colormix
```

Users should position the pointer (cross hair) on the RGB slider bars and press the left mouse button. The RGB values for the mixed color is displayed beneath the slider bars. Once the proper color mix is obtained, use those RGB values as the color parameters for the GKS SET COLOR command.

EXAMPLE

In this example, GKS SET COLOR changes Color Index 4 (BLUE) to a light blue. All subsequent output primitives set to blue (i.e., GKS SET PLINE COLOR_INDEX BLUE) would be displayed using a light blue or, more exactly, the values of RGB defined below.

```
gli> gks set color 4 0.2 0.6 0.8
```

In this example the GKS SET COLOR command is used to define a colormap, which is then displayed on the screen.

```
gli> i := 1..72
gli> r := i/72
gli> g := r
gli> b := r
gli> gks set color i+7 1-r 1-g 1-b
gli> gus colormap
```

SET FILL COLOR_INDEX

Sets the current fill area color index entry in the GKS state list.

FORMAT

GKS SET FILL COLOR_INDEX color

PARAMETERS

color
Fill area color index (see the table below).

TABLE 6-10 Default GKS Color Index

Color	Red	Green	Blue
WHITE	1.000	1.000	1.000
BLACK	0.000	0.000	0.000
RED	1.000	0.000	0.000
GREEN	0.000	1.000	0.000
BLUE	0.000	0.000	1.000
CYAN	0.000	1.000	1.000
YELLOW	1.000	1.000	0.000
MAGENTA	1.000	0.000	1.000

DESCRIPTION

GKS SET FILL COLOR_INDEX defines the color of subsequent fill area output primitives. GKS uses the default foreground color for the default fill area color index.

The default colors can be altered using the GKS SET COLOR command.

Note: On machines with limited color capability, red, green, blue values will be mapped to their closest approximations. Colors may vary slightly from machine to machine.

EXAMPLE

In this example, the RGB values assigned to blue in the current GKS state list will be used to fill subsequent polygons.

```
gli> gks set fill color_index blue
```

SET FILL INT_STYLE

Sets the current fill area interior style entry in the GKS state list.

FORMAT

GKS SET FILL INT_STYLE style

PARAMETERS

style
Fill area interior style (see the table below).

TABLE 6-11 Fill Area Styles

Interior Style	Description
HOLLOW	No filling. Just draw the bounding polyline.
SOLID	Fill the interior of the polygon, using the fill color index.
PATTERN	Fill the interior of the polygon using the style index as a pattern index.
HATCHED	Fill the interior of the polygon using the style index as a cross-hatched style.

DESCRIPTION

GKS SET FILL INT_STYLE defines interior style entry for subsequent fill area output primitives. GKS uses this value for subsequent GKS FILL_AREA commands until you specify another value.

EXAMPLE

In this example, the fill area style is changed so that the GKS FILL_AREA command colors a polygon blue.

```
gli> x := 0.1 0.9 0.9
gli> y := 0.1 0.1 0.9
gli> gks set fill color_index blue
gli> gks set fill int_style solid
gli> gks fill_area x,y
```

SET PLINE COLOR_INDEX

Sets the current polyline color index entry in the GKS state list.

FORMAT

GKS SET PLINE COLOR_INDEX color

PARAMETERS

color
Polyline color index (see the table below).

TABLE 6-12 GKS Color Index

Color	Red	Green	Blue
WHITE	1.000	1.000	1.000
BLACK	0.000	0.000	0.000
RED	1.000	0.000	0.000
GREEN	0.000	1.000	0.000
BLUE	0.000	0.000	1.000
CYAN	0.000	1.000	1.000
YELLOW	1.000	1.000	0.000
MAGENTA	1.000	0.000	1.000

DESCRIPTION

GKS SET PLINE COLOR_INDEX defines the color of subsequent polyline output primitives.

GKS uses the default foreground color (black) for the default polyline color index.

Note: Refer to GKS SET COLOR to modify the RGB values for the default GKS color index.

EXAMPLE

This command determines that subsequent polylines (axes, curves on a line graph, grids, etc.) will be displayed in a color using the values of RGB assigned to blue in the GKS color index.

```
  gli> gks set pline color_index blue
```

SET PLINE LINETYPE

Sets the current line type entry in the GKS state list.

FORMAT

GKS SET PLINE LINETYPE line-type

PARAMETERS

line-type
Polyline type (see the table below).

TABLE 6-13 Polyline Types

Line Type	**Description**
SOLID	Solid line.
DASHED	Dashed line.
DOTTED	Dotted line.
DASH_DOTTED	Dashed_dotted line.
DASH_2_DOT	Sequence of one dash followed by two dots.
DASH_3_DOT	Sequence of one dash followed by three dots.
LONG_DASH	Sequence of long dashes.
LONG_SHORT_DASH	Sequence of a long dash followed by a short dash.
SPACED_DASH	Sequence of dashes double spaced.
SPACED_DOT	Sequence of dots double spaced.
DOUBLE_DOT	Sequence of pairs of dots.
TRIPLE_DOT	Sequence of groups of three dots.

DESCRIPTION

GKS SET PLINE LINETYPE defines the device-dependent or GKS-dependent polyline type for subsequent polyline output primitives.

EXAMPLE

This command determines that subsequent polylines (axes, curves on a line graph, grids, etc.) will be displayed using a dashed linetype.

```
gli> gks set pline linetype dashed
```

SET PLINE LINEWIDTH

Sets the current line width entry in the GKS state list.

FORMAT

GKS SET PLINE LINEWIDTH scale-factor

PARAMETERS

scale-factor
Scale factor applied to the nominal line width.

DESCRIPTION

GKS SET PLINE LINEWIDTH defines the line width of subsequent polyline output primitives. The polyline line width is calculated as the nominal line width generated on the workstation multiplied by the line width scale factor. This value is mapped by the workstation to the nearest available line width.

The default line width is 1.0, or 1 times the line width generated on the graphics device.

Note: Maximum and minimum values for the scale-factor are device dependent.

EXAMPLE

This command determines that subsequent polylines (axes, curves on a line graph, grids, etc.) will be displayed with a line width twice the default size.

```
gli> gks set pline linewidth 2.0
```

SET PMARK COLOR_INDEX

Sets the current polymarker color index entry in the GKS state list.

FORMAT

GKS SET PMARK COLOR_INDEX color

PARAMETERS

color
Polymarker color index (see the table below).

TABLE 6-14 GKS Color Index

Color	Red	Green	Blue
WHITE	1.000	1.000	1.000
BLACK	0.000	0.000	0.000
RED	1.000	0.000	0.000
GREEN	0.000	1.000	0.000
BLUE	0.000	0.000	1.000
CYAN	0.000	1.000	1.000
YELLOW	1.000	1.000	0.000
MAGENTA	1.000	0.000	1.000

DESCRIPTION

GKS SET PMARK COLOR_INDEX defines the color of subsequent polymarker output primitives. GKS uses the default foreground color (black) for the default polymarker color index.

Note: Refer to GKS SET COLOR to modify the RGB values for the GKS color index.

EXAMPLE

This command determines that subsequent polymarkers will be displayed in a color using the values of RGB assigned to blue in the GKS color index.

```
gli> gks set pmark color_index blue
```

SET PMARK SIZE

Sets the current marker size entry in the GKS state list.

FORMAT

GKS SET PMARK SIZE scale-factor

PARAMETERS

scale-factor
Scale factor applied to the nominal marker size.

DESCRIPTION

GKS SET PMARK SIZE defines the size of subsequent polymarker output primitives. The polymarker size is calculated as the nominal size generated on the workstation multiplied by the marker size scale factor. This value is mapped by the workstation to the nearest available size.

Polymarker type DOT is always displayed as the smallest displayable dot.

The default scale factor is 2.0, or2 times the size generated on the graphics device.

Note: Maximum and minimum sizes for polymarkers are device dependent.

EXAMPLE

To set the marker size to twice the default:

```
gli> gks set pmark size 2.0
```

To set the marker size to one half the default:

```
gli> gks set pmark size 0.5
```

SET PMARK TYPE

Sets the current marker type entry in the GKS state list.

FORMAT

GKS SET PMARK TYPE marker-type

PARAMETERS

marker-name
Polymarker type (see the table below).

TABLE 6-15 Polymarker types

Marker Name	**Symbol**
DOT	Smallest displayable dot.
ASTERISK	Asterisk.
CIRCLE	Hollow circle.
DIAGONAL_CROSS	Diagonal cross.
SOLID_CIRCLE	Filled circle.
TRIANGLE_UP	Hollow triangle pointing upward.
SOLID_TRI_UP	Filled triangle pointing upward.
TRIANGLE_DOWN	Hollow triangle pointing downward.
SOLID_TRI_DOWN	Filled triangle pointing downward.
SOLID_TRI_LEFT	Filled triangle pointing left
SOLID_TRI_RIGHT	Filled triangle point right
TRI_UP_DOWN	Hollow triangles pointing up and down overlaid
SQUARE	Hollow square.
SOLID_SQUARE	Filled square.
BOWTIE	Hollow bowtie.
SOLID_BOWTIE	Filled bowtie.
STAR	Hollow star
SOLID_STAR	Filled Star
DIAMOND	Hollow diamond
SOLID_DIAMOND	FIlled Diamond
HGLASS	Hollow hourglass.
SOLID_HGLASS	Filled hourglass.
HOLLOW PLUS	Hollow plus sign
SOLID PLUS	Solid plus sign

SET PMARK TYPE continued

DESCRIPTION

GKS SET PMARK TYPE defines the device-dependent or GKS-dependent polymarker type for subsequent polymarker output primitives. Polymarkers appear centered over their specified coordinates.

Polymarker colors and sizes are controlled using SET COLOR and SET PMARK SIZE commands.

EXAMPLE

```
  gli> gks set pmark type bowtie
```

GKS TEXT

Generates a character string using the current text attributes in the GKS state list.

FORMAT

GKS TEXT x-pos, y-pos, string

PARAMETERS

x-pos, y-pos
World coordinate position of the text string.

string
A 1 to 80 character ASCII text string to be written to the display surface.

DESCRIPTION

GKS TEXT writes a character string at the specified world coordinates according to the current text attributes. The shape of the characters within the text string may vary depending on the current text attributes, the current normalization transformation, and the particular workstation capabilities.

The starting position is given in world coordinates and is transformed by the current normalization transformation.

For information on changing the text attributes, including fonts, refer to the different GKS SET TEXT command descriptions in this chapter

EXAMPLE

```
gli> gks text 0.1, 0.5, Sample Text
```

SET TEXT ALIGN

Sets the current horizontal and vertical alignment for text in the GKS state list.

FORMAT

GKS SET TEXT ALIGN hor-align ver-align

PARAMETERS

hor-align
Horizontal text alignment (see the table below).

TABLE 6-16 Horizontal Alignment

Horizontal Alignment	Description
NORMAL, LEFT	Left justify.
CENTER	Center justify.
RIGHT	Right justify.

ver-align
Vertical text alignment (see the table below).

TABLE 6-17 Vertical alignment

Vertical Alignment	Description
TOP	Aligned with the top of the characters.
CAP	Aligned with the cap of the characters.
HALF	Aligned with the half line of the characters.
NORMAL, BASE	Aligned with the base line of the characters.
BOTTOM	Aligned with the bottom line of the characters.

DESCRIPTION

GKS SET TEXT ALIGN specifies how the characters in a text primitive will be aligned in horizontal and vertical space. Subsequent text is aligned using this value until another value is specified. The default, NORMAL NORMAL, indicates horizontal left alignment and vertical baseline alignment.

SET TEXT ALIGN continued

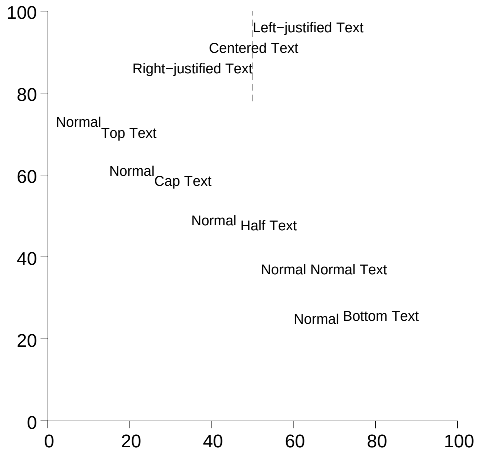
EXAMPLE

This command positions subsequent text using the text string and world coordinate position provided by a GKS TEXT instruction.

```
gli> gks set text align center half
```

The figure below shows several alignment options provided by this command.

FIGURE 6-8 Text Alignment



SET TEXT COLOR_INDEX

Sets the current text color index entry in the GKS state list.

FORMAT

GKS SET TEXT COLOR_INDEX color

PARAMETERS

color
Text color index (see the table below).

TABLE 6-18 GKS Color Index

Color	Red	Green	Blue
WHITE	1.000	1.000	1.000
BLACK	0.000	0.000	0.000
RED	1.000	0.000	0.000
GREEN	0.000	1.000	0.000
BLUE	0.000	0.000	1.000
CYAN	0.000	1.000	1.000
YELLOW	1.000	1.000	0.000
MAGENTA	1.000	0.000	1.000

DESCRIPTION

GKS SET TEXT COLOR_INDEX defines the color of subsequent text output primitives.

GKS uses the default foreground color (black) for the default text color index.

Note: Refer to GKS SET COLOR to modify the RGB values for the default GKS color index.

EXAMPLE

This command determines that subsequent text will be displayed in a color using the values of RGB assigned to blue in the GKS color index.

```
gli> gks set text color_index blue
```

SET TEXT EXPFAC

Sets the current character expansion factor (width to height ratio) in the GKS state list.

FORMAT

GKS SET TEXT EXPFAC factor

PARAMETERS

factor
Text expansion factor applied to the nominal text width-to-height ratio.

DESCRIPTION

GKS SET TEXT EXPFAC defines the width of subsequent text output primitives. The expansion factor alters the width of the generated characters, but not their height. Subsequent GKS TEXT commands use the value given until another value is specified.

The GKS SET TEXT EXPFAC default is 1, or one times the normal width-to-height ratio of the text.

RESTRICTION

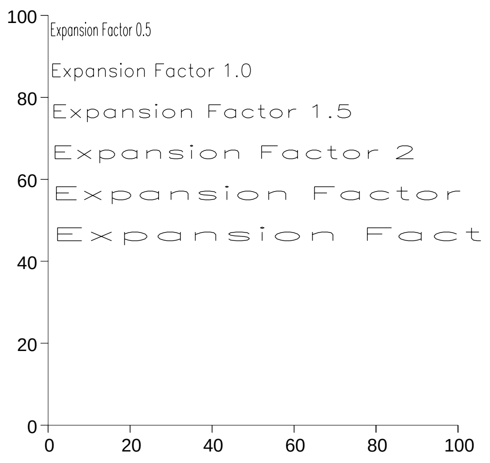
SET TEXT EXPFAC supports only stroke precision fonts.

EXAMPLE

To set the width-to-height ratio to three times the original stroke font design:

```
gli> gks set text expfac 3
```

FIGURE 6-9 Set Text Expansion Factor



SET TEXT FONTPREC

Sets the current text font and precision entries in the GKS state list.

FORMAT

GKS SET TEXT FONTPREC font, prec

PARAMETERS

font
Text font family (see the tables below).

TABLE 6-19 Available Software Fonts

Font ID	Font Name	Font ID	Font Name
-1	Standard	-17	German
-2	Mathematical	-18	Gothic
-3, -6, -9, -12, -15	Roman	-19	Anglo-Saxon
-4, -7, -10	Greek	-20	Scientific
-5, -13,	Script	-21	Symbolic
-8, -11, -16	Italics	-22	Cartographic
-14	Cryllic	-23	Roman (bold)

TABLE 6-20 Hardware Fonts

Font ID	Font Name	Font ID	Font Name
1, 9, 17, 25	Avant Garde Book	5, 13, 21, 29	New Century Schoolbook
2, 10, 18, 26	Courier	6, 14, 22, 30	Souvenir
3, 11,19, 27	Helvetica	8, 16, 24, 32	Times
4, 12, 20, 28	Lubalin Graph	7	Symbol

Note: Refer to Appendix B for examples of hardware and software fonts. Some devices may not support the set of hardware fonts listed above.

prec
Text precision (see the table below).

TABLE 6-21 Text Precision

Text Precision	Description
STRING	String precision (higher quality)
CHAR	Character precision (medium quality)
STROKE	Stroke precision (lower quality)

SET TEXT FONTPREC continued

DESCRIPTION

GKS SET TEXT FONTPREC defines the font family and text precision for subsequent text output primitives. Appendix B provides complete examples of GLI hardware and software fonts.

Specify a positive Font ID from Table 6 -20 to select hardware (device-dependent) fonts. Specify a negative Font ID from Table 6 -19 to select software (device-independent) fonts. Hardware fonts may not be supported by certain devices.

The appearance of a font depends on the text precision value specified. STRING, CHARACTER or STROKE precision allows for a greater or lesser realization of the text primitives, for efficiency. STRING is the default precision for GLI and produces the highest quality output. However, STRING requires the most computing resources and may therefore be time consuming for some devices.

If clipping occurs when STRING precision is invoked the entire text string is not drawn. When CHARACTER precision is specified, only that portion of the text string outside the viewport boundaries is not drawn. Text drawn at STROKE precision will be clipped in mid-character if clipping occurs.

Note: Hewlett - Packard systems running HP-UX do not support hardware fonts ITC Avant Garde Gothic, ITC Lubalin Graph and ITC Souvenir.

RESTRICTION

STRING and CHARACTER precision do not support the commands GKS SET TEXT UPVEC, GKS SET TEXT PATH, and GKS SET TEXT EXPFAC. Use STROKE precision when these commands are required.

EXAMPLE

In this example, a cartographic software font is invoked with high text precision.

```
gli> gks set text fontprec -22, stroke
```

SET TEXT HEIGHT

Sets the current character height entry in the GKS state list.

FORMAT

GKS SET TEXT HEIGHT value

PARAMETERS

value
Text height value.

DESCRIPTION

GKS SET TEXT HEIGHT defines the height of subsequent text output primitives. Text height is defined as a percentage of the default window. GKS uses this value for subsequent GKS TEXT commands until another value is specified.

GLI uses the default text of 0.027 (2.7% of the height of the default window).

RESTRICTION

SET TEXT HEIGHT supports only stroke precision fonts.

EXAMPLE

In this example, the text height is changed to 5% of the height of the default window.

```
gli> gks set text height 0.05
```

SET TEXT PATH

Define the current direction in which subsequent text will be drawn.

FORMAT

GKS SET TEXT PATH text-path

PARAMETERS

text-path
Text path (see the table below).

TABLE 6-22 Text Direction

Direction	Description		
DOWN		p	d
UP		u	o
LEFT	tfel		w
RIGHT	right		n

DESCRIPTION

GKS SET TEXT PATH defines the direction text output primitives will be written from the text string starting point. The GKS state list uses this value for subsequent GKS TEXT commands until another value is specified.

This command allows text strings to be written:

- to the right along the path,
- to the left along the path,
- upwards in perpendicular direction from the path, or
- downwards in a perpendicular direction from the path.

The text path is initially set to RIGHT.

RESTRICTION

SET TEXT PATH supports only stroke precision fonts.

EXAMPLE

```
gli> gks set text path right
```

SET TEXT UPVEC

Sets the current character text angle up vector entry in the GKS state list.

FORMAT

GKS SET TEXT UPVEC x-value, y-value

PARAMETERS

x-value, y-value
Text up vector.

DESCRIPTION

GKS SET TEXT UPVEC defines the vertical rotation of subsequent text output primitives. The GKS state list uses this value for subsequent GKS TEXT commands until another value is specified. The text up vector is initially set to (0,1), horizontal to the baseline.

Note: If values outside the range [-1, 1] are used to set the TEXT UPVEC, invalid results may occur.

RESTRICTION

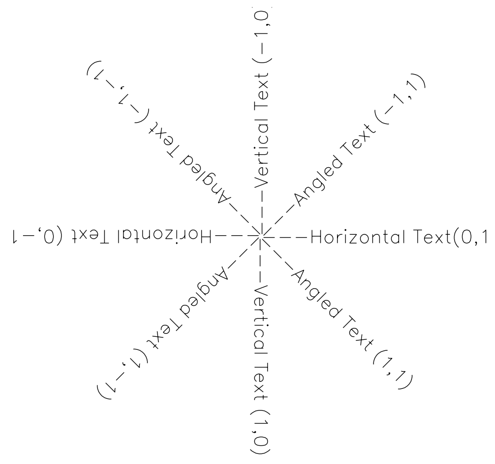
SET TEXT UPVEC supports only stroke precision fonts.

EXAMPLE

In this example, the GKS character up vector is changed so the text string, Sample Text, is displayed with a 45-degree angle of rotation.

```
gli> gks set text upvec 1, 1
gli> gks text 0.5 0.5 Sample Text
```

FIGURE 6-10 Text Angle and Parameters



SET VIEWPORT

Sets the limits of the normalized device coordinate space in the GKS state list.

FORMAT

GKS SET VIEWPORT x-min, x-max y-min, y-max

PARAMETERS

x-min, x-max y-min, y-max
Data specification. A data specification may be:

- An arithmetic expression;
- A constant;
- The name of a variable or function;
- A sub-range or range specification.

DESCRIPTION

GKS SET VIEWPORT defines the rectangular portion of the Normalized Device Coordinate (NDC) space to be associated with the specified normalization transformation. The NDC viewport and World Coordinate (WC) window define the normalization transformation through which all GKS output primitives pass. The WC window is mapped onto the rectangular NDC viewport which is, in turn, mapped onto the display surface of the open and active workstation, in device coordinates.

Note: See the introduction of this chapter for a complete discussion of the transformation process.

EXAMPLE

An application might require a filled contour graph with an associated legend. (This example assumes a suitable workstation window has been previously defined.)

A viewport could be defined to contain the contour graph.

```
gli> gks set viewport 0.05, 0.8, 0.05, 0.95
```

After the contour is drawn in the viewport described above, a viewport could then be redefined for the legend and positioned to the right side of the contour.

```
gli> gks set viewport 0.85, 0.9, 0.05, 0.95
```

SET WINDOW

Sets the limits of the world coordinate space in the GKS state list.

FORMAT

GKS SET WINDOW x-min, x-max y-min, y-max

PARAMETERS

x-min, x-max y-min, y-max
Limits of the window specified in world coordinates.

DESCRIPTION

GKS SET WINDOW defines the rectangular portion of the World Coordinate space (WC) to be associated with the specified normalization transformation. The WC window and the Normalized Device Coordinates (NDC) viewport define the normalization transformation through which all GKS output primitives are mapped. The WC window is mapped onto the rectangular NDC viewport which is, in turn, mapped onto the display surface of the open and active workstation, in device coordinates.

By default, GKS uses the range [0,1] x [0,1], in world coordinates, as the normalization transformation window.

Note: See the introduction of this chapter for a complete discussion of the transformation process.

EXAMPLE 1

The following example would scale a window and position it in the upper left hand quarter of the current workstation viewport.

```
gli> gks set window -3.0, 3.0, -1, 1
gli> gks set viewport 0, 0.5, 0.5, 1
```

EXAMPLE 2

In this instance, the window limits are set (using world coordinates) in the X direction to years 1982 to 1992 and in the Y direction to amounts $0.00 to $100,000.

```
gli> gks set window 1982, 1992, 0, 100000
gli> gks set viewport 0.05, 0.95, 0.05, 0.95
```

SET WS_VIEWPORT

Defines the size of the workstation graphics window in meters.

FORMAT

GKS SET WS_VIEWPORT x-min, x-max y-min, y-max

PARAMETERS

x-min, x-max, y-min, y-max
Dimensions of the workstation window specified in meters.

DESCRIPTION

GKS SET WS_VIEWPORT places a workstation window on the display of the specified size in meters. This command allows the workstation window to be accurately sized for a display or hardcopy device, and is often useful for sizing graphs for desktop publishing applications.

EXAMPLE

This example produces a workstation window with dimensions of 0.297 meters by 0.21 meters.

```
gli> gks set ws_viewport 0, 0.297, 0, 0.21
```

SET WS_WINDOW

Sets the area of the NDC viewport that is to be drawn in the workstation window.

FORMAT

GKS SET WS_WINDOW x-min, x-max y-min, y-max

PARAMETERS

x-min, x-max y-min, y-max
Minimum and maximum values of the viewport displayed. All coordinates are in NDC values, which lie in the range of [0, 1].

DESCRIPTION

GKS SET WS_WINDOW defines the rectangular area of the Normalized Device Coordinate space to be drawn in the workstation window.

By default, the workstation transformation will map the range [0,1] x [0,1] in NDC onto the largest square on the workstation's display surface. The aspect ratio of the workstation window in maintained at 1 to 1.

Note: It is recommended that this command not be used unless the user is familiar with the transformation process.

EXAMPLE

```
  gli> gks set ws_window 0, 0.5, 0.5, 1
```

SET XFORM

Sets the normalization transformation type in the GKS state list.

FORMAT

GKS SET XFORM type

PARAMETERS

type
Transformation type (see the table below).

TABLE 6-23 Workstation types

Transformation Type	Description
NDC	Normalized Device Coordinates.
WC	World Coordinates.

DESCRIPTION

GKS SET XFORM specifies the normalization transformation to be used by the associated window and viewport to transform subsequent output primitives from the device-dependent world coordinate system to the device-independent normalized device coordinate system.

GLI supports two coordinate systems: the normalized device coordinate space (NDC), and the world coordinate space (WC).

Normalized device coordinate space is defined to be the unit square. That is, the window limits correspond to the viewport limits of 0.0 to 1.0 in both the X and Y directions. NDC is not user-definable.

World coordinate space (WC) is a user-definable normalization transformation. You may only select one normalization transformation at a time.

EXAMPLE

```
  gli> gks set xform ndc
```

UPDATE _WS

Clears the display space.

FORMAT

GKS UPDATE_WS

PARAMETERS

None

DESCRIPTION

GKS UPDATE_WS outputs the GKS buffer which redraws and displays generated graphics in the current X window. This command is useful to redraw graphs on terminals without backing store capability. GKS UPDATE_WS forces the regeneration of the display and redraws previously obscured window areas.

The symbol ‘flush’ has been defined to be equivalent to GKS UPDATE_WS in the file ‘glistartup.gli’.

EXAMPLE

```
gli> gks update_ws
```

or, alternatively

```
gli> flush
```

CHAPTER 7

Imaging

Table of Contents

Table of Contents continued

GLI Imaging

GLI includes a complete system for processing and displaying images. An image consists of a 2-D array of pixels where each pixel represents a color and intensity value for the corresponding data element in the array. This chapter describes the functions and commands included in GLI for image processing.

If you would like to view a demonstration of the GLI IMAGE program, enter the following at the GLI prompt:

```
gli> image_demo
```

Using IMAGE you can:

- read, write and display an image
- export or import image data to or from GLI variables
- apply colortables to an image
- stretch or compress an image
- flip an image around the horizontal or vertical axes
- rotate an image in 90 degree steps
- filter an image
- enhance the edges of an image
- invert an image
- equalize the intensity histogram of an image
- modify the contrast of an image
- animate a sequence of images

GLI supports several image formats, including:

- Portable PixMap (PPM)
- Portable BitMap (PBM)
- Portable ColorMap (PCM)
- Portable GrayMap (PGM)

The IMAGE module is accessible through a command line interface or an X Windows point-and-click interface. However, a system configuration capable of supporting X Windows is required to use the point-and-click interface.

The GLI IMAGE point-and-click interface can be started by entering:

```
gli> image
```

To invoke IMAGE commands from the GLI prompt, use the following command syntax:

```
gli> image [any valid image command]
```

Animation

Using the IMAGE module on systems compatible with X WIndows, any graphic or image stored in GLI can be animated to show data fluctuations. Refer to GKS Commands, OPEN_WS, for more information on GLI animation.

Image Formats

The GLI IMAGE module supports the image formats shown in the table below. These formats are also supported by PBMplus, which is an image format conversion program available on many public ftp servers. PBMplus converts a variety of formats to those supported by GLI IMAGE

TABLE 7-1 Image Formats

Format Name	**Description**
Portable PixMap (PPM)	Supports 24-Bit color images.
Portable ColorMap (PCM)	Supports 8-Bit color images containing a color table with a maximum of 256 color entries.
Portable GrayMap (PGM)	Supports lowest common denominator for grayscale.
Portable BitMap	Supports the lowest common denominator for monochrome.

Portable PixMap

The PPM (Portable PixMap) format is an image format for 24-Bit color images. It is fully compatible with the PPM format of the PBMplus package.

The PPM format is defined as:

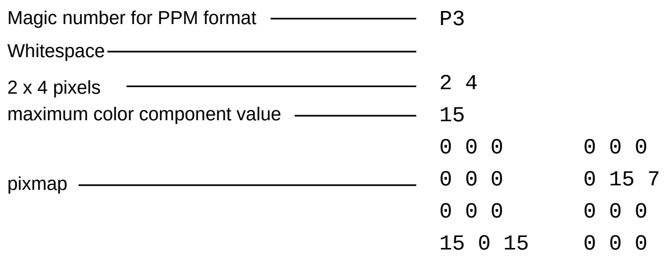
- A “magic number” for identifying the file type. A PPM file’s magic number are the two characters “P3”.
- Whitespace, including either: blanks, TABs, carrier returns or line feeds).
- A width, formatted as ASCII characters in decimal notation.
- Whitespace.
- A height, specified in ASCII decimal notation.
- Whitespace.
- The maximum color component value, again in ASCII decimal. In contrast to the PBMplus package, the GLI IMAGE module supports only a maximum color component value of 255.
- White space.
- Width * height pixels, each group of three ASCII decimal values corresponding to three colors (red, green, blue) are considered a pixel. Allowed intensity values are between 0 and the specified maximum value, starting at the top left corner of the pixmap, proceeding in normal English reading order. A value of 0 means that color is off, and the maximum value means that color is at maximum intensity.

- Characters from a “#” to the next end of line are ignored and designate comments.
- No line should be longer than 70 characters.

There is also a variant of the PPM format called RAWBIT. RAWBIT files are smaller and significantly faster to read and write. This variant differs from PPM in the following ways:

- The “magic number” is “P6” instead of “P3”.
- The pixel values are stored as plain bytes, instead of ASCII decimal.
- Whitespace is not allowed in the pixel area, and only a single character of whitespace (typically a new line) is allowed after the maximum value.

FIGURE 7-1 Example of PPM Image Format



```
P3

2 4
15
0 0 0      0 0 0
0 0 0      0 15 7
0 0 0      0 0 0
15 0 15    0 0 0
```

Portable ColorMap

The PCM (Portable ColorMap) format is an image format for 8 Bit color images containing a color table with a maximum of 256 color entries.

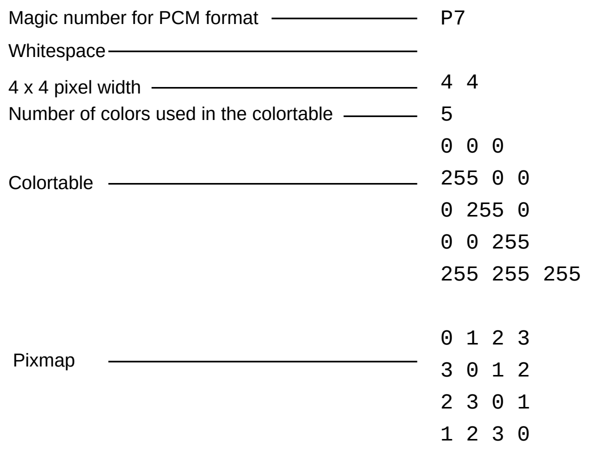
PCM format is defined as including:

- A “magic number” for identifying the file type. A PCM file’s magic number are the two characters “P7”.
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.
- Whitespace.
- A height, in ASCII decimal.
- Whitespace.
- The number of colors used in the colortable, again in ASCII decimal.
- Whitespace.
- The colortable, consisting of RGB triples. Each RGB triple consists of three integer numbers between 0 and 255. An integer value of 0 means that the color is off, and a value of 255 means that the color is at maximum intensity.

- Width * height pixels, each three ASCII decimal values between 0 and the number of colors minus one, starting at the top left corner of the pixmap, proceeding in normal English reading order. Each value is an index specifying a color in the colortable.
- Characters from a “#” to the next end of line are ignored (comments).
- No line should be longer than 70 characters.
- Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a pixmap.
- There is also a variant on the format called RAWBIT format. This variant is different in the following ways:

- The “magic number” is “P8” instead of “P7”.
- The values of the colortable and the pixel values are stored as plain bytes, instead of ASCII decimal.
- Whitespace is not allowed in the pixels area, and only a single character of whitespace (typically a newline) is allowed after the maximum value. There must not be any spaces between the colortable and pixel values.
- RAWBIT files are smaller and many times faster to read and write.

FIGURE 7-2 Example of PCM Image Format


Magic number for PCM format
Whitespace
4 x 4 pixel width
Number of colors used in the colortable
Colortable
Pixmap

```
P7

4 4
5
0 0 0
255 0 0
0 255 0
0 0 255
255 255 255

0 1 2 3
3 0 1 2
2 3 0 1
1 2 3 0
```

Portable GrayMap

The Portable GrayMap (PGM) format is the lowest common denominator grayscale file format. The PGM format is defined as including:

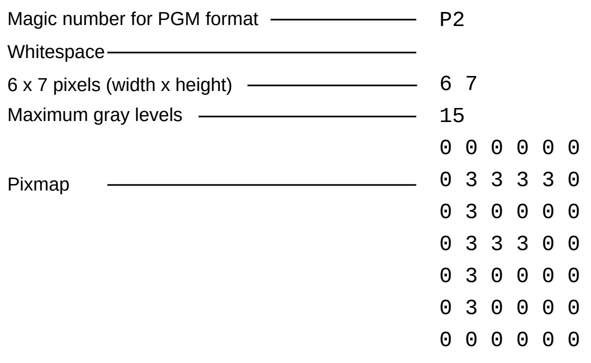
- A “magic number” for identifying the file type. A PGM file’s magic number are the two characters “P2”.
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.
- Whitespace.
- A height, again in ASCII decimal.

- Whitespace.
- The maximum gray value, again in ASCII decimal. In contrast to the PBMplus package the image system supports only a maximum gray value of 255.
- Whitespace.
- Width * height gray values, each in ASCII decimal, between 0 and the specified maximum value, separated by whitespace. Values start at the top left corner of the graymap, proceeding in normal English reading order. A value of 0 means black, and the maximum value means white.
- Characters from a "#" to the next end of line are ignored (comments).
- No line should be longer than 70 characters.
- Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a graymap.

There is also a variant to the PGM format called RAWBIT format. This variant is different in the following ways:

- The "magic number" is "P5" instead of "P2".
- The gray values are stored as plain bytes, instead of ASCII decimal.
- No whitespace is allowed in the gray section, and only a single character of whitespace (typically a new line) is allowed after the maximum value.
- The files are smaller and many times faster to read and write.

FIGURE 7-3 Example of PGM Image Format



```
P2

6 7
15
0 0 0 0 0 0
0 3 3 3 3 0
0 3 0 0 0 0
0 3 3 3 0 0
0 3 0 0 0 0
0 3 0 0 0 0
0 0 0 0 0 0
```

Portable BitMap

The portable bitmap format is a lowest common denominator monochrome file format. The definition is as follows:

- A "magic number" for identifying the file type. A PBM file's magic number are the two characters "P1".
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.

- Whitespace.
- A height, again in ASCII decimal.
- Whitespace.
- Width * height bits, each either ‘1’ or ‘0’, starting at the top left corner of the bitmap, proceeding in normal English reading order.
- The character ‘1’ means black, ‘0’ means white.
- Whitespace in the bits section is ignored.
- Characters from a “#” to the next end of line are ignored (comments).
- No line should be longer than 70 characters.

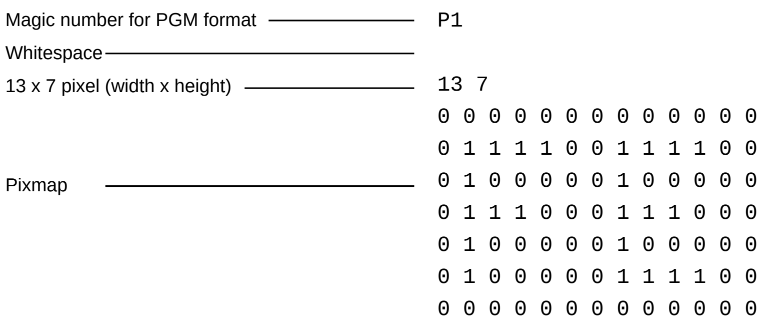
Programs that read the PBM format should be as lenient as possible, accepting anything that looks remotely like a bitmap.

There is also a variant for the PBM format called RAWBIT format. RAWBIT files are eight times smaller and significantly faster to read and write.

This RAWBIT variant is different in the following ways:

- The “magic number” is “P4” instead of “P1”.
- The bits are stored eight per byte, high bit first, low bit last.
- No whitespace is allowed in the bits section, and only a single character of whitespace (typically a new line) is allowed after the height.

FIGURE 7-4 Example of PGM Image Format



```
P1

13 7
0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 1 1 1 1 0 0
0 1 0 0 0 0 0 1 0 0 0 0 0
0 1 1 1 0 0 0 1 1 1 0 0 0
0 1 0 0 0 0 0 1 0 0 0 0 0
0 1 0 0 0 0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
```

Image Commands

The remainder of this chapter describes the commands available with the GLI IMAGE utility. IMAGE commands use the following style:

```
gli> image [any valid image command]
```

All commands listed in this chapter are also available through the GLI's X Windows interface. To invoke the IMAGE point-and-click interface on systems capable of supporting X Windows, enter:

```
  gli> image
```

Note: The images displayed in this chapter may not represent the image quality shown on your display. Many X Window and graphics terminal users can display images at significantly higher resolution than the examples shown in this chapter.

CONTRAST

Alters the contrast of an image.

FORMAT

IMAGE CONTRAST *source destination level*

PARAMETERS

source
original image

destination
Name of image with improved contrast

level
A value between the range of -1.0 and 1.0

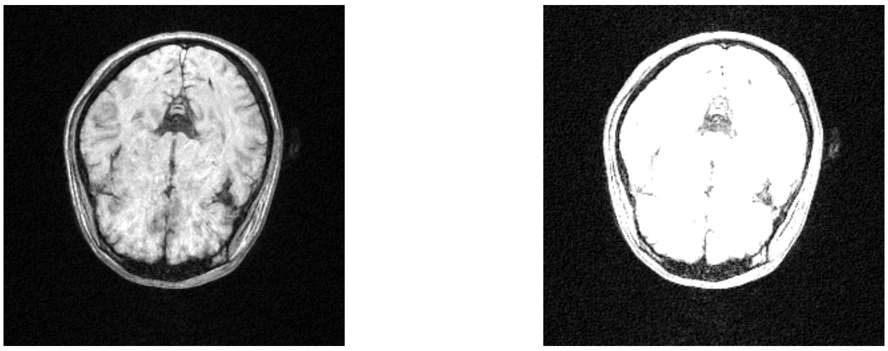
DESCRIPTION

CONTRAST is used to change the differences between light and dark areas to improve image quality. A contrast level of 0 leaves the image unchanged, a level of less than zero reduces the contrast, and a contrast level greater than zero increases the contrast in the image.

EXAMPLE

```
gli> image read medical31.pgm medical
gli> image contrast medical contrast_image 0.5
gli> image display contrast_image pixel
```

FIGURE 7-5 Original Image (left) and Image with Contrast Modified (right)



COPY

Creates a new image by making a copy of an existing image.

FORMAT

IMAGE COPY *source destination*

PARAMETERS

source
Name of the existing image which is to be copied

destination
Name of the output image into which the source image is to be copied.

DESCRIPTION

COPY creates a duplicate of a source image that exists in system memory allocated to GLI. This command can be used to save an image for processing and manipulation without modifying the source image.

Use the IMAGE SHOW command to list all GLI image files in system memory.

EXAMPLE

```
gli> image copy old_image new_image
gli> image show
```

CUT

Copies a region of an existing image into an output image.

FORMAT

IMAGE CUT *source, destination, x-min, x-max, y-min, y-max*

PARAMETERS

source
Name of the existing image from which a portion is to be cut.

destination
Name of the output image into which the source image is to be cut.

x-min, x-max, y-min, y-max
These variables specify the coordinates of a rectangular region of the source image which are to be copied into the destination image. The coordinates are given in pixels.

DESCRIPTION

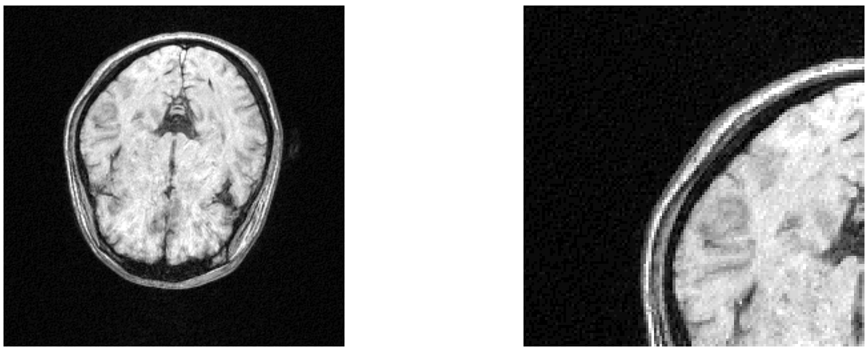
CUT allows a portion of a source image to be copied into a destination image.

Use the IMAGE SHOW command to list all GLI image files in system memory.

EXAMPLE

```
gli> image read medical31.pgm medical
gli> image cut medical cut_image 1 128 1 128
gli> image display cut_image pixel
```

FIGURE 7-6 Original Image (left) and portion of Image (right)



DELETE

Deletes an image from system memory.

FORMAT

IMAGE DELETE *image_name*

PARAMETER

image_name
The name of the image to be removed from memory.

DESCRIPTION

Deletes an image from system memory and frees that memory for other applications.

Use the IMAGE SHOW command to list all GLI image files in system memory.

EXAMPLE

```
gli> image delete my_image
```

DISPLAY

Displays an image on the active device.

FORMAT

IMAGE DISPLAY *image_name transformation*

PARAMETER

image_name
The name of the image to be displayed.

transformation
The transformation for display of the image, either pixel, normal, or ndc.

DESCRIPTION

Displays an image on all active devices. The image must be stored in or converted to one of the formats listed below. (See the Image Format section of this chapter for additional information.)

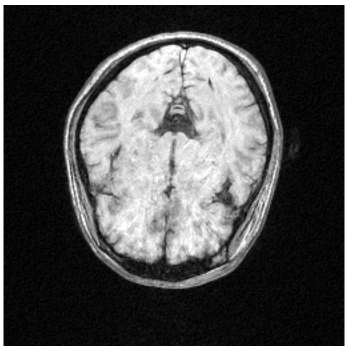
- Portable PixMap (PPM)
- Portable BitMap (PBM)
- Portable ColorMap (PCM)
- Portable GrayMap (PGM)

When an image is displayed with a PIXEL transformation, the image is displayed with each pixel in the image matching a pixel on the screen. When the image is displayed with a NORMAL transformation, the current GKS window is used to display the image. If NDC is specified, the image is displayed in the current GKS viewport. (See the discussion on GKS transformations in Chapter 8 for more information.)

EXAMPLE

```
gli> image read medical31.pgm medical
gli> image display medical pixel
```

FIGURE 7-7 Display an Image



EDGE

Detects edges of an image stored in PGM (Portable GrayMap) format.

FORMAT

IMAGE EDGE *source destination*

PARAMETER

source
The name of the existing image to which the edge detection algorithm is applied.

destination
The name of the resulting image after edge detection is applied.

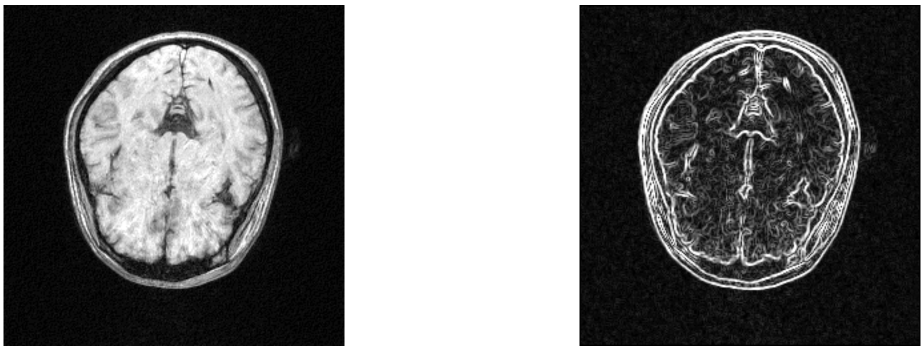
DESCRIPTION

EDGE uses a detection technique that takes the Pythagorean sum of two Sobel gradient operators at 90 degrees to each other. EDGE is used to enhance and display an image that contains ‘edges’ or areas of an image that rapidly transition between light and dark.

EXAMPLE

```
gli> image read medical31.pgm medical
gli> image edge medical edge_image
gli> image display edge_image pixel
```

FIGURE 7-8 Original Image (left) and Image after Edge is applied (right)



ENHANCE

Enhances an image displayed in PGM (Portable GrayMap) format.

FORMAT

IMAGE ENHANCE *source destination level*

PARAMETERS

source
The name of the existing image that is to be enhanced.

destination
The name of the resulting image after enhancement has been applied. The resulting image has the same dimensions as the original image.

level
A level between 0.0 and 1.0 that determines the degree of enhancement to be applied to the source image. A level of 0.0 indicates no enhancement whereas 1.0 specifies maximum enhancement.

DESCRIPTION

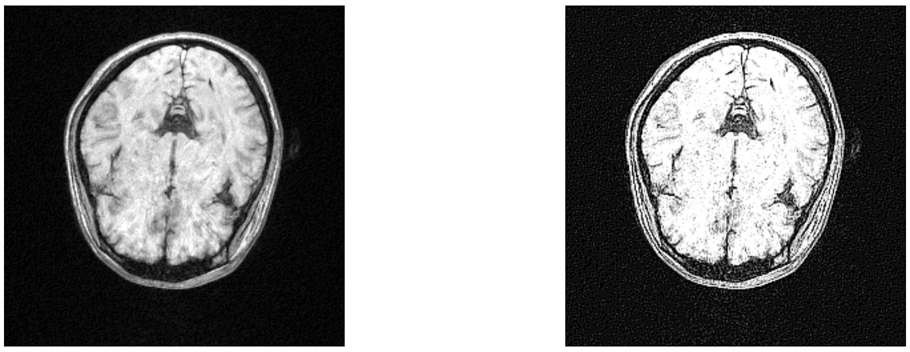
ENHANCE improves the definition between light and dark regions so that these regions are more visible.

ENHANCE uses a technique originally published by J. F. Jarvis, et al., 1976.

EXAMPLE

```
gli> image read medical31.pgm medical
gli> image enhance medical enhance_image 0.75
gli> image display enhance_image pixel
```

FIGURE 7-9 Original Image (left) and Image after Enhance is applied (right)



EXPORT

Stores the contents of an image as a GLI variable.

FORMAT

IMAGE EXPORT *source variable*

PARAMETERS

source
The name of the existing image to be exported.

variable
The GLI variable name (1 to 31 alpha-numeric characters) containing the exported image.

Note: Refer to DEFINE VARIABLE in Chapter 3 for more information on GLI variables.

DESCRIPTION

EXPORT converts an image from the internal image format to a GLI variable which can be used as any other GLI variable. The variable will contain the grayscale values for each pixel in the image within the range 0.0 to 1.0.

EXAMPLE

These commands would store the image *medical.pgm* as a GLI variable named *Z.*

```
gli> image read medical31.pgm medical
gli> image export medical z
```

FLIP

FLIP rotates an image around its horizontal or vertical axes.

FORMAT

IMAGE FLIP *source destination direction*

PARAMETERS

source
The name of the original image to be rotated.

destination
The name assigned to the resulting image after FLIP is applied.

direction
When set to VERTICAL, the image is rotated around its vertical axis. When set to HORIZONTAL, the image is rotated around its horizontal axis.

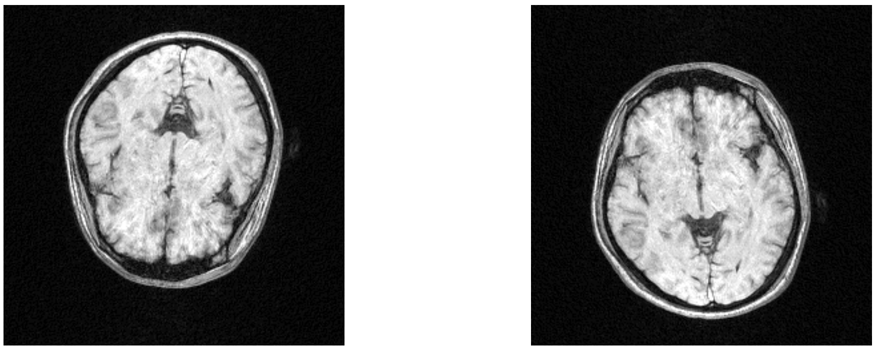
DESCRIPTION

FLIP rotates an image 180 degrees around its horizontal or vertical axis and stores the resulting image using a destination image name.

EXAMPLE

```
  gli> image read medical31.pgm medical
  gli> image flip medical flip_image horizontal
  gli> image display flip_image pixel
```

FIGURE 7-10 Original Image (left) and Image after Flip is applied



GAMMA

Performs a gamma correction on an image to adjust image brightness and contrast.

FORMAT

IMAGE GAMMA *source destination value*

PARAMETERS

source
The name of the existing image to which the gamma corrections is applied.

destination
The name of the resulting gamma corrected image.

value
The value for the gamma correction. A gamma value of 1.0 does not alter the image. A value less than 1.0 darkens the image, whereas a value of greater than 1.0 lightens it. For color images such as PPM or PCM, the gamma correction is applied to the RGB channels and should be greater than 0.

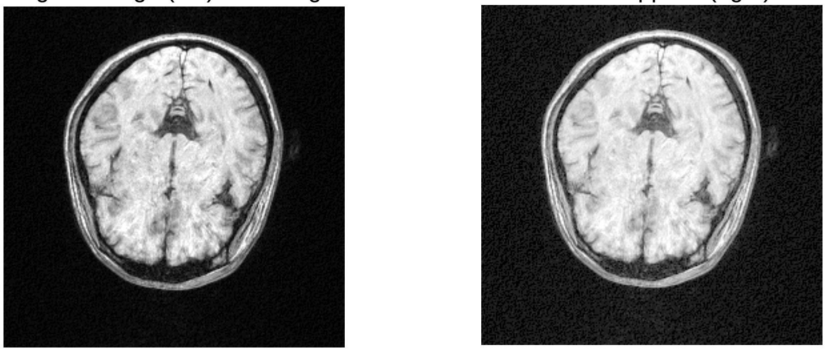
DESCRIPTION

GAMMA changes the intensity of the image in the same manner as the intensity control on a monitor alters the intensity of the image on the screen.

EXAMPLE

```
gli> image read medical31.pgm medical
gli> image gamma medical image_gamma 1.5
gli> image display image_gamma pixel
```

FIGURE 7-11 Original Image (left) and Image after Gamma correction is applied (right)



HISTOGRAM

Smooths the distribution of image data intensities.

FORMAT

IMAGE HISTOGRAM *source destination*

PARAMETERS

source
The name of the existing image to which the histogram equalization is to be applied.

destination
The name of the resulting equalized image.

DESCRIPTION

HISTOGRAM spreads the intensity distribution of an image by equalizing the intensity histograms. Using this function, you can improve the information content of an image.

EXAMPLE

This example reads an image, smooths it with a histogram filter, and then displays the resulting image.

```
gli> image read medical31.pgm medical
gli> image histogram medical histogram_image
gli> image display histogram_image pixel
```

IMPORT

Imports the contents of a GLI variable into an image.

FORMAT

IMAGE IMPORT ***image, variable, x-dimension***

PARAMETERS

image
The name of the image to be created.

variable
The name of the GLI variable containing the image data. (Refer to Chapter 3 for information on GLI variables.)

x-dimension
The width (x axis length) of the original data.

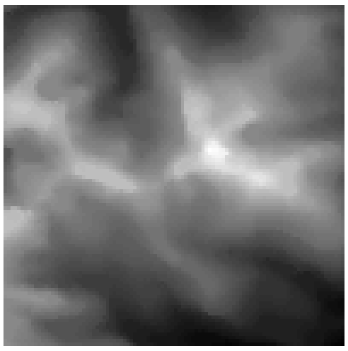
DESCRIPTION

This command produces a PGM format image with a maximum gray level of 255 from the GLI data that has been imported into it.

EXAMPLE

```
gli> read ‘GLI_DEMO’xdemo5 z
gli> image import my_image z 100
gli> image display my_image pixel
```

FIGURE 7-12 Display an Image



INVERT

Inverts an image.

FORMAT

IMAGE INVERT *source destination*

PARAMETERS

source
The name of the image to be inverted.

destination
The name of the resulting inverted image.

DESCRIPTION

INVERT reverses the pixel values of an image. Black pixels will be turned to white, and colors will be reversed or made to look like a photo negative. (e.g. green will become magenta, blue will become yellow, etc.)

EXAMPLE

This example reads an image into GLI and then inverts and displays that image.

```
gli> image read medical31.pgm medical
gli> image invert medical invert_image
gli> image display invert_image pixel
```

MEDIAN

Performs a median cut filtration on an image.

FORMAT

IMAGE MEDIAN *source destination*

PARAMETERS

source
The name of the image to be filtered.

destination
The name of the resulting filtered image.

DESCRIPTION

MEDIAN will produce an image that looks smoother than the original image. It can also remove small errors from an image, and at the same time, the MEDIAN command will modify the original color and gray levels of the original image. A decrease in overall image sharpness may be caused by using a median cut filter.

EXAMPLE

This example reads in an image, performs a median cut on the image, and then displays the result.

```
gli> image read medical31.pgm medical
gli> image median medical median_image
gli> image display median_image pixel
```

NORMALIZE

Enhance an image by spreading the intensity distribution across the full range of values available.

FORMAT

IMAGE NORMALIZE *source destination*

PARAMETERS

source
The name of the image to be normalized.

destination
The normalized image.

DESCRIPTION

NORMALIZE enhances the contrast of an image by spreading the intensity values of the image across the full range of allowed intensities. An image that has used only a portion of the allowed intensities will use the entire range of values after normalization.

EXAMPLE

This example reads in an image, normalizes that image, and then displays the result.

```
gli> image read medical31.pgm medical
gli> image normalize medical normal_image
gli> image display normal_image pixel
```

PBM

Convert an image into a Portable BitMap image.

FORMAT

IMAGE PBM *source destination [level]*

PARAMETERS

source
The name of the image to be converted.

destination
The name of the converted image.

level
An optional parameter which specifies a threshold value for the final output pixels.

DESCRIPTION

The command PBM converts an image which is not in PBM into a PBM image. Threshold levels should be within the range of 0.0 to 1.0, and if you omit this parameter a default value of 0.5 will be used.

EXAMPLE

This example converts an image to PBM format.

```
gli> image pbm old_image pbm_image
```

PCM

Convert an image into a Portable ColorMap image.

FORMAT

IMAGE PCM *source destination [ncolors]*

PARAMETERS

source
The name of the image to be converted.

destination
The name of the converted image.

ncolors
An optional parameter which specifies the maximum number of colors to be used. The value of ncolors should be in the range of 1-256.

DESCRIPTION

PCM converts an image in another format to a Portable ColorMap image format. If you omit the ncolors parameter, a default value of 256 is used. A color quantization using an octree algorithm is performed, if required, to assign colors to the new image.

EXAMPLE

This example converts an image to PCM format.

```
gli> image pcm old_image pcm_image
```

PGM

Converts an image into a Portable GrayMap image.

FORMAT

IMAGE PGM *source destination [maxgray]*

PARAMETERS

source
The name of the image to be converted.

destination
The name of the converted image.

maxgray
Optional maximum value for the gray level in the range of 0-255.

DESCRIPTION

PGM converts an image that is in another format into a Portable GrayMap image. If you omit the maxgray parameter a default value of 255 is used.

EXAMPLE

This example converts an image to PGM format.

```
gli> image pgm old_image pgm_image
```

PPM

Converts an image into a Portable PixMap image.

FORMAT

IMAGE PPM *source destination*

PARAMETERS

source
The name of the image to be converted.

destination
The name of the converted image.

DESCRIPTION

PPM converts an image from any other GLI image format to a Portable PixMap format image.

EXAMPLE

This example converts an image to PPM format.

```
gli> image PPM old_image ppm_image
```

READ

Reads an image from a file.

FORMAT

IMAGE READ *file name*

PARAMETERS

file
The name of the input file to be read.

name
The name of the image to be created.

DESCRIPTION

READ gets an image from a file and creates an image which can then be displayed or modified within GLI. All image formats are read with this command, with no special parameters required.

EXAMPLE

Read an image in from a file and then display that image.

```
gli> image read medical31.pgm medical
gli> image display medical pixel
```

RENAME

Renames an image.

FORMAT

IMAGE RENAME *old_name new_name*

PARAMETERS

old_name
The name by which the image in now known.

new_name
The new name of the image

DESCRIPTION

RENAME changes the name of an image to a new name.

EXAMPLE

Change the name of an image.

```
gli> image rename your_image my_image
```

RGB_GAMMA

Performs a gamma correction on the red, green, and blue values of an image to correct images with poor color.

FORMAT

IMAGE RGB_GAMMA *source destination rvalue gvalue bvalue*

PARAMETERS

source
The name of the image to which the gamma correction is to be applied.

destination
The name of the gamma corrected image.

rvalue
The value for the gamma correction on the red channel.

gvalue
The value for the gamma correction of the green channel.

bvalue
The value for the gamma correction of the blue channel.

DESCRIPTION

The RGB_GAMMA function corrects images with a non-linear color response, thereby improving the color balance of an image. A gamma value of 1 leaves the image unchanged, whereas less than 1 darkens it, and greater than 1 lightens it. The gamma correction value should be greater than zero.

EXAMPLE

Correct the gamma value for an image.

```
gli> image rgb_gamma old_image corrected_ image 0.5 0.2 2.0
```

ROTATE

Rotates an image by an angle.

FORMAT

IMAGE ROTATE *source destination angle [anti-alias]*

PARAMETERS

source
The name of the image to be rotated.

destination
The name of the rotated image.

angle
Specifies an angle in the range 0-360 degrees.

anti-alias
ON-Turns anti-aliasing on.

OFF-Turns anti-aliasing off.

DESCRIPTION

The rotation of the image is done with three shear operations. Setting anti-aliasing to ON will avoid jagged edges and similar artifacts, however, it also means that the original color or gray levels in the image are modified.This modification converts PBM images to PGM images, and converts PCM images to PPM images. To prevent this conversion of image type, anti-aliasing should be set to OFF. The dimensions of the new image may be greater than or equal to the dimensions of the source image, depending on the specified angle.

EXAMPLES

Read in an image and rotate it, then display the result.

```
gli> image read medical31.pgm medical
gli> image rotate medical rotated_image 30
gli> image display rotated_image pixel
```

SCALE

Scales an image to the specified dimensions.

FORMAT

IMAGE SCALE *source destination width height*

PARAMETERS

source
The name of the image to be scaled.

destination
The name of the scaled image.

width
The width of the scaled image, in pixels.

height
The height of the scaled image, in pixels.

DESCRIPTION

SCALE re-sizes an image to the specified width and height. After enlarging an image, you may find it useful to smooth the resulting image with the IMAGE command MEDI-AN.

EXAMPLE

Change the size of an image and display the result on the screen.

```
gli> image read medical31.pgm medical
gli> image scale medical scaled_image 484 484
gli> image display scaled_image pixel
```

SET COLOR

Changes the color table of a Portable ColorMap image.

FORMAT

IMAGE SET COLOR *image colormap*

PARAMETERS

image
The name of an existing image for which the color table is to be changed.

colormap
Specifies one of the colormaps shown in the table below.

TABLE 7-2 PCM colormaps

Name
Uniform
Temperature
Grayscale
Glowing
Rainbow
Geologic
Greenscale
Cyanscale
Bluescale
Magentascale
Redscale

DESCRIPTION

SET COLOR changes the color table of a PCM format image to one of the pre-defined GLI colormaps (see the command GUS SET COLORMAP). The new color table is applied to the image, which can then be saved with the new color map.

EXAMPLE

Change the color table of a PCM image.

```
  gli> image set color pcm_image geologic
```

SHEAR

Shears (or deforms) an image by an angle.

FORMAT

IMAGE SHEAR *direction source destination angle [anti-alias]*

PARAMETERS

xdirection, ydirection
Specifies shearing in the x and y directions.

source
The name of the image to be sheared.

destination
The name of the sheared image.

angle
An angle in the range of -90 to 90 degrees.

anti-alias
ON-Specifies that anti-aliasing is turned on.

OFF-Specifies that anti-aliasing is turned off

DESCRIPTION

Shearing deforms an image by a specified angle in a specified direction. Setting anti-aliasing to ON will avoid jagged edges and similar artifacts, but it will also modify the original colors or gray levels in the image. This change in color or gray levels will cause PCM images to become PPM images, and PBM images will be converted to PGM images

EXAMPLE

Read in an image and then shear the image, and display the result.

```
gli> image read medical31.pgm medical
gli> image shear xdirection medical sheared_image 45
gli> image display sheared_image pixel
```

SHOW

Displays information about all images.

FORMAT

IMAGE SHOW

PARAMETERS

None.

DESCRIPTION

SHOW displays information about all images currently in memory in GLI.

EXAMPLE

Show information about all images currently in GLI.

```
gli> image show
```

WRITE

Writes an image to a file.

FORMAT

IMAGE WRITE *file name*

PARAMETERS

file
The name of the output file to be written.

name
The name of the image to be written to the file

DESCRIPTION

WRITE puts an image into a file for later use. All four GLI image formats are supported: PBM, PCM, PGM, and PPM.

EXAMPLE

Write an image to a file.

```
gli> image write newimage.sav image_name
```

APPENDIX A

GLI Installation Guide

Installing GLI

This appendix will enable you, the system manager, to install the Graphics Language Interpreter (GLI) software on your system. We assume that you are familiar with your system and that all supporting hardware and software have been installed.

Preparing Your System

The installation of GLI on your machine requires the following:

- The GLI media. The label on the media will indicate the range of CPUs and versions of the operating system for which this media will work properly.
- A computer running a supported operating system (VAX/VMS V5.5 or higher, RISC Ultrix V4.4 or higher, SunOS v.4.x, etc.)
- For VMS systems, approximately 15 Megabytes of free disk space.
- For UNIX systems, approximately 25 Megabytes of free disk space.

The loading process will be identical for all types of media except in the name of the device file. Installation will take approximately 10 minutes to complete.

UNIX Installation

To simplify installation, you may find it preferable to load GLI into a common system directory, e.g. */usr/local*. This guide assumes that you chose this location for GLI.

Log on as the superuser (*root*) and change to the */usr/local* directory:

```
# cd /usr/local
```

Physically Load The Media

See your system documentation for instructions on loading media on your tape drive.

Extract the Files from the Tape:

```
# tar xvf devicefile
```

Refer to the system documentation for the appropriate default load device. devicefile on the command line is usually one of the following:

TABLE A-1 Media *device* Definitions

Media Type	*Device*
Digital TK50 or 9-track tape	***/dev/nrmt0h***
Sun 1/4” cartridge	***/dev/nrst8***
Sun 9-track 1600/6250 BPI tape	***/dev/nrmt8***

Edit Your .cshrc File

To start GLI from any location on your system, you must edit your .cshrc file and place in it a source command. For example, depending upon the location of GLI on your system, the .cshrc file might contain a line similar to:

```
% source /usr/local/gli/glisetup.csh
```

The GLISETUP file described below defines the GLI environment and contains device logicals that describe the workstations you are using.

FIGURE A-1

GLISETUP Contents

Note: If you place GLI in a location other than */usr/local/gli* you will need to change the *glisetup*.csh file.

Note Also: If your default graphics workstation is something other than an X Window device, you will need to change the GLI_WSTYPE variable (using *setenv*) to the appropriate workstation type (see Table A-2 below).

For example, to change the device to a VT340 terminal enter:

```
% setenv GLI_WSTYPE 17
```

TABLE A-2

Available Workstations

Type	Workstation
2	GKSM Output Metafile
5	Workstation Independent Segment Storage (WISS)
7, 8	CGM Binary, Clear Text (Computer Graphics Metafile)
16, 17	DIGITAL VT330, VT340 Video Terminal
38	DIGITAL LN03 PLUS
41	VAX UIS
51, 53	HP-GL Graphics Plotter (HP75xx, HP74xx)
61, 62	PostScript, Color PostScript Printer
63, 64	Display PostScript with Compuserve GIF dump (b/w, color)
72	Tektronix 401x Series Terminal
82	Tektronix 42xx Series Terminal
92	DEC LJ250 Companion Color Printer
103, 104	Portable BitMap (72, 75 dpi)
201	TAB 132/15-G Terminal
204	MONTEREY MG200 Display Terminal
207	IBM Personal Computer
210, 211	X Display
214	X Display with Sun rle rasterfile dump
215	X Display with Compuserve GIF dump
217	X Display with frame buffer

Invoke GLI

Invoke the GLI with the GLI command:

```
% gli
```

GLI indicates that it is ready to accept commands by displaying the GLI prompt (gli>):

```
G L I
RISC version 4.5 (UNIX)
patchlevel 4.5.4, 20 Nov 95
```

```
Copyright @ 1986-1995, Josef Heinen, Jochen Werner
Copyright @ 1995, ZAM, Forschungszentrum Juelich GmbH (GR-Software)

Send bugs and comments to J.Heinen@KFA-Juelich.de

gli>
```

If you receive an error message, check the contents of the *glisetup* file (see below) or contact your system administrator.

Unix Startup Command Files

Startup command files define your specific GLI working environment. These files contain definable symbols which specify command aliases, printer types, and other variables for both system-wide and user requirements.

If your system supports more than one GLI user, it may be beneficial to provide several startup files for different user needs.

GLISTARTUP is the template command file distributed with GLI and it is intended for system-wide use. Additionally, a GLIINI command file can be created by each user for their own requirements.

The GLISTARTUP File

When you begin a session, GLI first looks for a system-wide GLISTARTUP file at the location defined by the GLI_HOME environment variable. The default location of GLISTARTUP is the installation directory. This file initializes the system-wide GLI working environment (i.e., defines command abbreviations, sets initial values for graphics attributes, etc.)

Figure A-2 shows the default GLISTARTUP file with some useful symbols.

FIGURE A-2 Contents of the default GLISTARTUP file (glistartup.gli)

```
define symbol auto*plot = gus autoplot
define symbol axe*s = gus axes_2d
define symbol conn*ect = gks open_ws
define symbol cop*y = gks copy_sg
define symbol demo = @ ‘GLI_DEMO’demo.gli
define symbol disc*onnect = gks close_ws
define symbol f*lush = gks update_ws
define symbol gr = grsoft
define symbol gri*d = gus grid
define symbol image_demo = @ ‘GLI_DEMO’image.gli
define symbol pag*e = gks clear_ws
define symbol pl*ot = gus plot
define symbol sou*rce = @
define symbol xdemo = @ ‘GLI_DEMO’xdemo.gli
define symbol tcldemo = @ ’GLI_DEMO’browse.gli
define symbol defc*olors = @ ‘GLI_DEMO’defcolors.gli
define symbol view = import
!
gks open_gks
gks set xform wc
gks set viewport 0.2,0.9 0.2,0.9
gks set asf individual
gks set pmark size 2
gks set pmark type asterisk
gks set text font 3 string
gks set text height 0.027
```

The GLIINI File
This optional file is intended for creating and maintaining the user-specific working environment. The default location for GLIINI is the users' home directory, and it can be created using a text editor (see Chapter 3, Command Procedures). Figure A-3 illustrates a sample script for GLIINI.

FIGURE A-3

Contents of a sample GLIINI file

```
! Define local user symbols
define symbol ls = $ls -la
define symbol state = gks inquire gks_state
define symbol man = help
```

VMS Installation

To simplify installation, you may find it preferable to load GLI into a common system directory, e.g. SYS$COMMON:[000000]. This guide assumes that you chose this location for GLI.

Log on as the ***SYSTEM*** user and change to the common system directory:

```
$ SET DEFAULT SYS$COMMON:[000000]
```

Physically Load the Media
See your system documentation for instructions on loading media on your tape drive.

Extract the Files from the Tape:

```
$ BACKUP devicename [*...]
```

Refer to the system documentation for the appropriate default load device. The devicename on the command line is usually one of those listed in the following table:

TABLE A-3

Media *device* Definitions

Media Type	***Device***
Digital TK50, TK70, TZ30	***MKBx00:*** (i.e., MKB500:)
Digital 9-track tape	***MUA0:***

Once the files have been extracted from the tape you are ready to begin GLI.

Tailoring Your VMS Environment
Before starting GLI, you must setup the GLI working environment on your system by activating the GLISETUP file in the directory where it is installed.

Enter the following, where SYS$COMMON is the directory path to GLI on your system:

```
$ SET DEFAULT SYS$COMMON:[GLI]
$ @GLISETUP
```

Once GLI is installed, place the following command in your system-wide login command procedure (SYS$MANAGER:SYLOGIN.COM):

```
$ @SYS$COMMON:[GLI]GLISETUP
```

The GLISETUP file defines the GLI environment and contains device logicals that describe the workstations you are using.

FIGURE A-4 Partial Contents of the GLISETUP File

```
$!
$! Define foreign commands and symbols
$!
$ gli :== $gli_home:gli
$ glid*ecgks :== $gli_home:glidecgks
$ cgmview :== $gli_home:cgmview
$ gli_libs :== gli_home:libgr/lib,libgli/lib,libsight/lib,libimage/lib,
libgus/lib,libcrtl/lib,libfrtl/lib,libgks/lib,glilink/opt
$!
$! Define GR/GR3 logical name
$!
$ define/nologgrgks GLIGKS
$!
$! Define GLI logical names
$!
$ define/nologgli_pl $plotter
$ define/nologgli_lw $laser_printer
```

Note: If your default graphics workstation is something other than an X Window device, change the GLI_WSTYPE variable (using DEFINE) to the appropriate workstation type (see Table A-4 below).

For example, to change the device to a VT340 terminal enter:

```
$ DEFINE GLI_WSTYPE 17
```

TABLE A-4 Available Workstations

Type	Workstation
2	GKSM Output Metafile
5	Workstation Independent Segment Storage (WISS)
7, 8	CGM Binary, Clear Text (Computer Graphics Metafile)
16, 17	DIGITAL VT330, VT340 Video Terminal
38	DIGITAL LN03 PLUS
41	VAX UIS
51, 53	HP-GL Graphics Plotter (HP75xx, HP74xx)
61, 62	PostScript, Color PostScript Printer
63, 64	Display PostScript with Compuserve GIF dump (b/w, color)
72	Tektronix 401x Series Terminal
82	Tektronix 42xx Series Terminal
92	DEC LJ250 Companion Color Printer
103, 104	Portable BitMap (72, 75 dpi)
201	TAB 132/15-G Terminal
204	MONTEREY MG200 Display Terminal
207	IBM Personal Computer
210, 211	X Display
214	X Display with Sun rle rasterfile dump
215	X Display with Compuserve GIF dump
217	X Display with frame buffer

Invoke GLI

Invoke the GLI with the command:

```
$ GLI
```

GLI indicates that it is ready to accept commands by displaying the GLI prompt (gli>):

```
G L I
VAX version 4.5 (VMS)
patchlevel 4.5.4, 20 Nov 95

Copyright @ 1986-1995, Josef Heinen, Jochen Werner
Copyright @ 1995, ZAM, Forschungszentrum Juelich GmbH (GR-Software)

Send bugs and comments to J.Heinen@KFA-Juelich.de

gli>
```

If you receive an error message, check the contents of the GLISETUP file (see below) or contact your system administrator.

VMS Startup Command Files

Startup command files define your specific GLI working environment. These files contain user-definable symbols which specify command aliases, printer types, and other variables specific to your requirements.

If your system supports more than one GLI user, it may be beneficial to provide several startup files for different user needs.

GLISTARTUP is the template command file distributed with GLI and it is intended for system-wide use. Additionally, a GLIINI command file can be created by each user for their own requirements.

The GLISTARTUP File

When you begin a session, GLI first looks for a system-wide GLISTARTUP file at the location defined by the GLI_RC logical name. The default location of GLISTARTUP is the installation directory. This file initializes the system-wide GLI working environment.

Figure A-2 shows the default GLISTARTUP file with some useful symbols.

FIGURE A-5 Contents of the default GLISTARTUP file (glistartup.gli)

```
define symbol auto*plot = gus autoplot
define symbol axe*s = gus axes_2d
define symbol conn*ect = gks open_ws
define symbol cop*y = gks copy_sg
define symbol demo = @ ‘GLI_DEMO’demo.gli
define symbol disc*onnect = gks close_ws
define symbol f*lush = gks update_ws
define symbol gr = grsoft
define symbol gri*d = gus grid
define symbol image_demo = @ ‘GLI_DEMO’image.gli
define symbol pag*e = gks clear_ws
define symbol pl*ot = gus plot
define symbol sou*rce = @
define symbol xdemo = @ ‘GLI_DEMO’xdemo.gli
define symbol tcldemo = @ ’GLI_DEMO’browse.gli
define symbol defc*olors = @ ‘GLI_DEMO’defcolors.gli
define symbol view = import
!
gks open_gks
gks set xform wc
gks set viewport 0.2,0.9 0.2,0.9
gks set asf individual
gks set pmark size 2
gks set pmark type asterisk
gks set text font 3 string
gks set text height 0.027
```

The GLIINI File

This optional file is intended for creating and maintaining the user-specific working environment. The default location for GLIINI is the users’ login directory, and it can be created using a text editor (see Chapter 3, Command Procedures). The figure below illustrates a sample script for GLIINI.

FIGURE A-6

Contents of a sample GLIINI file

```
! Define local user symbols
define symbol dir = $dir/size
define symbol state = gks inquire gks_state
define symbol dcl = $
```

Run the GLI Demonstration Programs

GLI includes demonstration programs for terminal and X Window systems.These programs can be used to test your GLI installation and demonstrate the general capabilities of the product.

Note: If the GLI demonstrations fail to execute properly, make sure GLI is in the proper operating state (type `initialize` or `init` at the GLI command prompt).

Image Demonstration

GLI includes an imaging module that displays, filters and manipulates raster images in a variety of formats. The imaging module can also be used to animate a sequence of image files. To view a demonstration, enter the following command at the GLI command prompt or refer to the Image chapter in this manual.

```
gli> image_demo
```

ASCII Terminal

The next demonstration program may be executed from an ASCII terminal device or an X Windows device by entering:

```
gli> demo
```

A keyboard-driven demonstration interface appears as in the figure below.

FIGURE A-7 Demonstration Menu for Terminal Devices

Use the Up and Down arrow keys to move vertically through the menu and use the RETURN key to move horizontally.

- Choose any item to see individual demonstration files.
- Choose *Movie* to see a demonstration of all the files. Enter RETURN after each graph to view the next one

- Press the Do key (PF1, or Esc + 1 on Sun keyboards) to execute your choice.
- Press the Exit key (PF4, or Esc + 4 on Sun keyboards) to exit the demonstration interface.

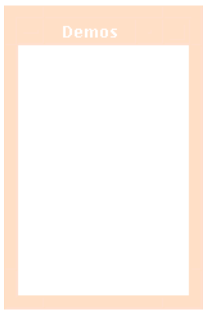
X Windows Terminal

The second demonstration program supports X Window devices only (you must have the X Window software installed on your system). To activate this demo, type:

```
gli> xdemo
```

A pop-up point-and-click demonstration interface appears as in this figure.

FIGURE A-8 Demonstration Menu for X Window Devices



Use the mouse to select the demonstrations:

- Choose any item to see individual demonstration files.
- Choose *Movie* to see the demonstration files in continuous series.
- Choose *Finish* to exit the GLI Demonstration Interface.

The Introduction and Basic Usage chapters in this manual include information and examples that illustrate GLI’s general features and capabilities. For a look at more advanced GLI applications, review the Design Samples in Appendix C.

Quitting GLI

Enter the QUIT command whenever you are finished using the GLI system. QUIT saves your GLI session to the GLI.JOU file.

```
gli> quit
```

Enter the EXIT command to terminate GLI without creating the journal file and saving your session instructions.

```
gli> exit
```

APPENDIX B

GLI Font Support

GLI supports hardware and software fonts to provide versatility and flexibility for a variety of applications. The hardware fonts supported by GLI are resident in many PostScript and X Window devices (see Table B-2). Subject to the limitations of the device itself, hardware fonts can yield higher quality output, but they may also require greater computing resources and therefore be more time consuming to generate. GLI's software fonts produce results independent of the device and generally provide more flexibility.

GKS commands provide near total control over all characteristics of the text string. GUS utilities automate the text capability of GLI. (See GKS and GUS chapters for a detailed discussion of text.)

Software Fonts

GLI's software fonts are determined by the GKS installed. GLI's implementation of GKS supports 22 software fonts, known as Hershey fonts. These fonts were digitized by Dr. A.V. Hershey of the Naval Surface Weapons Laboratory and have been supplied by the National Bureau of Standards. The Hershey fonts are not mono-spaced; each character is not necessarily the same size as its character box. In most cases, the character box is larger than the character, although some characters (for example, those with descenders) may be drawn outside its box.

Several GKS text commands are appropriate for use only with software fonts at stroke precision. These commands are: GKS SET TEXT EXPFAC, GKS SET TEXT PATH, and GKS SET TEXT UPVEC. Refer to the GKS chapter for a complete discussion of text commands.

By default, GLI uses hardware font 3 (Helvetica) at string precision. Software fonts are specified at stroke precision using a negative font identifier as listed in Table B-1. To specify the Simplex Roman software font, the command would be:

```
  gli> gks set text fontprec -3 stroke
```

TABLE B-1 Software Fonts

Font Identifier	Font Name
-1	Cartographic Roman
-2	Cartographic Greek
-3	Simplex Roman

-4	Simplex Greek
-5	Simplex Script
-6	Complex Roman
-7	Complex Greek
-8	Complex Italic
-9	Triplex Roman
-10	Triplex Greek
-11	Triplex Italic
-12	Duplex Roman
-13	Complex Script
-14	Complex Cyrillic
-15	Roman
-16	Italic
-17	Gothic German
-18	Gothic English
-19	Gothic Italian
-20	Scientific
-21	Symbolic
-22	Cartographic

Hardware Fonts

Table B-2 lists the fonts resident in many PostScript and X Window devices supported by GLI. Since hardware fonts are dependant upon the capability of the device itself, the specific fonts available and their characteristics such as resolution may vary between devices. The generation of hardware fonts at high precision may be time consuming for some computer systems.

Refer to the GKS and GUS chapters for a complete discussion of commands that control font selection and text. By default, GLI uses hardware font 3 (Helvetica) at string precision. To specify another hardware font, the command would be similar to:

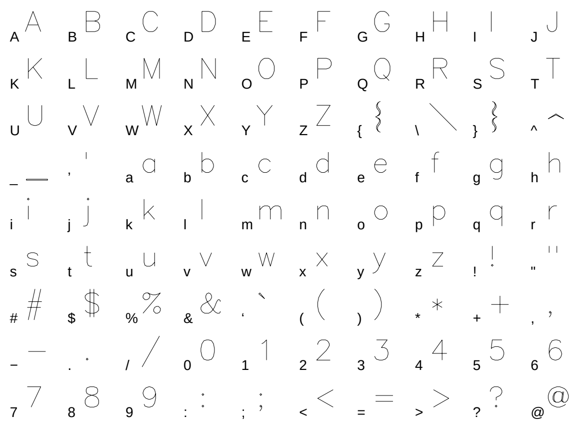
```
gli> gks set text fontprec 5 string
```

When hardware fonts are required, the precision should be set to string.

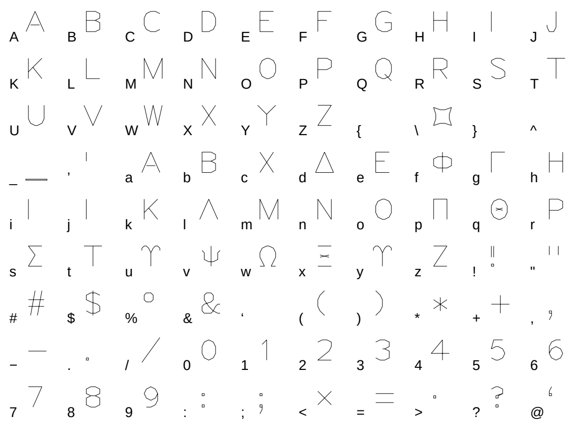
Certain GLI text commands have no effect when used at string precision. The GLI text commands supported only by stroke precision include: GKS SET TEXT EXPFAC, GKS SET TEXT PATH, and GKS SET TEXT UPVEC.

Note: X Windows software for Hewlett-Packard 9000 systems does not support ITC Avant Garde Gothic, ITC Lubalin Graph and ITC Souvenir.

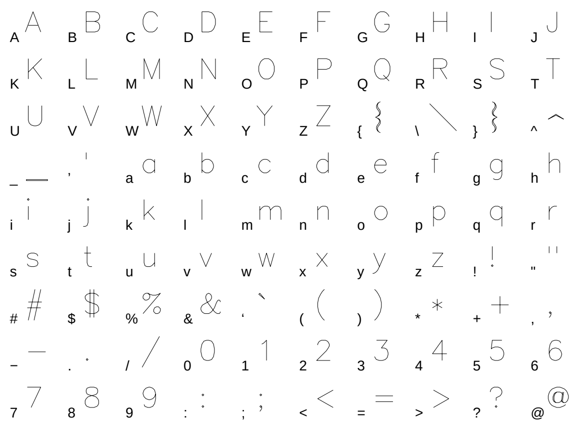
Font −1 Cartographic Roman



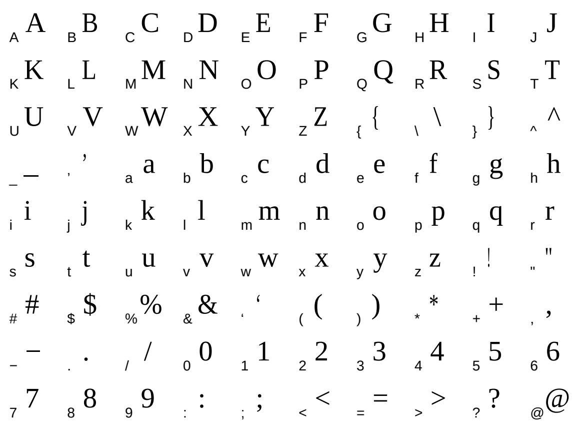
Font −2 Cartographic Greek



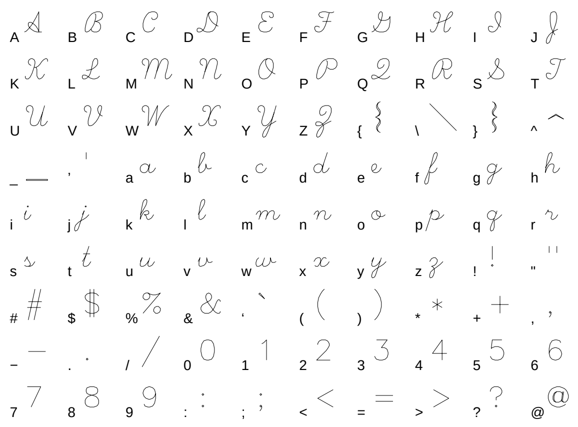
Font −3 Simplex Roman



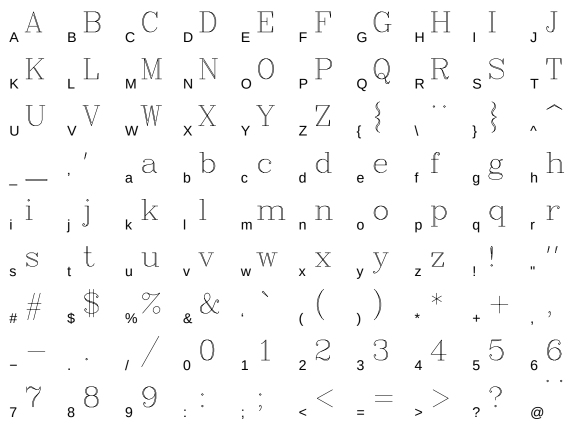
Font 4 Lubalin Graph−Book



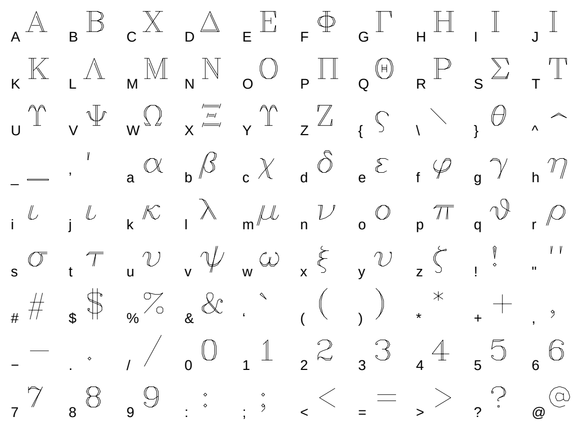
Font −5 Simplex Script



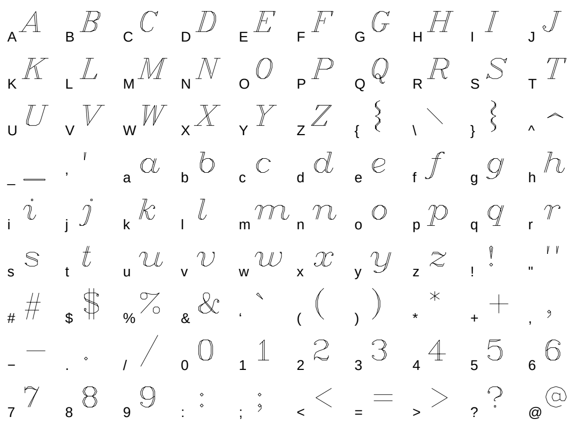
Font −6 Complex Roman



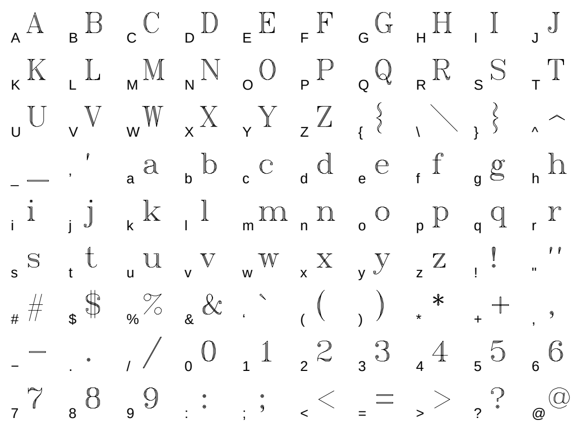
Font −7 Complex Greek



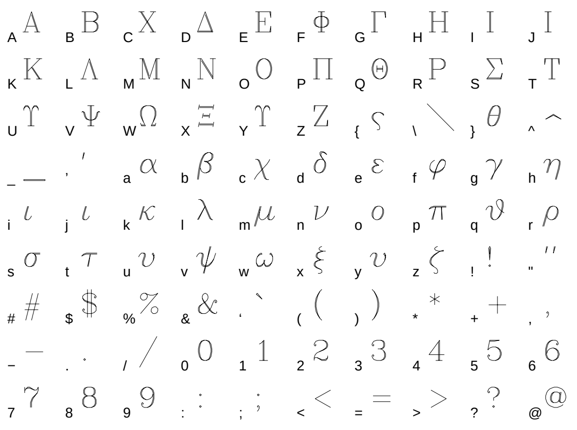
Font −8 Complex Italic



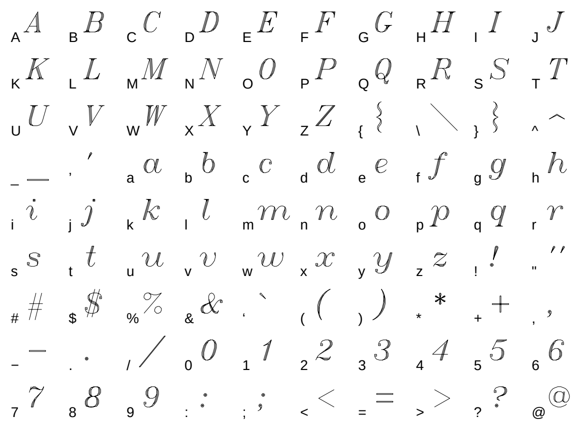
Font −9 Triplex Roman



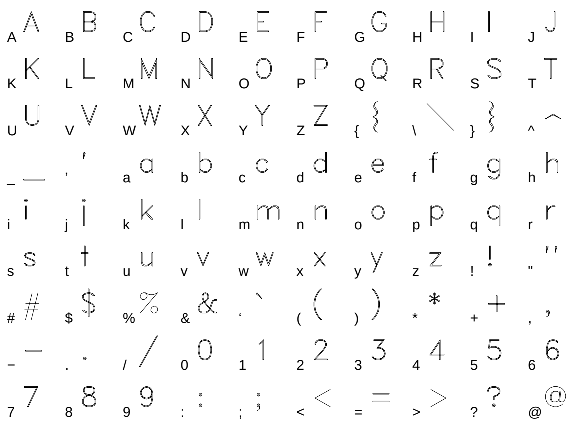
Font −10 Triplex Greek



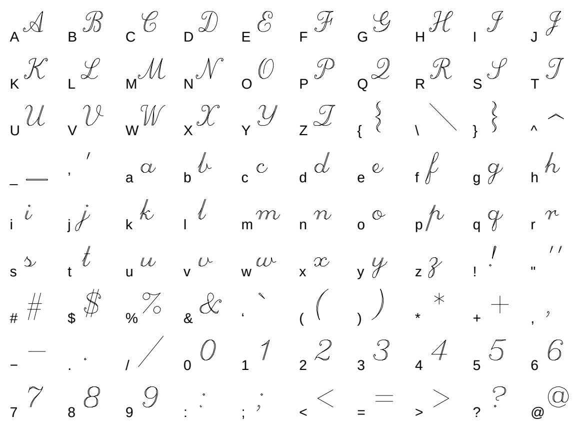
Font −11 Triplex Italic



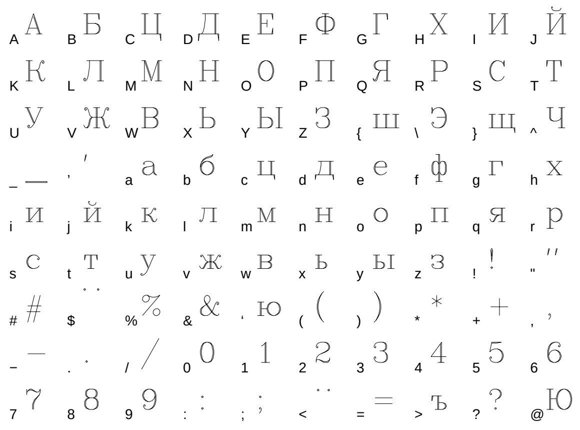
Font −12 Duplex Roman



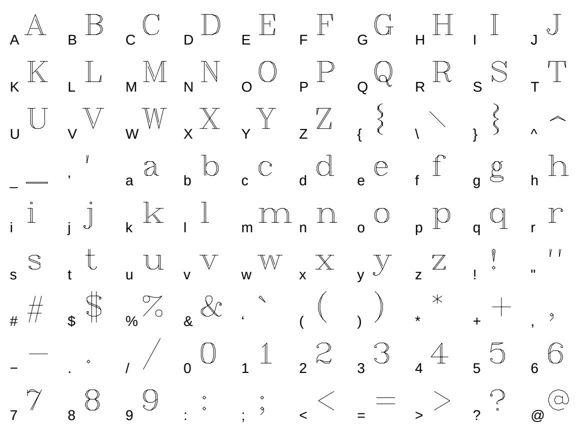
Font −13 Complex Script



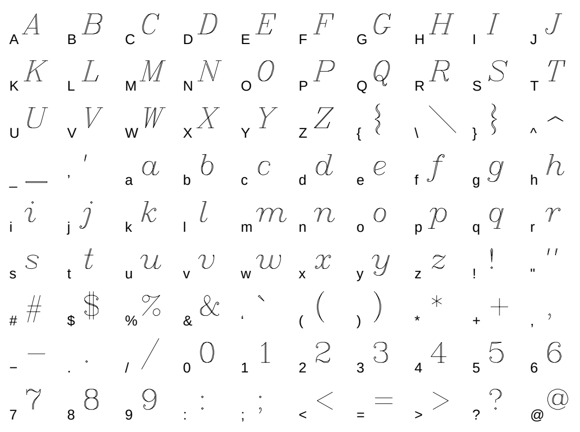
Font −14 Complex Cyrillic



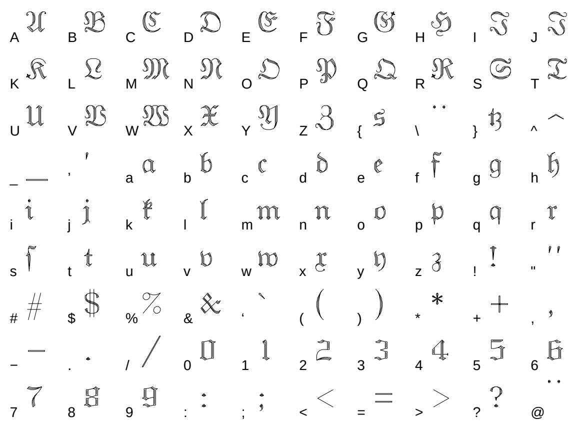
Font −15 Roman



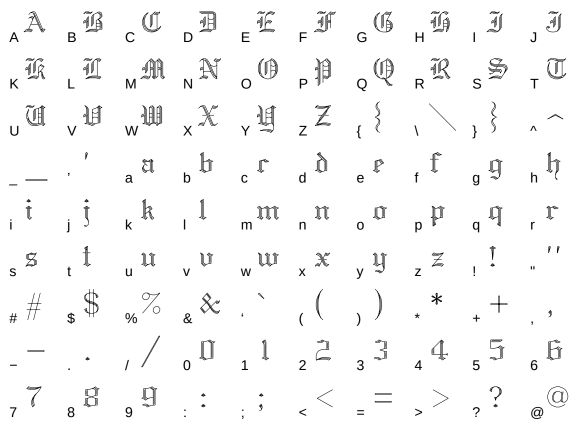
Font −16 Italic



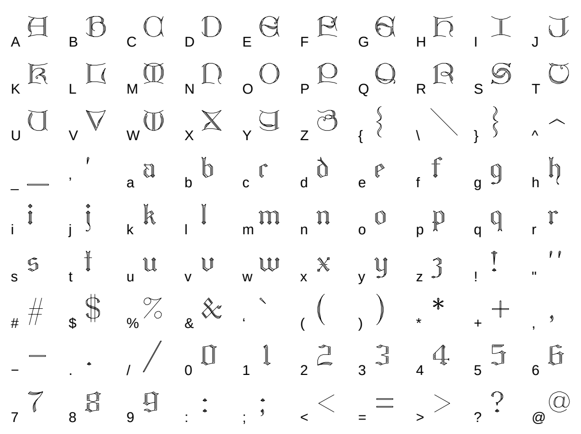
Font −17 Gothic German



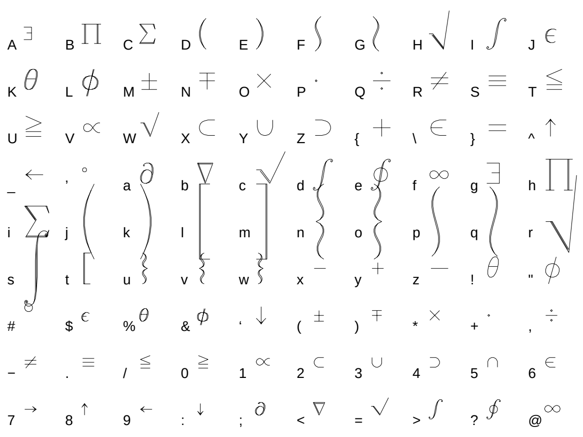
Font −18 Gothic English



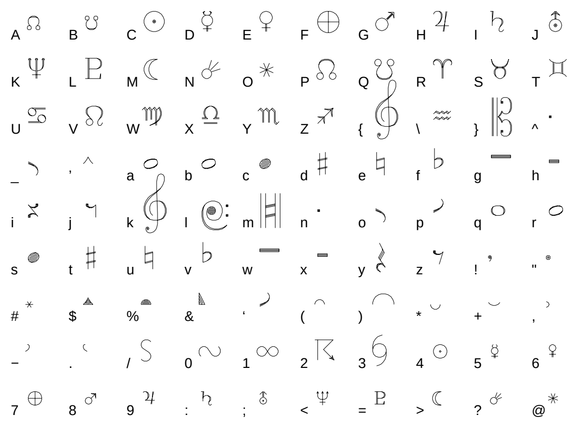
Font −19 Gothic Italian



Font −20 Scientific



Font −21 Symbolic



Font −22 Cartographic

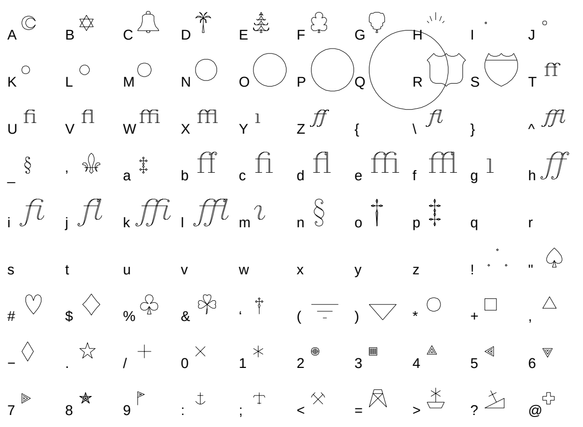


TABLE 2-2 Hardware Fonts

Font Identifier	Font Name
1	Avant Garde - Book
2	Courier
3	Helvetica
4*	Lubalin Graph - Book
5	New Century School Book - Roman
6*	Souvenir - Light
7	Symbol
8	Times - Roman
9	Avant Garde - Demi
10	Courier - Bold
11	Helvetica - Bold
12*	Lubalin Graph - Demi
13	New Century School Book - Bold
14*	Souvenir - Demi
15	(Font number reserved)
16	Times - Bold
17	Avant Garde - Book - Oblique
18	Courier - Oblique
19	Helvetica - Oblique
20*	Lubalin Graph - Book - Oblique
21	New Century School Book - Italic
22*	Souvenir - Light - Italic
23	(Font number reserved)
24	Times - Italic
25	Avant Garde - Demi - Oblique
26	Courier - Bold - Oblique
27	Helvetica - Bold - Oblique
28*	Lubalin Graph - Demi - Oblique
29	New Century School Book - Bold - Italic
30*	Souvenir - Demi - Light
31	(Font number reserved)
32	Times - Bold - Italic

Note: The referenced (*) fonts listed in table B-2 are not supported by the Apple Laser Writer printer.

Font 1 Avant Garde−Book

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 2 Courier

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 3 Helvetica

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 4 Lubalin Graph−Book

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 5 New Century Schoolbook−Roman

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 6 Souvenir−Light

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 7 Symbol

A Α	B Β	C Χ	D Δ	E Ε	F Φ	G Γ	H Η	I Ι	J ϑ
K Κ	L Λ	M Μ	N Ν	O Ο	P Π	Q Θ	R Ρ	S Σ	T Τ
U Υ	V ς	W Ω	X Ξ	Y Ψ	Z Ζ	{ {	\ ∴	} }	^ ⊥
_ _	' ∍	a α	b β	c χ	d δ	e ε	f ϕ	g γ	h η
i ι	j φ	k κ	l λ	m μ	n ν	o ο	p π	q θ	r ρ
s σ	t τ	u υ	v ϖ	w ω	x ξ	y ψ	z ζ	! !	" ∀
# #	$ ∃	% %	& &	` ‾	((	))	* ∗	+ +	, ,
- −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ ≅

Font 8 Times−Roman

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
- −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 9 Avant Garde−Demi

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
- −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 10 Courier−Bold

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
- −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 11 Helvetica−Bold

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
- −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 12 Lubalin Graph−Demi

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
- −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 13 New Century Schoolbook−Bold

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 14 Souvenir−Demi

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 16 Times−Bold

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 17 Avant Garde−Book−Oblique

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 18 Courier−Oblique

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 19 Helvetica−Oblique

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 20 Lubalin Graph−Book−Oblique

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 21 New Century Schoolbook−Italic

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 22 Souvenir−Light−Italic

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 24 Times−Italic

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ _	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	` ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 25 Avant Garde−Demi−Oblique

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ —	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	' ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 26 Courier−Bold−Oblique

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ —	' ′	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	' ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 27 Helvetica−Bold−Oblique

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ —	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	' ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 28 Lubalin Graph−Demi−Oblique

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ —	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	' ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 29 New Century Schoolbook−Bold−Italic

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ —	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	' ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 30 Souvenir−Demi−Italic

A A	B B	C C	D D	E E	F F	G G	H H	I I	J J
K K	L L	M M	N N	O O	P P	Q Q	R R	S S	T T
U U	V V	W W	X X	Y Y	Z Z	{ {	\ \	} }	^ ^
_ —	' ’	a a	b b	c c	d d	e e	f f	g g	h h
i i	j j	k k	l l	m m	n n	o o	p p	q q	r r
s s	t t	u u	v v	w w	x x	y y	z z	! !	" "
# #	$ $	% %	& &	' ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 0	1 1	2 2	3 3	4 4	5 5	6 6
7 7	8 8	9 9	: :	; ;	< <	= =	> >	? ?	@ @

Font 32 Times−Bold−Italic

A ***A***	B ***B***	C ***C***	D ***D***	E ***E***	F ***F***	G ***G***	H ***H***	I ***I***	J ***J***
K ***K***	L ***L***	M ***M***	N ***N***	O ***O***	P ***P***	Q ***Q***	R ***R***	S ***S***	T ***T***
U ***U***	V ***V***	W ***W***	X ***X***	Y ***Y***	Z ***Z***	{ ***{***	\ \	} ***}***	^ ^
_ –	' ’	a ***a***	b ***b***	c ***c***	d ***d***	e ***e***	f ***f***	g ***g***	h ***h***
i ***i***	j ***j***	k ***k***	l ***l***	m ***m***	n ***n***	o ***o***	p ***p***	q ***q***	r ***r***
s ***s***	t ***t***	u ***u***	v ***v***	w ***w***	x ***x***	y ***y***	z ***z***	! ***!***	" ″
# #	$ $	% %	& &	' ‘	((	))	* *	+ +	, ,
− −	. .	/ /	0 ***0***	1 ***1***	2 ***2***	3 ***3***	4 ***4***	5 ***5***	6 ***6***
7 ***7***	8 ***8***	9 ***9***	: :	; ;	< <	= =	> >	? ?	@ @

APPENDIX C

Design Samples

GLI contains example programs in the GLI/DEMO directory. These examples are available to the user and are listed here.

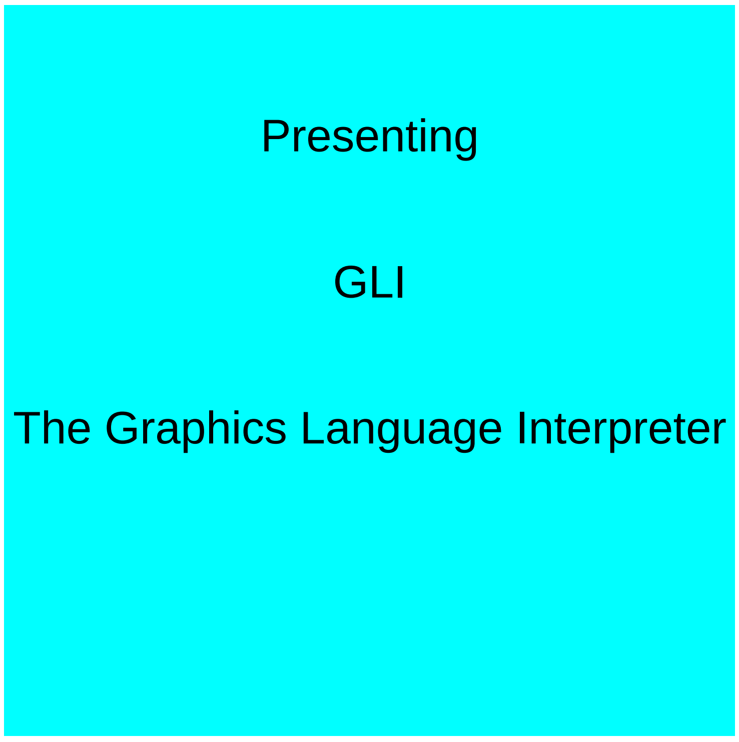
Demonstration Menu (demo.fdv)

```
!N0(Copyright @ 1986-1993 by Josef Heinen,DO-PF1,,,EXIT-PF4)

Move the cursor to the desired item and then press the DO or the ENTER key.

        Main Title !^1           Data Fitting !^2
        Columbia !^3             Enlargement !^4
        Sine Wave !^5            Earnings !^6
        Spectrometer !^7         Pyrolyzation !^8
        Y-log !^9                Regression !^10
        Surface !^11             Competition !^12
        FFT's !^13               3-D Curves !^14
        XY-Graph !^15            Text Title !^16
        Fonts !^17               Characters !^18
        Movie !^26

Sun Keyboard Users use ESC 1 (DO) and ESC 4 (EXIT)
```

demo1.gli

```
!This program generates the main title for the GLI demos
page
gks set fill color cyan
gks set fill int_style solid
gks set xform wc
gks set viewport 0.1 0.9 0.1 0.9
gks set window 0 1 0 1
x := 0 0 1 1
y := 0 1 1 0
gks fill_area x y
gks set text height 0.036/0.8
gks set text align center normal
gks text .5 .8 Presenting
gks text .5 .6 GLI
gks text .5 .4 The Graphics Language Interpreter
del var *
gks set text align normal normal
gks set text height 0.018
gks set fill int_style hollow
gks set fill color black
```

FIGURE C-1 Demo1.gli

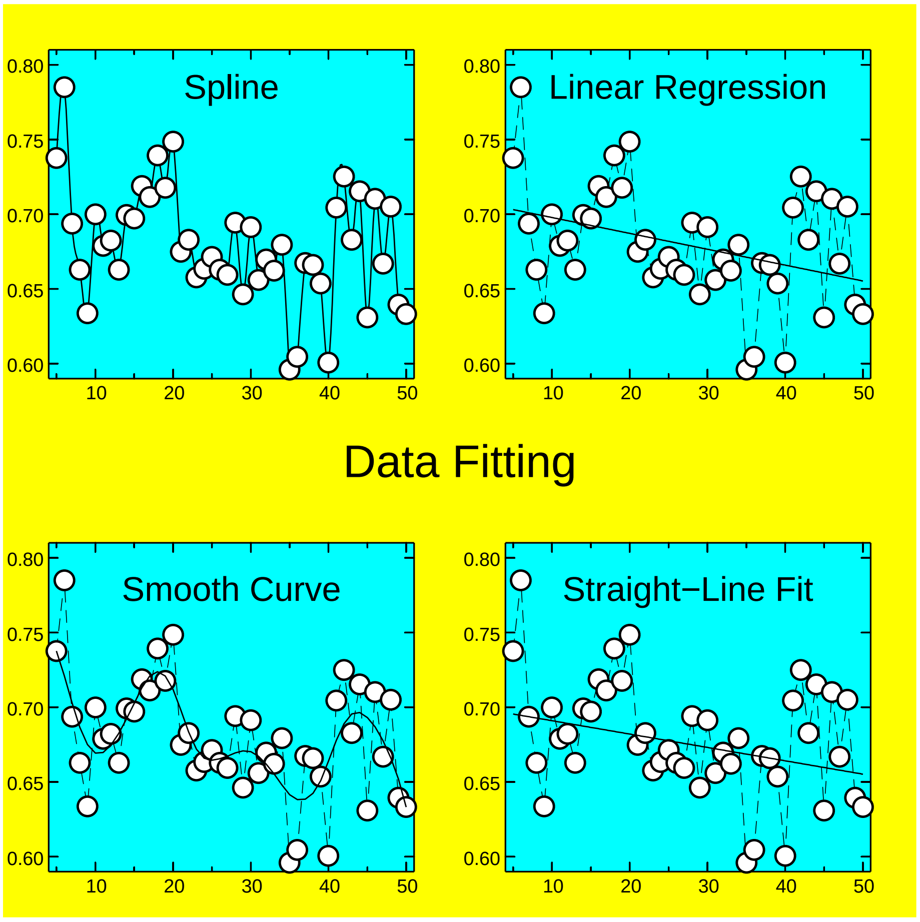


demo2.gli

```
         page
         gks set text font 3 string
         gks set pmark color black
         a := 4 4 51 51 4
         b := 0.59 0.81 0.81 0.59 0.59

  ! background
         gks set xform ndc
         x := 0 0 1 1 0
         y := 0 1 1 0 0
         gks set fill color_index yellow
         gks set fill int_style solid
         gks fill x y
         gks set xform wc
         gks set fill color_index cyan

  ! smooth
         gks set viewport 0.05 0.45 0.05 0.41
         gks set window 4 51 0.59 0.81
         gks fill a b
         gks set pline linewidth 2
         gks set text height 0.015
         gus axes_2d 5 0.05 4 0.59 2 1 0.005
         gus axes_2d 5 0.05 51 0.81 -2 -1 -0.005
         gks set pmark type circle
         gks set pline linewidth 1
         gks set pline linetype dashed
         read ‘GLI_DEMO’demo2.dat x y

         gus polyline x y
         gus polymark x y
         smooth y
         gks set pline linewidth 1.7
         gks set pline linetype solid
         gus polyline x y

  ! spline
         gks set viewport 0.05 0.45 0.59 0.95
         gks set window 4 51 0.59 0.81
         gks fill a b
         gks set pline linewidth 2
         gks set text height 0.015
         gus axes_2d 5 0.05 4 0.59 2 1 0.005
         gus axes_2d 5 0.05 51 0.81 -2 -1 -0.005

         gks set pline linewidth 1
         gks set pline linetype dashed
         gks set pmark type circle
         read ‘GLI_DEMO’demo2.dat x y
```

```
        gks set pline linewidth 1.7
        gks set pline linetype solid
        gus spline x y
        gus polymark x y


! linear regression
        gks set viewport 0.55 0.95 0.59 0.95
        gks set window 4 51 0.59 0.81
        gks fill a b
        gks set pline linewidth 2
        gks set text height 0.015
        gus axes_2d 5 0.05 4 0.59 2 1 0.005
        gus axes_2d 5 0.05 51 0.81 -2 -1 -0.005
        gks set pmark type circle
        gks set pline linewidth 1
        gks set pline linetype dashed
        gus polyline x y
        gus polymark x y
        gks set pline linewidth 1.7
        gks set pline linetype solid
        gus linreg x y

! linear fit
        gks set viewport 0.55 0.95 0.05 0.41
        gks set window 4 51 0.59 0.81
        gks fill a b
        gks set pline linewidth 2
        gks set text height 0.015
        gus axes_2d 5 0.05 4 0.59 2 1 0.005
        gus axes_2d 5 0.05 51 0.81 -2 -1 -0.005
        gks set pmark type circle
        gks set pline linewidth 1
        gks set pline linetype dashed
        gus polyline x y
        gus polymark x y
        gks set pline linetype solid
        gks set pline linewidth 1.7
        gus linfit x y

! text
        gks set xform ndc
        gks set text font 3 string
        gks set text align centre half
        gks set text height 0.027
        gks text 0.25 0.36 Smooth Curve
        gks text 0.25 0.91 Spline
        gks text 0.75 0.91 Linear Regression
        gks text 0.75 0.36 Straight-Line Fit
        gks set text height 0.036
        gks text 0.5 0.5 Data Fitting
        gks set text font -1 string

        del var *
```

FIGURE C-2 demo2.gli

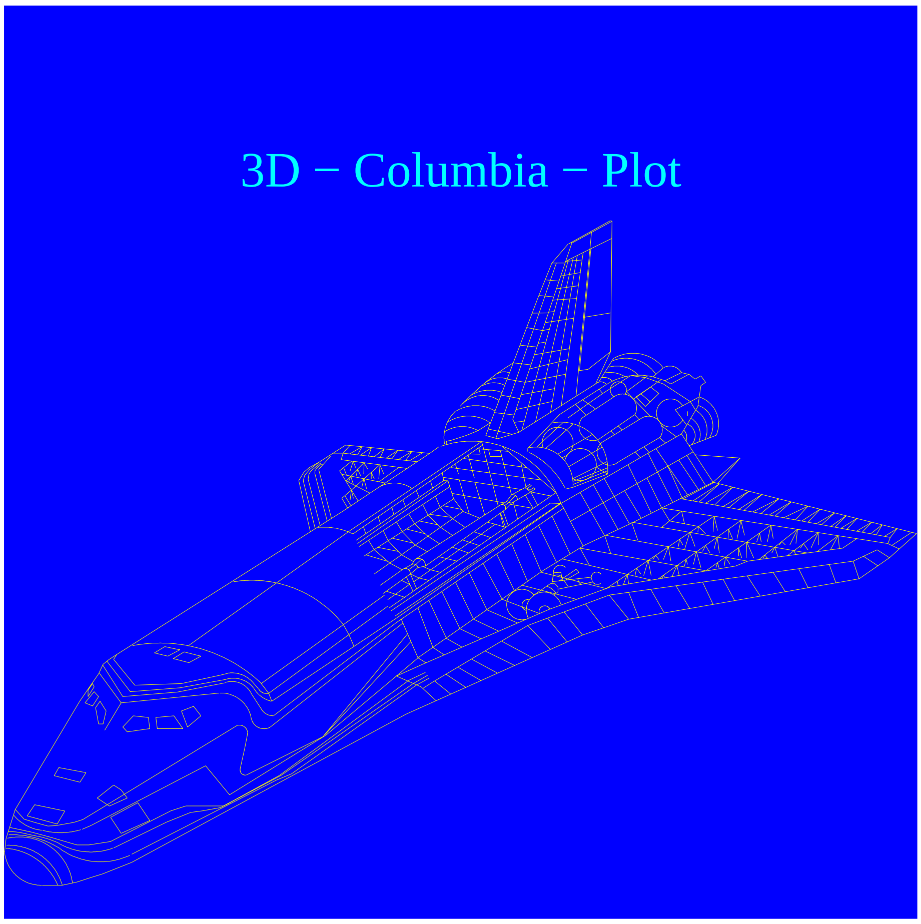


demo3.gli

```
 page
 gks set text font 8 string
 gks set text color cyan
 gks set text align centre half
 gks set text height 0.036

!Background
 gks set xform ndc
 gks set fill color blue
 gks set fill int_style solid
 a := 0 0 1 1 0
 b := 0 1 1 0 0
 gks fill a b
 gks text 0.5 0.82 3D - Columbia - Plot
 gks set xform wc

 read ‘GLI_DEMO’demo3.dat x y
 gks set pline color yellow
 gks set pline linewidth 0.5
 gks set viewport 0 1 0 1
 gks set window 0 13 0 13
 gks polyline x y
 gks set pline color black

 del var *
 gks set text font -1 string
 gks set text color black
```

FIGURE C-3

Demo3.gli

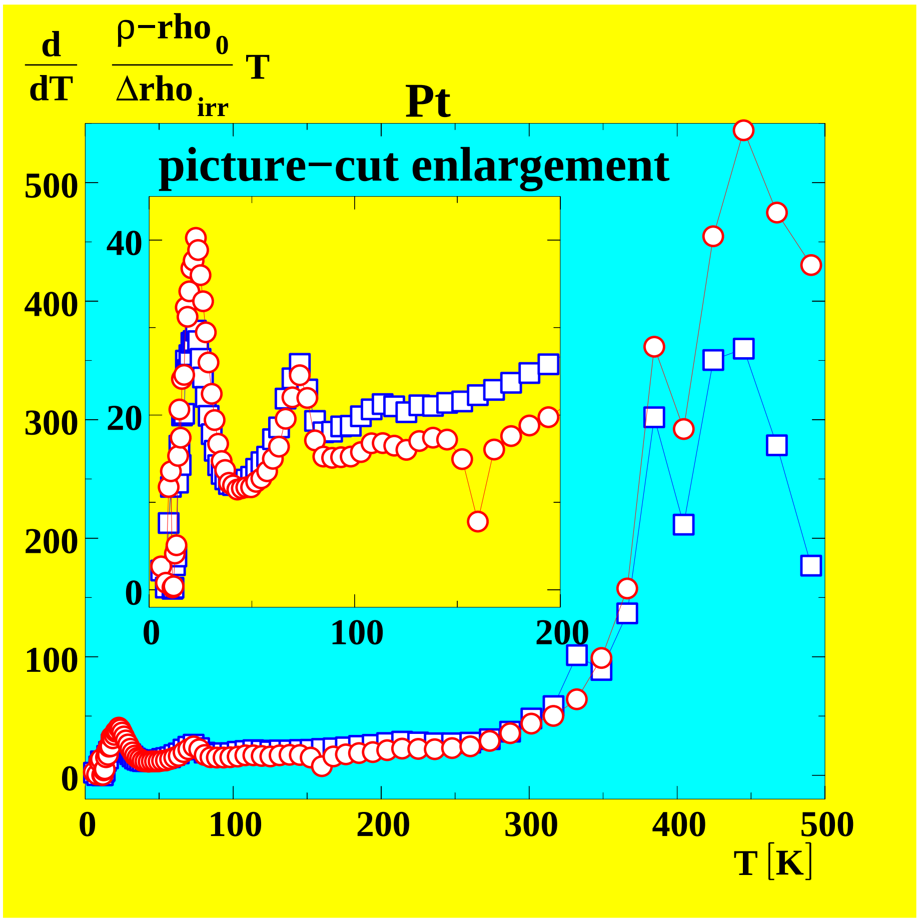


demo4.gli

```
page
gks set xform ndc
gks set text font 16 string
gks set text upvec 0 1
gks set text height 0.027
gks set text align left bottom
gks set text color_index black
gks set pline color_index black

! Background
gks set fill color_index yellow
gks set fill int_style solid
x:= 0 0 1 1 0
y:= 0 1 1 0 0
gks fill x y

! 1.Window
gks set fill color_index cyan
a:= 0.09 0.09 0.90 0.90 0.09
b:= 0.13 0.87 0.87 0.13 0.13
gks fill a b
gks set xform wc

!define Symbols
def var xmin = 0
def var xmax = 500
def var ymin = -20
def var ymax = 550
READ 'GLI_DEMO'demo4_1.dat X1,Y1
READ 'GLI_DEMO'demo4_2.dat X2,Y2


gks set viewport 0.09 0.9 0.13 0.87
gks set window xmin xmax ymin ymax
gus axes_2d 50,50 xmin,ymin 2,2 0.007
gus axes_2d 50,50 xmax,ymax -2,-2 -0.007
gks set pmark type square
gks set pmark color_index blue
gks set pline color_index blue
gks polyline x1,y1
gks polymarker X1,Y1
gks set pmark type circle
gks set pmark color_index red
gks set pline color_index red
gks polyline x2,y2
gks polymarker X2,Y2
gks set pline color_index black
gks set text upvec 0 1
gks set xform ndc
gks set text height 0.027
gks set text color_index blue

text:
gks set text color_index black
gks set text height 0.036
gks text 0.44 0.87 Pt
gks text 0.17 0.8 picture-cut enlargement
gks set text height 0.027
gus text 0.80 0.04 T <bracket\K>
gus text 0.02 0.88 d/{dT} { {<rho>-<rho\0>}/{<DELTA><rho\irr>}} T
gks set text upvec 0 1

! 2. Window
gks set xform ndc
gks set fill color_index yellow
d:= 0.16 0.16 0.61 0.61 0.16
g:= 0.34 0.79 0.79 0.34 0.34
gks fill d g
gks set xform wc

!
gks set viewport 0.16,0.61 0.34,0.79
gks set window 0,200 -2,45
gus axes_2d 50,10. 0,-2 2,2 0.007
gus axes_2d 50,10. 200,45 -2,-2 -0.007
gks set pmark type square
gks set pmark color_index blue
gks set pline color_index blue
gks polyline x1,y1
gks polymarker X1,Y1
gks set pmark type circle
gks set pmark color_index red
gks set pline color_index red
gks polyline x2,y2
gks polymarker x2,y2
gks set pline color_index black

del var *
```

FIGURE C-4

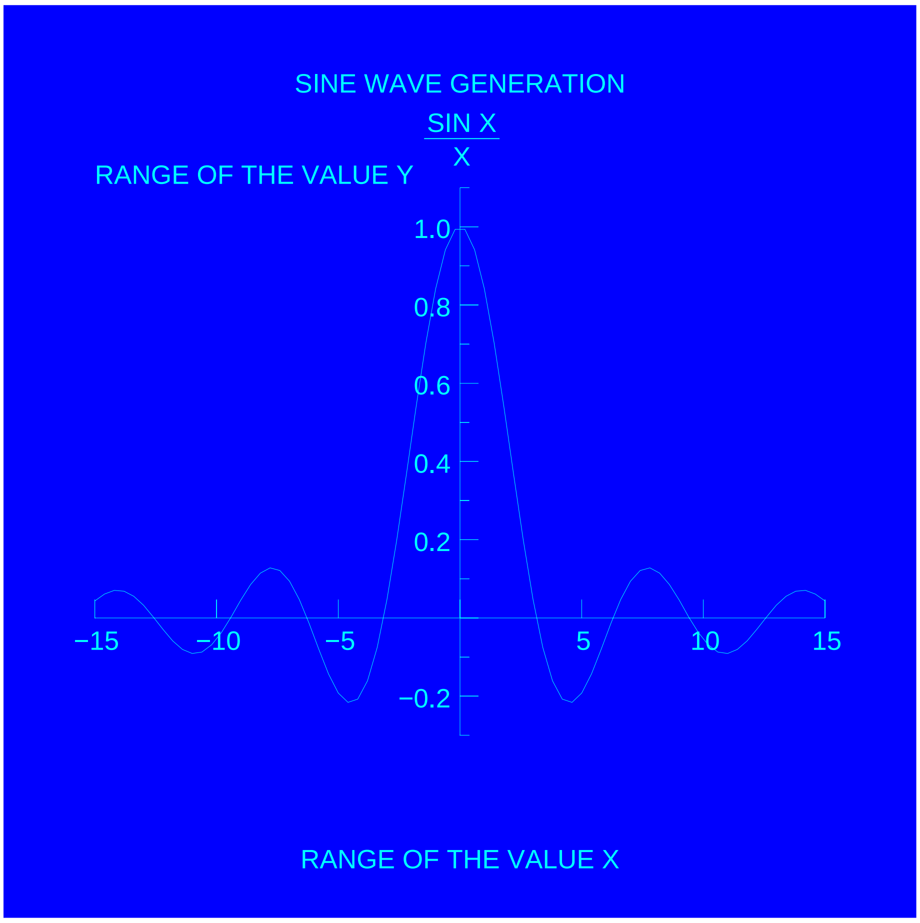
Demo4.gli



demo5.gli

```
 page
 gks set xform ndc
 gks set text align centre bottom
 gks set text font 3 string
 gks set pline color cyan
 gks set text color cyan
 gks set text height 0.021

! Background
 gks set fill int_style solid
 gks set fill color blue
 a := 0 1 1 0 0
 b := 0 0 1 1 0
 gks fill a b

!Text
 gks text 0.5 0.9 SINE WAVE GENERATION
 gus text 0.5 0.82 {SIN X}/X
 gks text 0.5 0.05 RANGE OF THE VALUE X
 gks set text align left bottom
 gks text 0.1 0.8 RANGE OF THE VALUE Y
 gks set xform wc


 gks set viewport 0.1,0.9 0.2,0.8
 gks set window -15,15 -0.3,1.1
 gus axes_2d 5,0.1 0,0 1,2 0.01
 x := -15(0.4)15
 f = sin(x)/x
 gks polyline x f

 gks set text font -1 string
 gks set pline color black
 gks set text color black

 del var *
 del fun *
```

FIGURE C-5 Demo5.gli

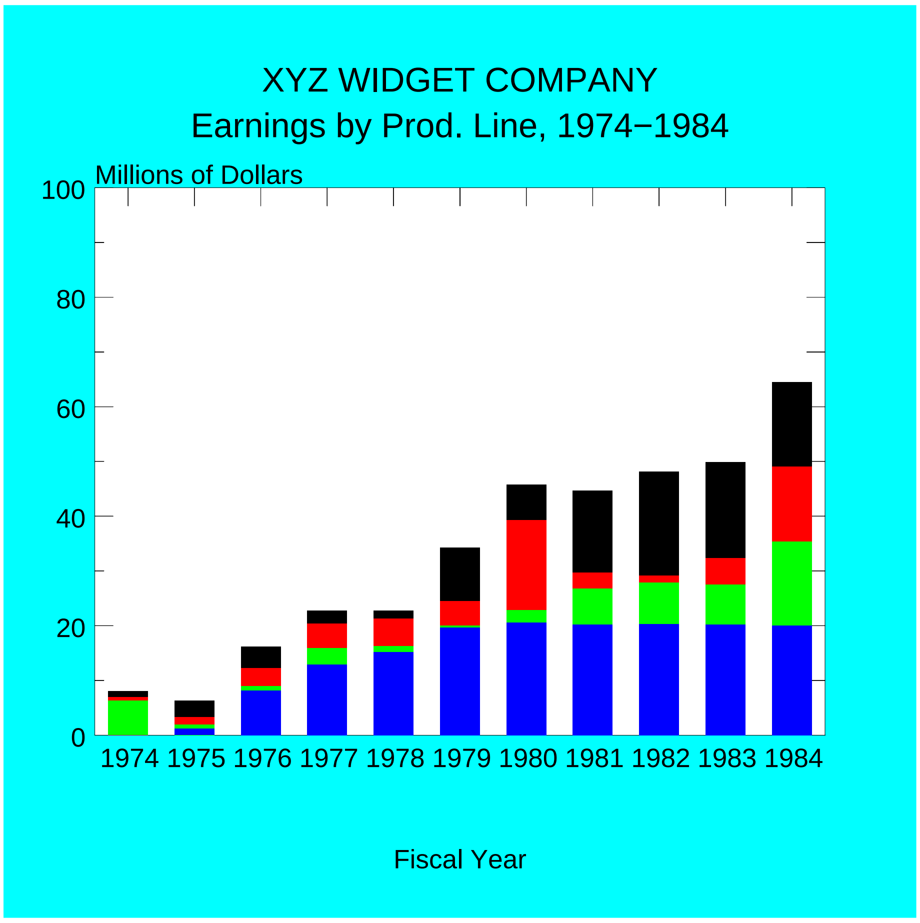


demo6.gli

```
page
gks set xform ndc
gks set text font 3 string
gks set text align centre bottom
gks set text height 0.027

gks set fill color_index cyan
gks set fill int_style solid
a:= 0 0 1 1 0
b:= 0 1 1 0 0
gks fill a b

gks text 0.5 0.9 XYZ WIDGET COMPANY
gks text 0.5 0.85 Earnings by Prod. Line, 1974-1984
gks set text height 0.021
gks text 0.5 0.05 Fiscal Year
gks set text align left bottom
gks text 0.1 0.8 Millions of Dollars

gks set fill color_index white
a:= 0.1 0.1 0.9 0.9 0.1
b:= 0.2 0.8 0.8 0.2 0.2
gks fill a b

gks set viewport 0.1,0.9 0.2,0.8
gks set window 1973.5,1984.5 0,100
gks set xform wc
gus axes_2d 1,10 1973.5,0 1,2 0.01
gus axes_2d 1,10 1984.5,100 0,0 -0.01
read ‘GLI_DEMO’demo6.dat x,y1,y2,y3,y4
gks set fill color_index black
gus bar_graph x y4
gks set fill color_index red
gus bar_graph x y3
gks set fill color_index green
gus bar_graph x y2
gks set fill color_index blue
gus bar_graph x y1
gks set fill color_index black

del var *
```

FIGURE C-6

Demo6.gli

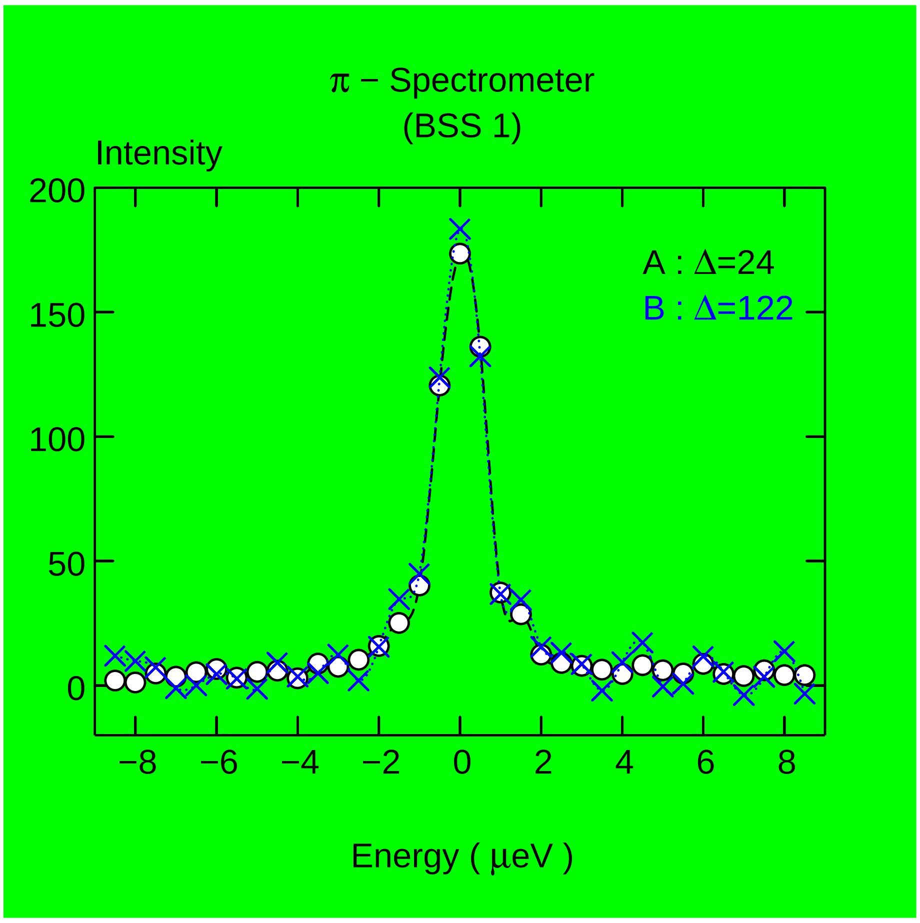


demo7.gli

```
 page
 gks set xform ndc
 gks set text font 3 string
 gks set pline color_index yellow
 gks set text color_index yellow
 gks set pline linewi 3
 gks set text align centre bottom
 gks set text height 0.027

! Background
 gks set xform ndc
 a := 0 0 1 1 0
 b := 0 1 1 0 0
 gks set fill color green
 gks set fill int_style solid
 gks fill a b
 gks set text color_index black
 gus text 0.5 0.9 <pi> - Spectrometer
 gus text 0.5 0.85 (BSS 1)
 gus text 0.5 0.05 Energy ( <mu>eV )
 gks set text align left bottom
 gus text 0.1 0.82 Intensity

 gks set text align left bottom
 gus text 0.7 0.7 A : <DELTA>=24
 gks set text color_index blue
 gus text 0.7 0.65 B : <DELTA>=122

 gks set text color_index black

 gks set xform wc
 gks set viewport 0.1,0.9 0.2,0.8
 gks set window -9,9 -20,200
 gks set pline color_index black
 gus axes_2d 2,50 -9,-20 1,1 0.01
 gus axes_2d 2,50 9,200 0,0 -0.01

 read ‘GLI_DEMO’demo7.dat energy,a,b
 gus set smoothing 1
 gks set pline linetype dashed
 gus spline energy a
 gks set pmark type circle
 gks set pmark color_index black
 gks polymarker energy a

 gks set pline color_index blue
 gks set pline linetype dotted
 gus spline energy b
 gks set pmark type diagonal_cross
 gks set pmark color_index blue
 gks polymarker energy b

 del var *
 gks set text color_index black
 gks set pline color_index black
 gks set pmark color_index black
 gks set pline linetype solid
 gks set pline linewi 1
 gus set smoothing 0
```

FIGURE C-7

Demo7.gli

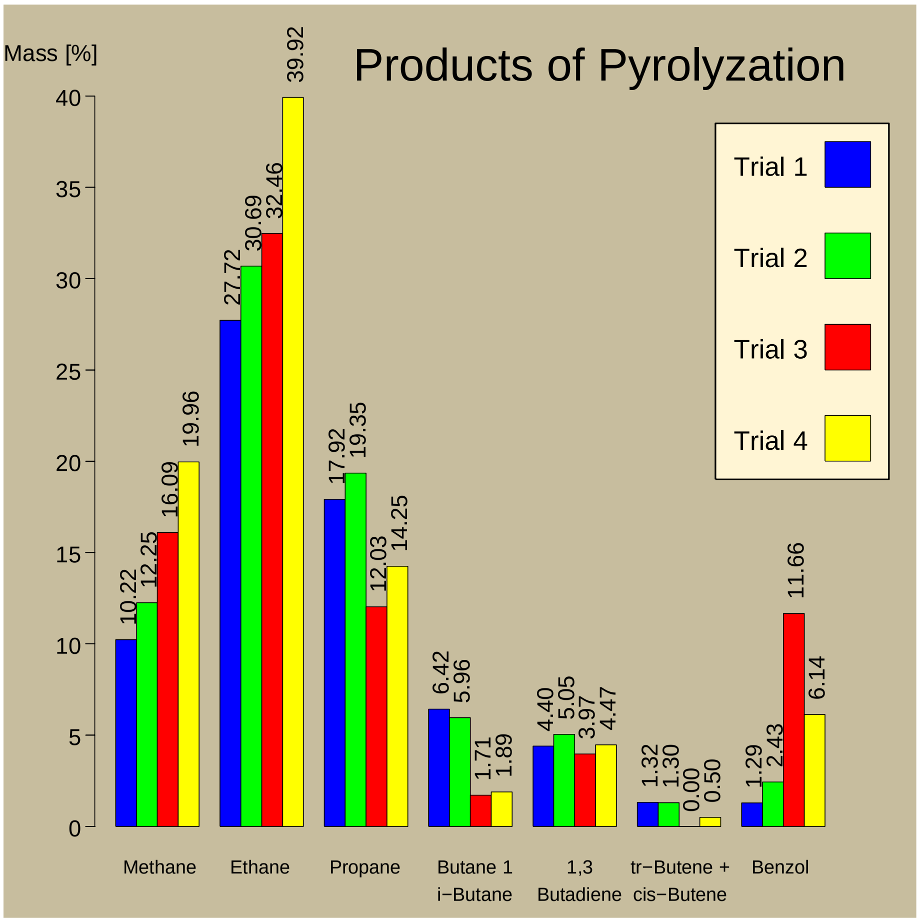


demo8.gli

```
page
gks set xform wc
gks set fill int_style solid
gks set viewport 0.1 0.9 0.1 0.9
gks set window 0 35 0 40
gks set text height 0.021
gks set text font 3 string
define symbol c1=blue
define symbol c2=green
define symbol c3=red
define symbol c4=yellow

! Background
gks set xform ndc
def var x 0,0,1,1,0
def var y 0,1,1,0,0
gks set color 5 0.78 0.74 0.62
gks set fill color cyan
gks fill x y
del var x
del var y
gks set xform wc
gus axes_2d 0 5 0 0 1 1 -0.005


! box x1 y1 x2 y2 color

! Methane
@ ‘GLI_DEMO’box 1 0 2 10.22 ‘c1’
@ ‘GLI_DEMO’box 2 0 3 12.25 ‘c2’
@ ‘GLI_DEMO’box 3 0 4 16.09 ‘c3’
@ ‘GLI_DEMO’box 4 0 5 19.96 ‘c4’
! Ethane
@ ‘GLI_DEMO’box 6 0 7 27.72 ‘c1’
@ ‘GLI_DEMO’box 7 0 8 30.69 ‘c2’
@ ‘GLI_DEMO’box 8 0 9 32.46 ‘c3’
@ ‘GLI_DEMO’box 9 0 10 39.92 ‘c4’
! Propane
@ ‘GLI_DEMO’box 11 0 12 17.92 ‘c1’
@ ‘GLI_DEMO’box 12 0 13 19.35 ‘c2’
@ ‘GLI_DEMO’box 13 0 14 12.03 ‘c3’
@ ‘GLI_DEMO’box 14 0 15 14.25 ‘c4’
! Butane1 / i-Butane
@ ‘GLI_DEMO’box 16 0 17 6.42 ‘c1’
@ ‘GLI_DEMO’box 17 0 18 5.96 ‘c2’
@ ‘GLI_DEMO’box 18 0 19 1.71 ‘c3’
@ ‘GLI_DEMO’box 19 0 20 1.89 ‘c4’
! Butadiene 1,3
@ ‘GLI_DEMO’box 21 0 22 4.40 ‘c1’
@ ‘GLI_DEMO’box 22 0 23 5.05 ‘c2’
@ ‘GLI_DEMO’box 23 0 24 3.97 ‘c3’
@ ‘GLI_DEMO’box 24 0 25 4.47 ‘c4’
! tr-Butene + cis-Butene
@ ‘GLI_DEMO’box 26 0 27 1.32 ‘c1’
@ ‘GLI_DEMO’box 27 0 28 1.30 ‘c2’
@ ‘GLI_DEMO’box 28 0 29 0.00 ‘c3’
@ ‘GLI_DEMO’box 29 0 30 0.50 ‘c4’
! Benzol
@ ‘GLI_DEMO’box 31 0 32 1.29 ‘c1’
@ ‘GLI_DEMO’box 32 0 33 2.43 ‘c2’
@ ‘GLI_DEMO’box 33 0 34 11.66 ‘c3’
@ ‘GLI_DEMO’box 34 0 35 6.14 ‘c4’

! Text
gks set xform ndc
gks set text align center normal
gks set text height 0.018
gus text 0.17 0.05 Methane
gus text 0.28 0.05 Ethane
gus text 0.395 0.05 Propane
gus text 0.515 0.05 Butane 1
gus text 0.515 0.02 i-Butane
gus text 0.63 0.05 1,3
gus text 0.63 0.02 Butadiene
gus text 0.74 0.05 tr-Butene +
gus text 0.74 0.02 cis-Butene
gus text 0.85 0.05 Benzol

! Main title and axes
gks set text height 0.036
gus text 0.65 0.92 Products of Pyrolyzation
gks set text height 0.018
gus text 0.05 0.94 Mass [%]

! Legend
gks set text align left half
gks set text height 0.021

! box
x := 0.78,0.78,0.97,0.97,0.78
y := 0.48,0.87,0.87,0.48,0.48
gks set color 7 1 0.96 0.83
gks set fill color magenta
gks fill x y
gks set pline linewidth 2
gks polyline x y
gks set pline linewidth 1

x := 0.9,0.9,0.95,0.95,0.9
```

```
y := 0.8,0.85,0.85,0.8,0.8
gks set fill color ‘c1’
gks fill x y
gks polyline x y
gus text 0.8 0.825 Trial 1

y := 0.7,0.75,0.75,0.7,0.7
gks set fill color ‘c2’
gks fill x y
gks polyline x Y
gus text 0.8 0.725 Trial 2

y := 0.6,0.65,0.65,0.6,0.6
gks set fill color ‘c3’
gks fill x y
gks polyline x y
gus text 0.8 0.625 Trial 3

y := 0.5,0.55,0.55,0.5,0.5
gks set fill color ‘c4’
gks fill x y
gks polyline x y
gus text 0.8 0.525 Trial 4

gks set pline color_index black
gks set pline linewidth 1.0
gks set fill color black
gks set text color black
gks set text fontprec -1 string
```

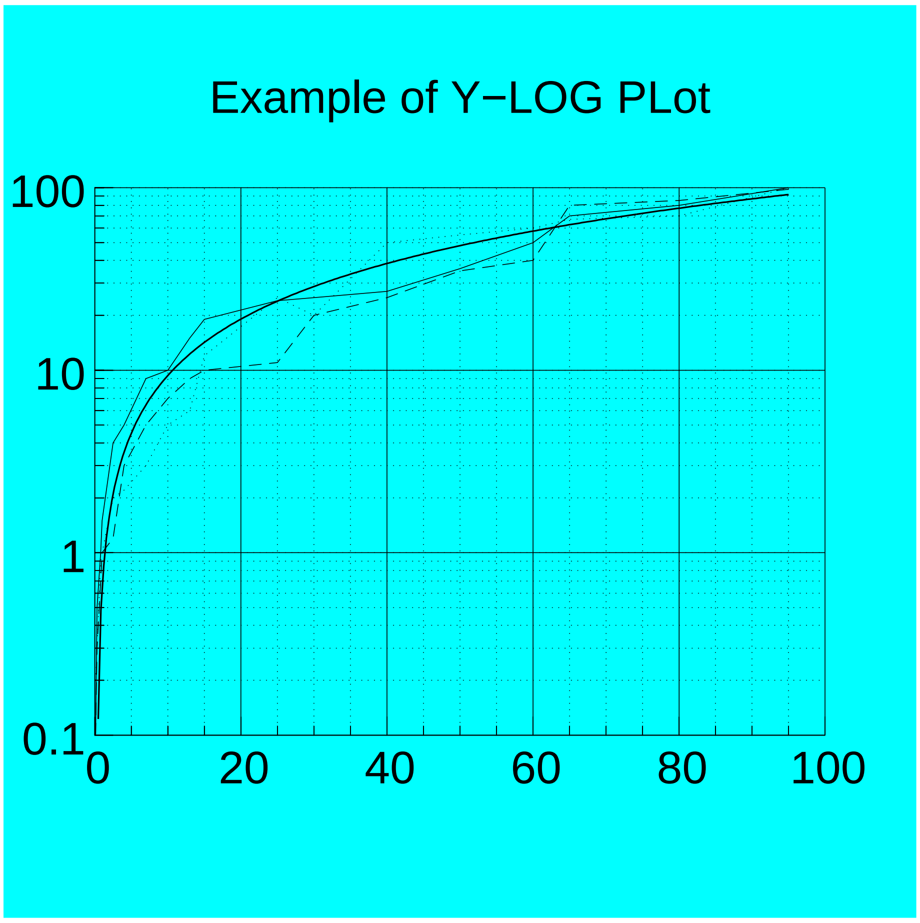
FIGURE C-8 Demo8.gli



demo9.gli

```
  PAGE
  gks set xform ndc
  gks set text font 3 string
  gks set text height 0.036
  gks set text align centre half

 ! Background
  gks set fill int_style solid
  gks set fill color_index cyan
  a:=0 0 1 1 0
  b:=0 1 1 0 0
  gks fill a b

  gks text 0.5 0.9 Example of Y-LOG PLot
  gks set xform wc
  read 'GLI_DEMO'demo9.dat x a b c
  gks set window 0 100 0.1 100
  gks set viewport 0.1,0.9 0.2,0.8
  gus set scale y_log
  gus axes_2d 5,1, 0,0.1, 4,1 0.01
  gus grid 5,1,0,0.1, 4,1
  gks set pline linetype solid
  gus polyline x a
  gks set pline linewidth 2
  gus linreg x a
  gks set pline linewidth 1
  gks set pline linetype dashed
  gus polyline x b
  gks set pline linetype dotted
  gus polyline x c
  gks set pline linetype solid
  gus set scale linear

  del var *
```

FIGURE C-9 Demo9.gli

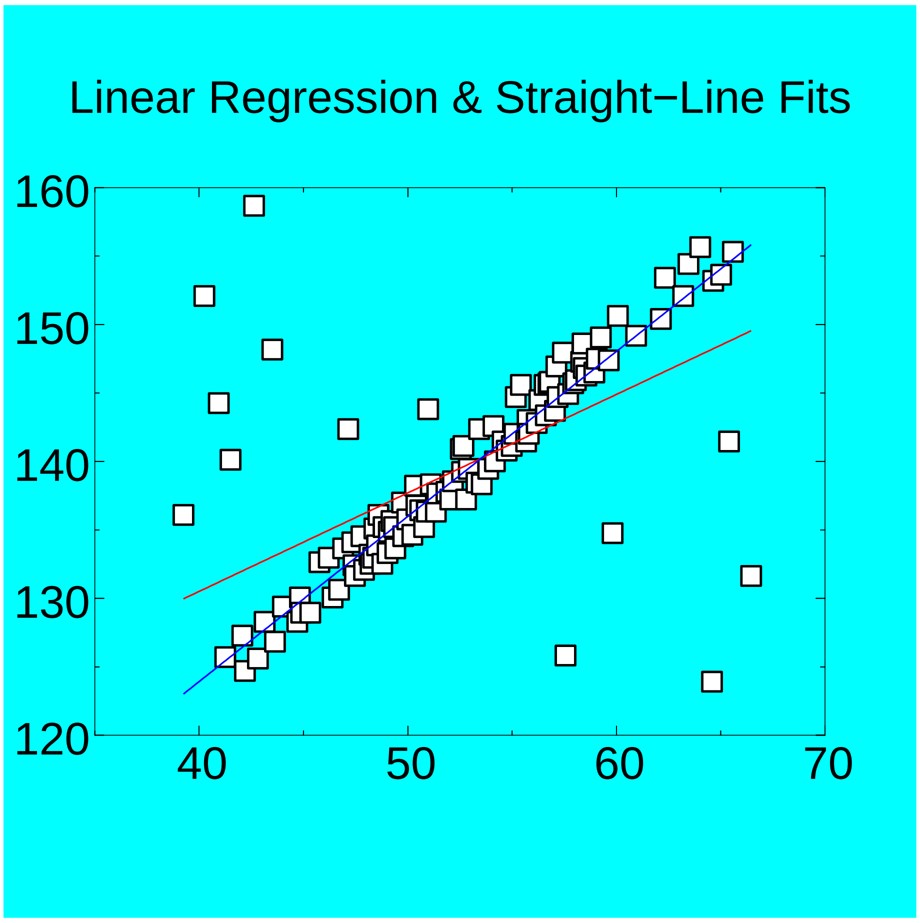


demo10

```
page
gks set xform ndc
gks set text font 3 string
gks set text align centre half
gks set text height 0.036

gks set fill color_index cyan
gks set fill int_style solid
a:=0 0 1 1 0
b:=0 1 1 0 0
gks fill a b

gks text 0.5 0.9 Linear Regression & Straight-Line Fits
gks set xform wc

gks set viewport 0.1,0.9 0.2,0.8
gks set window 35 70 120 160
gus axes_2d 5 5 35 120 2 2 0.005
gus axes_2d 5 5 70 160 -2 -2 -0.005

gks set pmark color_index black
gks set pmark type square
read ‘GLI_DEMO’demo10.dat x y
gus polymarker x y
gks set pline linewidth 2
gks set pline color_index red
gus linreg x y
gks set pline color_index blue
gus linfit x y
gks set pline linewidth 1
gks set pline color_index black

del var *
```

FIGURE C-10

Demo10.gli

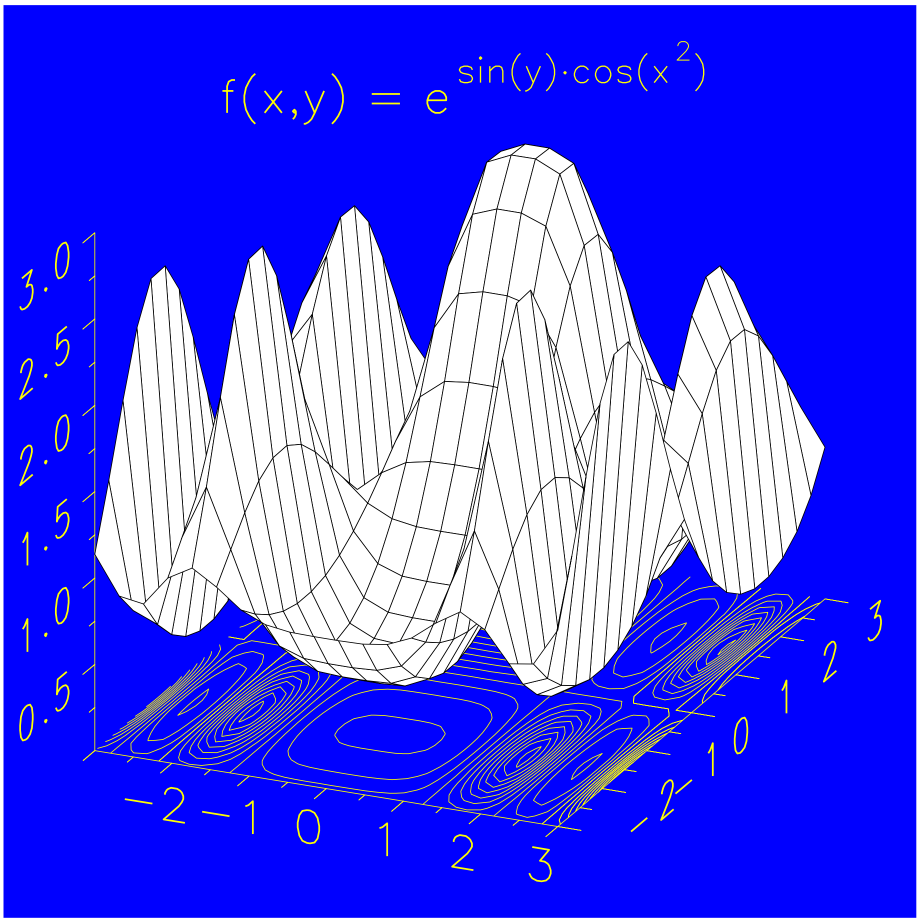


demo11.gli

```
 page
 gks set xform ndc
 gks set text font -3 stroke
 gks set text height 0.036
 gks set text align centre half
 gks set pline color yellow
 gks set pline linewi 3
 gks set text color yellow

! Background
 gks set fill color blue
 gks set fill int_style solid
 a := 0 0 1 1 0
 b := 0 1 1 0 0
 gks fill a b
 gus text 0.5 0.9 f(x,y) = e**{sin(y)<mult>cos(x**2)}
 gks set xform wc

 gks set window -3,3 -3,3
 gks set viewport 0.1,0.9 0.1,1
 gks set pline linewi 1

 x := -3(0.15)3
 y := x
 h := 0(0.2)2.8
 f = exp(sin(y)*cos(x**2))
 gus set space 0,30000 30,45
 gus contour x y h f

 gus set space 0,3 30,45
 gus axes_3d 0.5,0,0.25 -3,-3,0 2,0,2 -0.02
 gus axes_3d 0,0.5,0 3,-3,0 0,2,0 0.02
 gks set pline color black

 x := -3(0.315789)3
 y := x
 gks set fill color white
 gus surface x y f filled_mesh

 del var *
 del fun *
 gks set pline color black
 gks set text color black
 gks set pline linewi 1
 gks set text color black
 gks set text font -1 string
```

FIGURE C-11

Demo11.gli

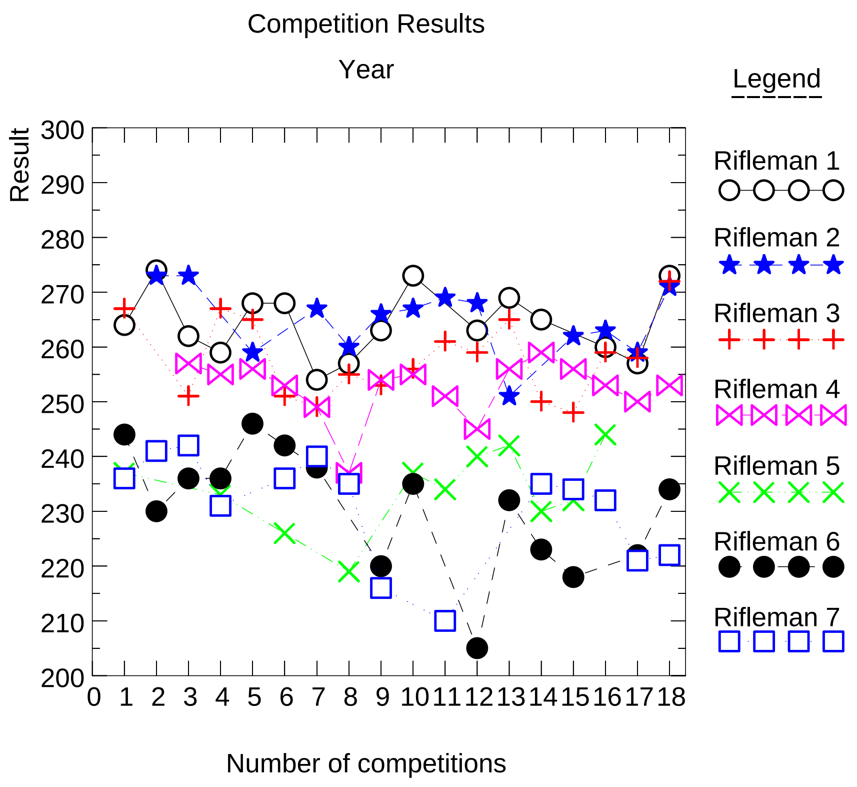


demo12.gli

```
page
gks set text align centre bottom
gks set text font 3 string
gks set text height 0.021
gks set pline color black
gks set xform ndc

gks set fill color_index white
gks set fill int_style solid
a:= 0 0 1 1 0
b:= 0 1 1 0 0
gks fill a b


gks text 0.4 0.9 Competition Results
gks text 0.4 0.85 Year
gks text 0.4 0.09 Number of competitions

gks set text align left bottom
gks set text upvec -1 0

gks set text align right base
gks text 0.03 0.8 Result
gks set xform wc

gks set text upvec 0 1
gks set viewport 0.1 0.75 0.2 0.8
gks set window 0,18.5 200,300
gus axes_2d 1,5 0,200 1,2 0.008
gus axes_2d 1,5 18.5,300 -1,-2 -0.008

gks set pmark type circle
read ‘GLI_DEMO’demo12_1.dat x y
gks set pline linetype solid
gks polyline x y
gks polymarker x y

gks set pmark type asterisk
gks set pmark color_index blue
read ‘GLI_DEMO’demo12_2.dat a b
gks set pline linetype dashed
gks set pline color_index blue
gks polyline a b
gks polymarker a b

gks set pmark type plus
gks set pmark color_index red
read ‘GLI_DEMO’demo12_3.dat c d
gks set pline linetype dotted
gks set pline color_index red
gks polyline c d
gks polymarker c d

gks set pmark type bowtie
gks set pmark color_index magenta
read ‘GLI_DEMO’demo12_4.dat k l
gks set pline linetype long_short_dash
gks set pline color_index magenta
gks polyline k l
gks polymarker k l

gks set pmark type diagonal_cross
gks set pmark color_index green
read ‘GLI_DEMO’demo12_5.dat m n
gks set pline linetype dash_2_dot
gks set pline color_index green
gks polyline m n
gks polymarker m n

gks set pmark type solid_circle
gks set pmark color_index black
read ‘GLI_DEMO’demo12_6.dat g h
gks set pline linetype spaced_dash
gks set pline color_index black
gks polyline g h
gks polymarker g h

gks set pmark type square
gks set pmark color_index blue
read ‘GLI_DEMO’demo12_7.dat i j
gks set pline linetype double_dot
gks set pline color_index blue
gks polyline i j
gks polymarker i j

gks set xform ndc
gks set text align centre bottom
gks set viewport 0.76 0.95 0.2 0.8
gks text 0.85 0.84 Legend
gks text 0.85 0.82 ------
gks text 0.85 0.75 Rifleman 1
gks text 0.85 0.666 Rifleman 2
gks text 0.85 0.583 Rifleman 3
gks text 0.85 0.50 Rifleman 4
gks text 0.85 0.416 Rifleman 5
gks text 0.85 0.333 Rifleman 6
gks text 0.85 0.25 Rifleman 7
gks set xform wc
gks set window 0,5 0,7
gks set pmark type square
gks set pmark color_index blue
x:=1,2,3,4
y:=0.44,0.44,0.44,0.44
gks set pline color_index blue
gks set pline linetype double_dot
gks polyline x y
gks polymarker x y

gks set pmark type circle
gks set pmark color_index black
y:= 6.2,6.2,6.2,6.2
gks set pline color_index black
gks set pline linetype solid
gks polyline x y
gks polymarker x y

gks set pmark type bowtie
gks set pmark color_index magenta
y:=3.33,3.33,3.33,3.33
gks set pline color_index magenta
gks set pline linetype long_short_dash
gks polyline x y
gks polymarker x y

gks set pmark type asterisk
gks set pmark color_index blue
y:=5.246,5.246,5.246,5.246
gks set pline color_index blue
gks set pline linetype dashed
gks polyline x y
gks polymarker x y

gks set pmark type plus
gks set pmark color_index red
y:=4.293,4.293,4.293,4.293
gks set pline color_index red
gks set pline linetype dotted
gks polyline x y
gks polymarker x y

gks set pmark type diagonal_cross
gks set pmark color_index green
y:=2.346,2.346,2.346,2.346
gks set pline color_index green
gks set pline linetype dash_2_dot
gks polyline x y
gks polymarker x y

gks set pmark type solid_circle
gks set pmark color_index black
```

```
y:=1.393,1.393,1.393,1.393
gks set pline color_index black
gks set pline linetype spaced_dash
gks polyline x y
gks polymarker x y

del var *
```

FIGURE C-12 Demo12.gli



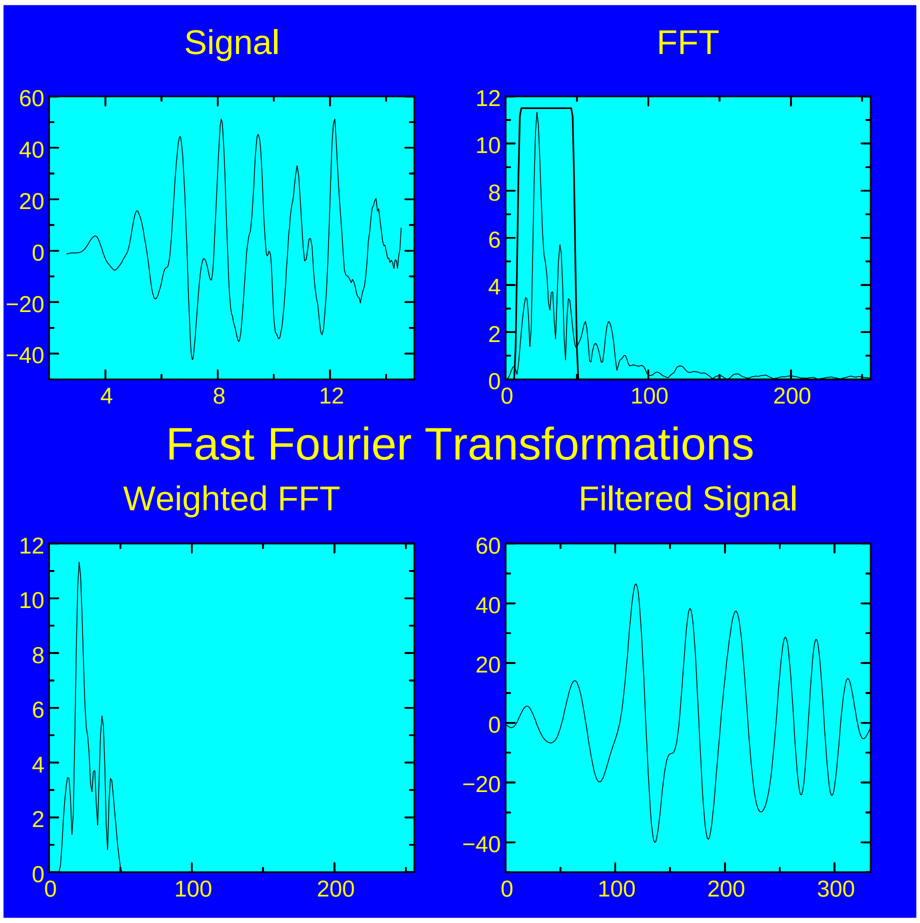
demo13.gli

```
        page
        gks set text font 3 string
        gks set pmark color black
        gks set text color yellow
        gks set pline linetype solid

! background
        gks set xform ndc
        x := 0 0 1 1 0
        y := 0 1 1 0 0
        gks set fill color_index blue
        gks set fill int_style solid
        gks fill x y
        gks set fill color_index cyan
        a := 0.05 0.05 0.45 0.45 0.05
        b := 0.05 0.41 0.41 0.05 0.05
        gks fill a b
        a := 0.55 0.55 0.95 0.95 0.55
        gks fill a b
        a := 0.05 0.05 0.45 0.45 0.05
        b := 0.59 0.9 0.9 0.59 0.59
        gks fill a b
        a := 0.55 0.55 0.95 0.95 0.55
        gks fill a b
        gks set xform wc

! signal
        gks set viewport 0.05 0.45 0.59 0.9
        gks set window 2 15 -50 60
        gks set pline linewidth 2
        gks set text height 0.018
        gus axes_2d 2 10 2 -50 2 2 0.005
        gus axes_2d 2 10 15 60 -2 -2 -0.005
        gks set pmark type circle
        gks set pline linewidth 1
        read ‘GLI_DEMO’demo13_1.dat x y
        gus polyline x y

! FFT
        gks set viewport 0.55 0.95 0.59 0.9
        gks set window 0 256 0 12
        gks set pline linewidth 2
        gks set text height 0.018
        gus axes_2d 50 1 0 0 2 2 0.005
        gus axes_2d 50 1 256 12 -2 -2 -0.005
        gks set pline linewidth 1
        gus set log
        gus fft y
        gus set nolog
```

```
       read ‘GLI_DEMO’demo13_2.dat bpx bpy
       gks set pline linewidth 2
       define function f bpy*11.5
       gus polyline bpx f


! weighted FFT
       gks set viewport 0.05 0.45 0.05 0.41
       gks set window 0 256 0 12
       gks set pline linewidth 2
       gks set text height 0.018
       gus axes_2d 50 1 0 0 2 2 0.005
       gus axes_2d 50 1 256 12 -2 -2 -0.005
       gks set pline linewidth 1
       read gus.log r i
       re = bpy * r
       im = bpy * i
       define function f sqrt(re**2 + im**2)
       gus polyline bpx f


! filtered signal
       gks set viewport 0.55 0.95 0.05 0.41
       gks set window 0 333 -50 60
       gks set pline linewidth 2
       gks set text height 0.018
       gus axes_2d 50 10 0 -50 2 2 0.005
       gus axes_2d 50 10 333 60 -2 -2 -0.005
       gks set pline linewidth 1
       gus inverse_fft re im

! text
       gks set xform ndc
       gks set text font 3 string
       gks set text align centre half
       gks set text height 0.027
       gks text 0.25 0.96 Signal
       gks text 0.75 0.96 FFT
       gks text 0.25 0.46 Weighted FFT
       gks text 0.75 0.46 Filtered Signal
       gks set text height 0.036
       gks text 0.5 0.52 Fast Fourier Transformations
       gks set pline linewidth 1
       gks set text color black

 del var *
       del fun *
       gus set nolog
       gks set text font -1 string
```

FIGURE C-13

Demo13.gli



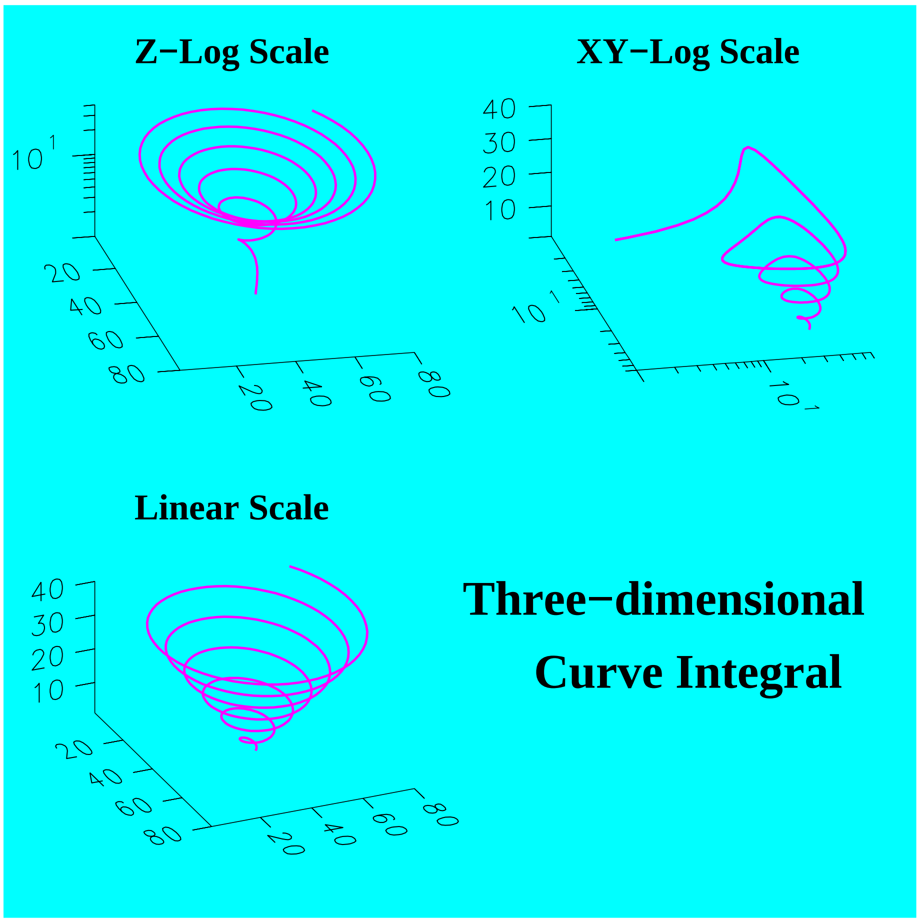
demo14.gli

```
        page
        gks set text font 3 string
        gks set text height 0.021
        gks set pline linew 3
        gks set pline linetype solid
        gks set pline color magenta
        t:=1(0.1)40
        def fun x 40+t*cos(t)
        def fun y 40+t*sin(t)
        def fun z t

! Background
        gks set xform ndc
        gks set fill color cyan
        gks set fill int_style solid
        a := 0 0 1 1 0
        b := 0 1 1 0 0
        gks fill a b
        gks set xform wc

! xyz_linear
        gks set viewport 0.1 0.45 0.1 0.41
        gks set window 1,80 1,80
        gus set scale linear
        gus set space 1 40
        gus curve x,y,z
        gks set pline color black
        gks set pline linewidth 1.0
        gus axes_3d
        gks set pline linewidth 3.0
        gks set pline color magenta


! z_log
        gks set viewport 0.1 0.45 0.6 0.91
        gus set space 1 40 70 60
        gus set scale z_log
        gus curve x,y,z
        gks set pline color black
        gks set pline linewidth 1.0
        gus axes_3d
        gks set pline linewidth 3.0
        gks set pline color magenta


! xy_log
        gks set viewport 0.6 0.95 0.6 0.91
        gks set window 1,70 1,70
        t:=1(0.1)30
```

```
               def fun x 30+t*cos(t)
               def fun y 30+t*sin(t)
               def fun z t
               gus set space 1 40 70 60
               gus set scale xy_log
               gus curve x,y,z
               gks set pline color black
               gks set pline linewidth 1.0
               gus axes_3d
               gks set pline color magenta
               gks set pline linewidth 3.0



       ! Text
               gks set xform ndc
               gks set text font 16 string
               gks set text color black
               gks set text height 0.036
               gks set text align centre half
               gks text 0.75 0.35 Three-dimensional
               gks text 0.75 0.27 Curve Integral
               gks set text height 0.027
               gks text 0.25 0.45 Linear Scale
               gks text 0.25 0.95 Z-Log Scale
               gks text 0.75 0.95 XY-Log Scale
               gks set pline linew 1
               gus set scale linear
               gks set xform wc
               gks set text font -1 string
               del var *
               del fun *
```

FIGURE C-14

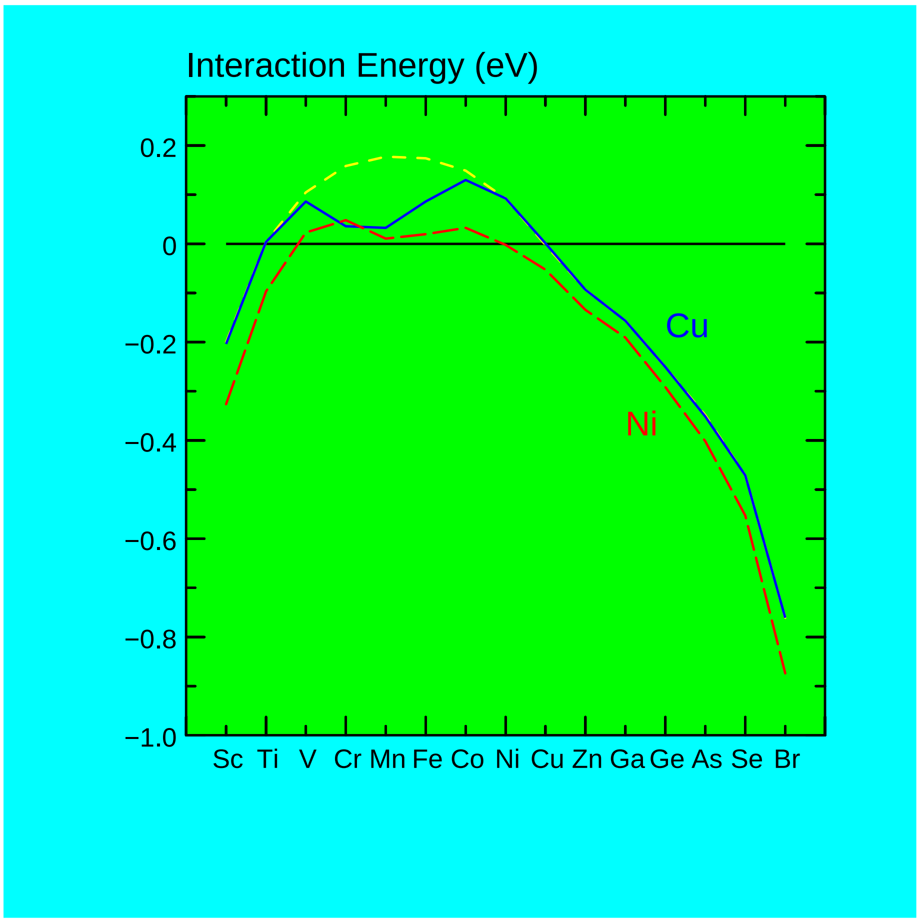
Demo14.gli



demo15.gli

```
 page
 gks set clip off
 gks set text font 3 string

 gks set pline linew 3
 gks set pline color black
 gks set text color black
 gks set text height 0.021

! Background
 gks set xform ndc
 x := 0 1 1 0 0
 y := 0 0 1 1 0
 gks set fill color cyan
 gks set fill int solid
 gks fill x y
 gks set xform wc

 gks set viewport 0.2 0.9 0.2 0.9
 gks set window 0 1 0 1
 gks set fill color green
 gks fill x y

!Axes
 gks set window 0 8 -1 0.3
 gus axes_2d 0.5 0.1 0 -1 -2 2 0.01
 gus axes_2d 0.5 0.1 8 0.3 -2 -2 -0.01

! Text
 gks set text height 0.027
 gks set text align left bottom
 gus text 0 .33 Interaction Energy (eV)
 gks set text color blue
 gus text 6 -0.2 Cu
 gks set text color red
 gus text 5.5 -0.4 Ni

 gks set text height 0.021
 gks set text align center top
 gks set text color black
 gus text .5 -1.02 Sc
 gus text 1 -1.02 Ti
 gus text 1.5 -1.02 V
 gus text 2 -1.02 Cr
 gus text 2.5 -1.02 Mn
 gus text 3 -1.02 Fe
 gus text 3.5 -1.02 Co
 gus text 4 -1.02 Ni
 gus text 4.5 -1.02 Cu
 gus text 5 -1.02 Zn
 gus text 5.5 -1.02 Ga
 gus text 6 -1.02 Ge
 gus text 6.5 -1.02 As
 gus text 7 -1.02 Se
 gus text 7.5 -1.02 Br

! Draw Polylines
 x := 0.5 7.5
 y := 0 0
 gks set pline linet solid
 gks set pline color black
 gks polyline x y

 gks set pline linew 3
 read ‘GLI_DEMO’demo15.dat x y1 y2 y3

 gks set pline color yellow
 gks set pline linet spaced_dash
 gus polyl x y3

 gks set pline linet solid
 gks set pline color blue
 gus polyl x y1
 gks set pline linet long_dash
 gks set pline color red
 gus polyl x y2

 del var *
 gks set text font -1 string
 gks set pline linet solid
 gks set pline linewidth 1
 gks set pline color black
 gks set clip on
```

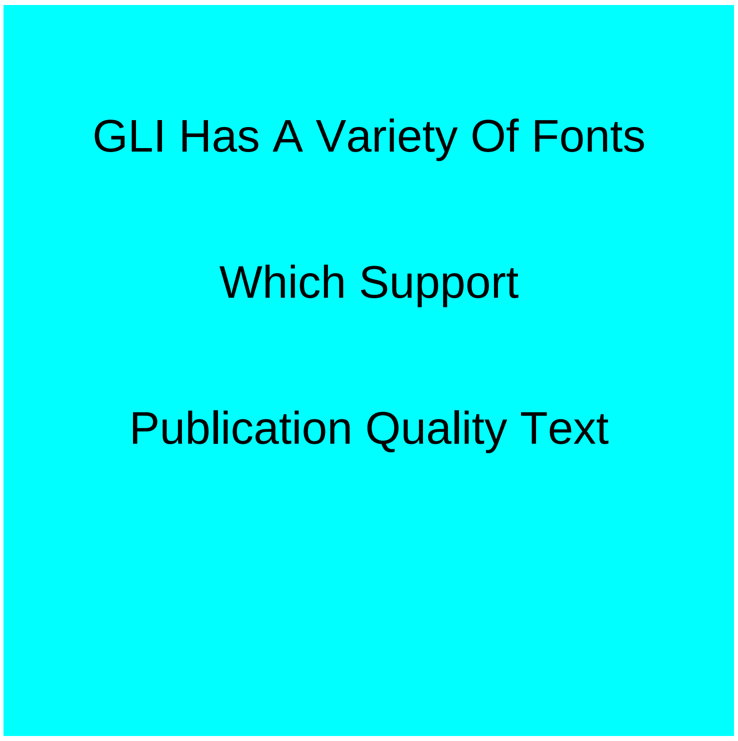
FIGURE C-15 Demo15.gli



demo16.gli

```
!This program generates a title for the GLI demos
page
gks set fill color cyan
gks set fill int_style solid
gks set xform wc
gks set viewport 0.1 0.9 0.1 0.9
gks set window 0 1 0 1
x := 0 0 1 1
y := 0 1 1 0
gks fill_area x y
gks set text height 0.036/0.8
gks set text align center normal
gks text .5 .8 GLI Has A Variety Of Fonts
gks text .5 .6 Which Support
gks text .5 .4 Publication Quality Text
del var *
gks set text align normal normal
gks set text height 0.02
gks set fill int_style hollow
gks set fill color black
```

FIGURE C-16 Demo16.gli


GLI Has A Variety Of Fonts

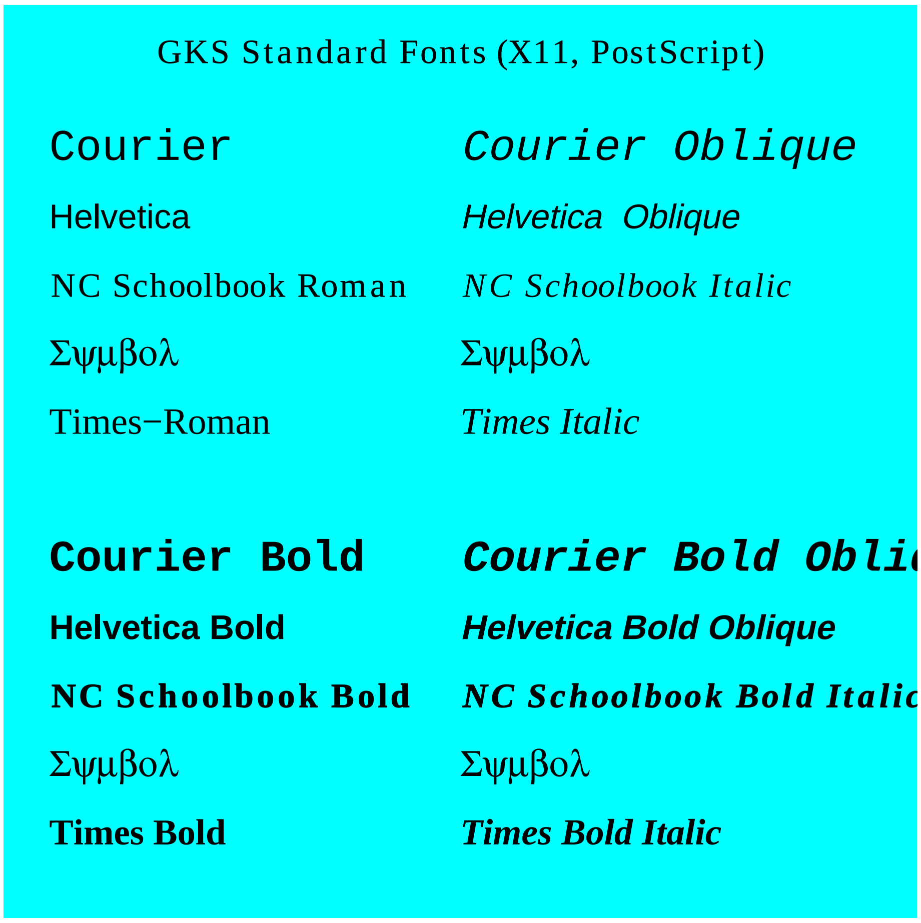
Which Support

Publication Quality Text

demo17.gli

```
page
gks set xform ndc
gks set text height 0.027

x := 0 1 1 0
y := 0 0 1 1
gks set fill int solid
gks set fill color cyan
gks fill x y

gks set text font 5 string
gks set text align center half
gks text .5 .95 GKS Standard Fonts (X11, PostScript)

gks set text align left bottom

t:=0
x:=0.05
y:=0.5
gks set text font t+2 string
gks text x y+0.325 Courier
gks set text font t+3 string
gks text x y+0.25 Helvetica
gks set text font t+5 string
gks text x y+0.175 NC Schoolbook Roman
gks set text font t+7 string
gks text x y+0.1 Symbol
gks set text font t+8 string
gks text x y+0.025 Times-Roman

t:=8
y:=0.05
gks set text font t+2 string
gks text x y+0.325 Courier Bold
gks set text font t+3 string
gks text x y+0.25 Helvetica Bold
gks set text font t+5 string
gks text x y+0.175 NC Schoolbook Bold
gks set text font t+7 string
gks text x y+0.1 Symbol
gks set text font t+8 string
gks text x y+0.025 Times Bold

t:=16
x:=0.5
y:=0.5
gks set text font t+2 string
gks text x y+0.325 Courier Oblique
gks set text font t+3 string
gks text x y+0.25 Helvetica Oblique
gks set text font t+5 string
gks text x y+0.175 NC Schoolbook Italic
gks set text font t+7 string
gks text x y+0.1 Symbol
gks set text font t+8 string
gks text x y+0.025 Times Italic

t:=24
y:=0.05
gks set text font t+2 string
gks text x y+0.325 Courier Bold Oblique
gks set text font t+3 string
gks text x y+0.25 Helvetica Bold Oblique
gks set text font t+5 string
gks text x y+0.175 NC Schoolbook Bold Italic
gks set text font t+7 string
gks text x y+0.1 Symbol
gks set text font t+8 string
gks text x y+0.025 Times Bold Italic

del var x y t
```

FIGURE C-17 Demo17.gli



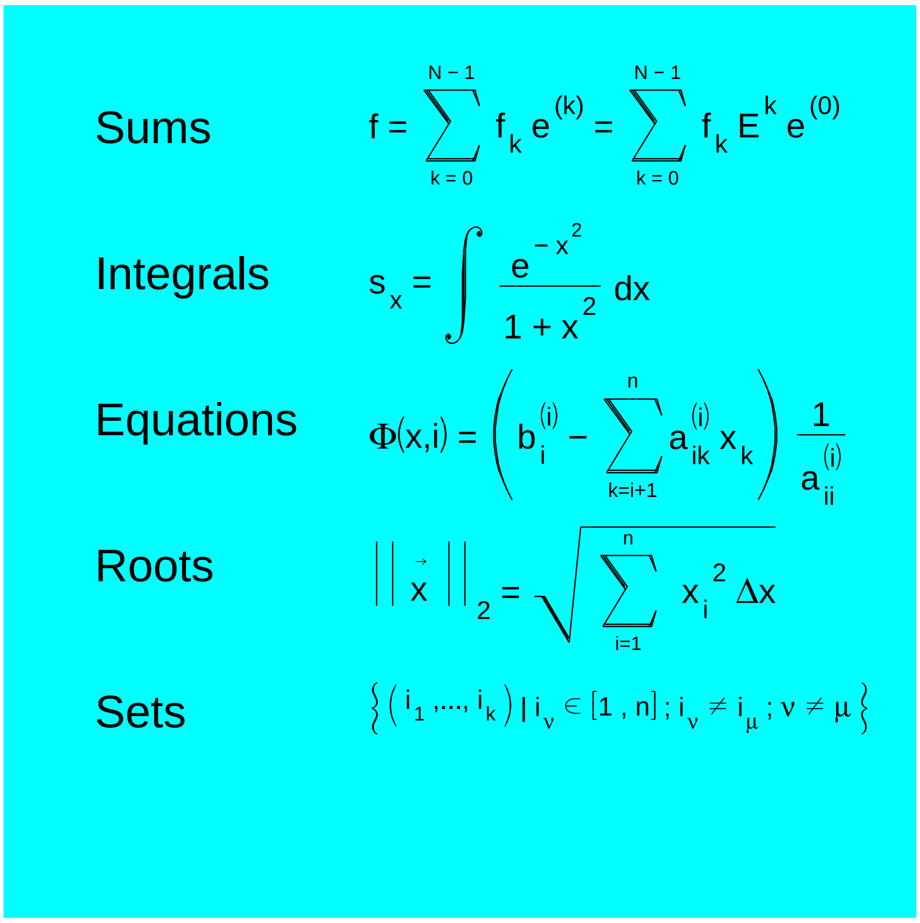
GKS Standard Fonts (X11, PostScript)

Courier	Courier Oblique
Helvetica	Helvetica Oblique
NC Schoolbook Roman	NC Schoolbook Italic
Σψμβολ	Σψμβολ
Times−Roman	Times Italic
Courier Bold	Courier Bold Obli
Helvetica Bold	Helvetica Bold Oblique
NC Schoolbook Bold	NC Schoolbook Bold Itali
Σψμβολ	Σψμβολ
Times Bold	Times Bold Italic

demo18.gli

```
        page
        gks set xform wc
        gks set text color black
        gks set text font 3 string
        gks set text align left half
        gus set text_slant 0

! Background
        gks set xform ndc
        a := 0 0 1 1 0
        b := 0 1 1 0 0
        gks set fill color cyan
        gks set fill int_style solid
        gks fill a b

! text
        gks set text height 0.036
        gus text 0.1 0.87 Sums
        gus text 0.1 0.71 Integrals
        gus text 0.1 0.55 Equations
        gus text 0.1 0.39 Roots
        gus text 0.1 0.23 Sets
        gks set text height 0.027
        gus text 0.4 0.87 f = <sum\{ <f\k> e**{(k)}}\k = 0\N - 1> =
<sum\{ <f\k> E**k e**{(0)}}\k = 0\N - 1>
        gus text 0.4 0.7 <s\x> = <integral\ e**{- x**2 }/{1 + x**2} dx>
        gus text 0.4 0.53 <PHI><par\x,i> = <par\<b\i\<par\i>> -
<sum\<a\ik\<par\i>> <x\k>\k=i+1\n>> 1/{<a\ii\<par\i>>}
        gus text 0.4 0.36 <<norm\ <x\\\\<right_arrow>> >\2> = <root\
<sum\ <x\i>**2 <DELTA>x\i=1\n>>
        gks set text height 0.021
        gus text 0.4 0.23 <brace\ <par\ <i\1> ,..., <i\k> > | <i\<nu>>
<IN> <bracket\1 , n> ; <i\<nu>> <NEQ> <i\<mu>> ; <nu> <NEQ> <mu> >
        gks set xform wc
        gks set text font 3 string

        del var *
```

FIGURE C-1 Demo18.gli



Sums $$f = \sum_{k=0}^{N-1} f_k\, e^{(k)} = \sum_{k=0}^{N-1} f_k\, E^k\, e^{(0)}$$

Integrals $$s_x = \int \frac{e^{-x^2}}{1 + x^2}\, dx$$

Equations $$\Phi(x,i) = \left(b_i^{(i)} - \sum_{k=i+1}^{n} a_{ik}^{(i)}\, x_k \right) \frac{1}{a_{ii}^{(i)}}$$

Roots $$\left\| \vec{x} \right\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2\, \Delta x}$$

Sets $$\left\{ \left(i_1, \ldots, i_k \right) \mid i_\nu \in [1, n] ; i_\nu \neq i_\mu ; \nu \neq \mu \right\}$$

APPENDIX D

Quick Reference for Commands and Functions

General Plotting

Command	Description
GUS AXES x-tick, y-tick, x-org, y-org, major-x, major-y, tick-size	Draw a pair of axis
GKS FILL_AREA x, y	Fill polygons
GKS POLYLINE x, y	Draw polygons
GKS POLYMARKER x, y	Draw marker symbols
GKS SET VIEWPORT xmin, xmax, ymin, ymax	Location of plot on screen, using normalized device coordinates
GKS SET WINDOW xmin, xmax, ymin, ymax	Scale viewport
GKS TEXT x, y, string	Output text

Three-Dimensional Plotting

Command	Description
GUS CONTOUR x, y, z, h, option, dim-x	Generate a 3D contour plot
GUS CURVE x, y, z, primitive	Draw 3D curves
GUS SURFACE x, y, z, option, dim-x	Generate a 3D surface plot

Cursor Manipulation

Command	Description
GKS REQUEST LOCATOR x, y	Read graphics cursor
GKS REQUST STROKE x, y	Get stroke input

Output Attribute

Command	Description
GKS SET FILL COLOR_INDEX index	Set the fill area color index
GKS SET FILL INT_STYLE style	Set the fill area interior style

GKS SET FILL STYLE_INDEX index	Set the fill area style index
GKS SET PLINE COLOR_INDEX index	Set the polyline color index
GKS SET PLINE LINETYPE linetype	Set the polyline linetype
GKS SET PLINE LINEWIDTH scale-factor	Set the linewidth scale factor
GKS SET PMARK COLOR_INDEX index	Set the polymarker color index
GKS SET PMARK SIZE scale-factor	Set the polymarker scale factor
GKS SET PMARK TYPE marker-type	Set the polymarker type
GKS SET TEXT ALIGN hor-align, ver-align	Set the text alignment
GKS SET TEXT COLOR_INDEX index	Set the text color index
GKS SET TEXT EXPFAC factor	Set the character expansion factor
GKS SET TEXT FONTPREC font, prec	Select text font and precision
GKS SET TEXT HEIGHT value	Set the character height
GKS SET TEXT PATH text-path	Set the text path
GKS SET TEXT UPVEC x-value, y-value	Set the character up vector

File Input/Output

Command	**Description**
APPEND file-spec var-name, ...	Append one or more data items to an output file
READ file-spec var-name, ...	Read an input file and assign the contents to one or more variables
WRITE file-spec var-name, ...	Write one or more data items to an output file

Graphic Utilities

Command	**Description**
GUS AUTOPLOT	Invoke the AUTOPLOT utility
GUS AUTOSCALE x, y	Perform automatic scaling of axes
GUS AXES_3D x-tick, y-tick, z-tick, x-org, y-org, z-org, major-x, major-y, major-z, tick-size	Draw three-dimensional axes
GUS FFT y	Draw a power spectrum
GUS GRID x-tick, y-tick, x-org, y-org, major-x, major-y	Draw a grid

GUS HISTOGRAM y	Draw a histogram
GUS INVERSE_FFT r-data, i-data	Draw the inverse FFT
GUS LINFIT x, y	Generate a robust straight-line fit
GUS LINREG x, y	Perform linear regression
GUS POLYLINE x, y, ...	Draw polygons
GUS POLYMARKER x, y, ...	Draw marker symbols
GUS SET SCALE scale, flip	Define type of transformation
GUS SET SPACE z-min, z-max, rotation, tilt	Scale Z-axis and define 3D perspective
GUS SET SMOOTHING smoothing	Set smoothing level
GUS SET WS_VIEWPORT window, size	Define orientation and size of the workstation viewport
GUS SPLINE x, y, ...	Generate a cubic spline-fit
GUS TEXT x, y, string	Output scientific text

Low Level

Command	**Description**
GKS CLEAR_WS	Clear the display surface space
GKS CLOSE_GKS	Close GKS
GKS CLOSE_WS	Close a workstation
GKS EMERGENCY_CLOSE	Emergency close GKS
GKS INQUIRE WS_CONNTYPE	Inquire all open workstations
GKS INQUIRE WS_TYPE	Inquire a list of available workstation types
GKS OPEN_GKS	Open GKS
GKS OPEN_WS	Open a workstation
GKS SET CLIPPING	Enable or disable clipping
GKS SET WS_VIEWPORT xmin, xmax, ymin, ymax	Set metric size of plot
GKS SET XFORM	Select a transformation

Imaging

Command	Description
IMAGE APPLY_FILTER mode, source, dest	Apply a filter to a fourier transformed image
IMAGE CAPTURE [file]	Capture an image to a file
IMAGE CONTRAST source, dest, lev	Modifies light and dark areas
IMAGE COPY source, dest	Creates a duplicate of a source image
IMAGE CUT source, dest, x-min, x-max, y-min, y-max	Copies a portion of a source image
IMAGE DELETE image_name	Deletes an image from memory
IMAGE DISPLAY image_name, trans	Displays an image on all active devices
IMAGE EDGE source, destination	Enhance ‘edges’ of an image
IMAGE ENHANCE source, dest, level	Improves definition between light and dark areas
IMAGE EXPORT source, variable	Converts an image from the image format to a GLI variable
IMAGE FFT source, dest	Apply a Fast Fourier Tarnsformation to an image
IMAGE FLIP source, dest, direction	Rotates an image around its vertical or horizontal axis
IMAGE GAMMA source, dest, value	Changes the intensity of an image
IMAGE GRADIENT source, dest	Calculates the gradient of an image
IMAGE HISTOGRAM source, dest	Spreads the intensity distribution of an image
IMAGE IMPORT image, variable, x-dimension	Generates a PGM format image from GLI data
IMAGE INVERSE_FFT source, dest	Do the inverse Fast Fourier Transformation
IMAGE INVERT source, destination	Reverses the pixel values of an image
IMAGE MEDIAN source, destination	Performs a median cut filtration on an image
IMAGE NORMALIZE source, dest	Enhances an image by spreading the intensity distribution across the full range of values available
IMAGE PBM source, dest [,level]	Converts an image into Portable BitMap format
IMAGE PCM source, dest [,ncolors]	Converts an image into Portable ColorMap format
IMAGE PGM source, dest [,maxgray]	Converts an image into Portable GrayMap format
IMAGE PPM source, destination	Converts any GLI image format to a Portable PixMap format
IMAGE PRINT	Prints an image
IMAGE READ file, name	Reads an image from a file
IMAGE RENAME old_name, new_name	Renames an image

IMAGE RGB_GAMMA source, dest, rvalue, gvalue, bvalue	Performes a gamma correction on the RGB values to correct images with poor color
IMAGE ROTATE source, dest, angle [,anti-alias]	Rotates an image by a specified angle
IMAGE SCALE source, dest, w, h	Resizes an image to the specified width and height
IMAGE SET COLOR image, colormap	Changes the color table of a Portable ColorMap image
IMAGE SHEAR direction, source, dest, angle [,anti-alias]	Shears (or deforms) an image by a specified angle
IMAGE SHOW	Displays information about all images currently in memory in GLI
IMAGE WRITE file, name	Writes an image to a file

APPENDIX E

Math Functions

Overview

This appendix contains descriptions of some of GLI's mathematical routines, and is provide for your reference.

The original program authors are: J. Heinen, R. Schmitz, and M. Steinert (C Version). References of research sources are provided.

FFT (d, n, sign)

Replaces DATA by its discrete Fourier transformation, if SIGN is input as 1; or replaces DATA by N times its inverse discrete Fourier transformation, if ISIGN is input as -1. DATA is a complex array of length N or, equivalently, a real array of length 2*N. N must be an integer power of 2 (this is not checked for!).

Reference
William H. Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling, Numerical Recipes, (Cambridge: Cambridge University Press, 1986) pp. 390-395.

REALFT (d, n, sign)

Calculates the Fourier transformation of a set of 2*N real valued data points. Replaces the data (which is stored in array DATA) by the positive frequency half of its complex Fourier transformation. The real valued first and last components of the complex transformation are returned as elements DATA(1) and DATA(2) respectively. N must be a power of 2. This routine also calculates the inverse transformation of a complex data array if it is the transformation of real data.

Reference
William H. Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling, Numerical Recipes, (Cambridge: Cambridge University Press, 1986) pp. 398-400.

LINREG (n, x, y, m, b, logging)

Compute the linear regression coefficients and perform an analysis of variance.

LINFIT (n, x, y, m, b, dev, logging)

Fits y = mx + b by the criterion of least absolute deviations. The arrays X and Y, of length N, are the input experimental points. The fitted parameters M and B are output, along with DEV which is the mean deviation (in y) of the experimental points from the fitted line.

SMOOTH (n, d, g, level)

Smooth a given sequence of data points. The smoother procedure is a non-linear one. As opposed to linear procedure its performance shouldn't be impaired if the data is not well behaved. Y is assumed to be a sequence of observed values of a function at equally spaced intervals. At the completion of the call Y is left un-modified and SMY gives the smoothed sequence.

SPLINE (n, x, y, m, t, s)

Compute a curve s(x) that passes through the N data points (X,Y) supplied by the user and computes the functional value S(T) at any point T on the curve. The procedure to determine s(x) involves the iterative solution of a set of simultaneous linear equations by Young's method of successive over-relaxation.

Reference

Ralston and Wilf, Mathematical Methods for Digital Computers, Vol. II (New York: John Wiley and Sons, 1967) pp. 156-158.

Greville, T.N.E., Editor, "Proceedings of An Advanced Seminar Conducted by the Mathematics Research Center", U.S. Army, University of Wisconsin, Madison. October 7-9, 1968. Theory and Applications of Spline Functions (New York, London: Academic Press, 1969), pp. 156-167.

NORMAL (n, x, y, m, d, f, logging)

Compute a normal curve overlay.

APPENDIX F

References

BECHLARS, BUHTZ [1986]: Joerg Bechlars, Rainer Buhtz, "*GKS in der Praxis*", Springer-Verlag Berlin Heidelberg New York Tokyo

DIGITAL [1984]: "*Introduction to VAX/VMS*", VAX/VMS Version 4.0, Digital Equipment Corporation Maynard, Massachusetts

DIGITAL [1986]: "*VAX/VMS DCL Concepts Manual*", VAX/VMS Version 4.4, Volume 1C, Digital Equipment Corporation Maynard, Massachusetts

DIN [1982]: DEUTSCHES INSTITUT FUER NORMUNG E.V., "*Graphisches Kernsystem (GKS)*", Informationsverarbeitung, DIN 66252, BEUTH Verlag GmbH

ENCARNACAO, STRASSER [1981]: J.Encarnacao, W.Strasser, "*Geräteunabhängige Graphische Systeme*", Drittes Darmstädter Kolloquium, R.OLDENBOURG Verlag München Wien

HARRINGTON [1983]: Steven Harrington, "*Computer Graphics*", A Programming Approach, McGRAW-HILL International Book Company

NEWMAN, SPROULL [1983]: William M.Newman, Robert F.Sproull, "*Principles of Interactive Computer Graphics*", McGRAW-HILL International Book Company

PRESS, FLANNERY, TEUKOLSKY, VETTERLING [1986]: William H. Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling, "*Numerical Recipes*", Cambridge University Press

RALSTON, WILF [1967]: Ralston and Wilf, "*Mathematical Methods for Digital Computers*", Vol. II, John Wiley and Sons, New York

APPENDIX G

Printing Graphics

Printing Graphics from GLI

There are four primary methods of printing graphics created using GLI:

- Create and print figure files.
- Change the standard output device and/or workstation type and print graphics in batch mode (detached process).
- Use WISS (Workstation Independent Segment Storage) to print to all open workstations from within GLI.
- Print from the SIGHT interface

GLI Workstations

GLI workstations, also referred to as logical workstations, represent the class of supported graphical input/output devices. Each class of device and its location is specified by the environment variables in the GLISETUP file. Table G-1 lists the generic workstations supported by GLI.

TABLE G-1 Available Workstations

Workstation Class	Environment Variable or VMS Logical Name	Description
FIGURE_FILE	GLI_FIG	Figure File (VAX Document, TeX, DECwrite, FrameMaker)
LASER_PRINTER	GLI_LW	Graphics Printer
PLOTTER	GLI_PL	Plotter
WISS		Workstation Independent Segment Storage
CGM	GLI_CGM	Computer Graphics Metafile
TERMINAL	GLI_CONID	Command Terminal, X Window
WK1… WK12		Multiple workstations of the same type (the second workstation is called WK1). Useful for X displays.

Note: GLI documentation refers to generic input/output logical devices as *workstations.*

Device Support

Table G-2 describes the physical devices supported by GLI's GKS.

TABLE G-2 Supported Workstation Types

Workstation Types	**Devices**
Figure_File	
61, 62	PostScript, Color PostScript
Laser Printers	
38	DIGITAL LN03 PLUS
61, 62	PostScript, Color PostScript Printers
Plotters	
51, 53	HP-GL Graphics Plotter (HP74xx, HP75xx)
Terminals	
72	Tektronix 401x Series Terminal
82	Tektronix 42xx Series Terminal
16, 17	DIGITAL VT330, VT340 Video Terminal
201	TAB 132/15-G Terminal
204	MONTEREY MG200 Display Terminal
207	IBM Personal Computer
Workstations (Terminal, WK1..WK12)	
41	VAX UIS
210	X Displays
211	X Display with PCM dump
213	X Display with frame buffer
214	X Display with Sun rle rasterfile dump
WISS	
5	Workstation Independent Segment Storage (WISS)
CGM	
7, 0x30007	Computer Graphics Metafile (Binary CGM, default for GLI)
8, 0x40007	Computer Graphics Metafile (Clear Text CGM)

Print to a Figure File

Graphs that do not require editing can be printed using a Figure File. A Figure is created containing device-specific instructions which are then sent to the supported device.

Open a Figure File

To generate a figure file containing PostScript instructions, use the GKS OPEN_WS command and specify the Workstation Class (Figure_File, Table G-1) and the Workstation Type (61, Table G-2) to be opened.

```
gli> gks open_ws figure_file 61
```

Create Your Graphic

Enter the GLI commands to create your graphic, or execute the command procedure that draws the graphic. You may find it helpful to create graphics using a command procedure. Then execute the command procedure with the @ instruction after issuing the GKS OPEN_WS command.

```
gli> @filename
```

Close the Figure File

Close the figure file immediately after the graphical instructions are executed.

```
gli> gks close_ws figure_file
```

Print the File

If a PostScript printer is connected to your default printing queue, use the GLI $ command to access your operating system and print the file.

On Unix:

```
gli> $lpr gli.eps
```

On VMS:

```
gli> $print gli.eps
```

If you elect to store the PostScript files for later printing or inclusion in a desktop publishing package, use the GLI $ command to access your operating system and rename the graph file. The graphic is stored in the default file *gli.eps*.

On Unix:

```
gli> $mv gli.eps newfile.eps
```

On VMS:

```
gli> $rename gli.eps newfile.eps
```

Use Figure Files with Other Interfaces

Graphs created using AUTOPLOT or SIMPLEPLOT can be printed from figure files as described above.

On Unix:

```
gli> $lpr gli.eps
```

On VMS:

```
gli> $print gli.eps
```

Redirect Standard Output

By changing the device and/or workstation type when invoking GLI, the graphical output will be sent to the named device instead of the default (your display).

Redirect Output

When starting GLI, specify the connection identification and the workstation type, using the format:

```
gli [-c conid] [-t wstype].
```

For example, to redirect your output to a PostScript file:

```
% gli -c figure -t 61
```

Create a Graph

Create a graph using a command procedure or a series of GLI instructions.

Quit GLI and Print

When you quit GLI, you can print the graphics from your operating system.

```
gli> quit
```

On Unix:

```
% lpr gli.eps
```

On VMS:

```
$ print gli.eps
```

Print Using WISS

Workstation Independent Segment Storage (WISS) retains device -independent output in memory. WISS output can be directed to multiple devices simultaneously, or it can be re-sized to exact specifications and then printed.

Open WISS

The WISS workstation opens a GKS segment that will contain the output in a device-independent format.

```
gli> gks open_ws wiss 5
gli> !create your graph(s) here
.
.
.
```

Additional graphs can be created and sent to the open segment.

Open a Figure File

First open the figure file, start the interface and draw the graphic.

```
gli> gks open_ws figure_file 61
gli> autoplot
.
.   !draw the graph
.
```

Close the Figure FIle

After the graphic is drawn, exit the interface, close the figure file, and print.

```
gli> gks close_ws figure_file
```

Close WISS

This command clears the display and closes the GKS segment while storing the graph(s):

```
gli> gks clear_ws
```

Modify the Output

After the WISS is closed, the graph can be resized and repositioned and multiple output devices can be opened.

```
gli> gks set viewport 0.85 0.15 0.85 0.15
gli> gks open_ws laser_printer 62
gli> gks open_ws plotter 51
```

Direct the Output

Using a single command, the output can be sent to all open devices simultaneously.

```
gli> gks copy_sg
```

Print SIGHT Graphics

Two options are offered to print graphics created using the SIGHT interface.

Using the Print Push Button

To print a graphic while using the SIGHT point-and-click Motif interface, click on the PRINT DRAWING button in the FILE pull-down menu. This captures the drawing into a GLI figure file (*gli.eps*) and automatically directs it to the printer using the a system print command which, by default, is defined by the GLI_LPR variable.

Printing SIGHT Graphics from GLI

Alternatively, you may elect to print graphics created with SIGHT from the GLI command language using:

```
gli> sight print
```

Note: GLI_LPR is the default print command and is defined in the GLISETUP file. You may find it beneficial to modify this file or declare your own print destination.

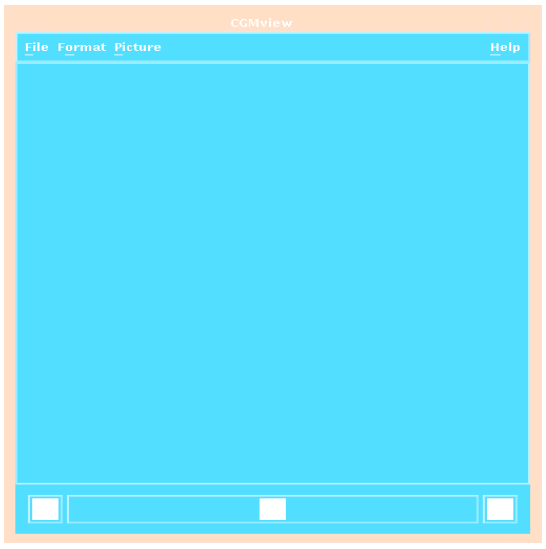
APPENDIX H

FileViewer

FileViewer is a stand-alone program that displays Computer Graphic Metafiles (CGM) files generated on any computer system. FileViewer can print CGM files to most popular graphics devices, including X Windows, PostScript (color and grayscale), HP plotters, and Digital graphics terminals. FileViewer supports Clear Text and Binary CGM encodings that conform to ANSI standard X3.122-1986.

FileViewer is compatible with workstations running the Motif (v. 1.1 or greater) implementation of X Windows and ANSI graphics terminals. Users without Motif software should use the command line version of the FileViewer.

FIGURE H-1 The CGM FileViewer



To start this program type ‘cgmview’ at the system prompt.

```
% cgmview
```

On computer systems with Motif software the interface shown above will appear on the screen. File’s can be viewed and manipulated with the commands listed in Table H-1.

TABLE H-1 FileViewer Commands

Menu Selection	Action
File	
Open	Opens a CGM file for viewing.
Quit	Quits the CGM FileViewer program.
Format	
Binary	Allows a binary encoded CGM to be read by the CGM FileViewer.
Clear Text	Allows a clear text encoded CGM to be read by the CGM FileViewer.
Picture	
Previous	View the CGM that had been selected previously.
Redraw	Redraws the CGM currently in the CGM FileViewer.
Next	Select the next CGM file for viewing.
Help	Currently not implemented.
Status Bar	
Left Arrow	Move the CGM to the left in the CGM FileViewer.
Right Arrow	Move the CGM to the right in the CGM FileViewer.
Equals	Center the CGM in the CGM FileViewer.

On systems without OSF/Motif, a command line version of the CGM FileViewer is implemented. To start the command line CGM FileViewer, enter:

```
cgmview [-c] [-h] [-m] [-p picture] [-t wstype] file
```

To look at CGM or Textronix 4010 files from within GLI, the GLI VIEW command can be used.

```
gli> view example.cgm cgm_binary gks
```

Note: Refer to the VIEW command in chapter 3 for more information.

TABLE H-2 Options for the CGM FileViewer

Option	Description
-c	Specifies the input file to be clear text format CGM file, default is binary format.
-h	Prints this information.
-m	Tells cgmview to maximize the output window.
-p picture	Picture number. CGM files can contain more than one picture, this option allows you to select which picture to view. The default is 1, and 0 selects all pictures in a file.
-t wstype	Workstation type. GLI_WSTYPE environment variable.

Examples:

```
% cgmview -t 61 europe.cgm > europe.eps
% cgmview -p 4 movie.cgm
% cgmview -c c_europe.cgmt
```

The first example tells the CGM FileViewer that output is to be displayed on workstation type 61, a PostScript printer. The next example sets the picture number to 4 to display the fourth picture in the file MOVIE.CGM. The last example tells the CGM FileViewer that the file C_EUROPE.CGM is in the clear text CGM format, not the binary format.

APPENDIX I

Remote Procedure Call

GLI supports the Remote Procedure Call (RPC) protocol from Sun Microsystems, Inc. and other computer companies. RPC allows user-defined functions to be added to GLI to provide additional functionality. For example, complex numerical calculations that are not part of the standard distribution can be added to GLI. Or, “read” and “write” procedures can be integrated with GLI to deal with binary IO.

Using RPC’s, GLI can call another program and execute that program as a GLI function. These programs can either be local to the users’ system or, in the case of a networked environment, on a different computer system altogether.

This section is intended to provide an overview of RPC. Detailed information can be obtained from a variety of sources, including most documentation for systems that support RPC.

RPC Overview

Remote Procedure Calls are high-level communications models that allow network applications to be developed without regard for the existence and function of the underlying network. RPC programs are designed to run within the client/server network model. The client (GLI) makes a procedure call which sends requests to the server. When these requests arrive, the server calls a dispatch routine. The dispatch routine performs the requested service, sends a reply back to the client (GLI), and then returns control to the client (GLI).

Requirements for a RPC

The GLI RPC interface is available for all computers running a variant of Unix that support Sun Microystems’ Remote Procedure Calls. A partial list of the computers supporting RPC capability is shown in the table below.

TABLE I-1 Systems Supporting RPC

Manufacturer	System	Operating System
Digital Equipment Corp.	DECstation, DECsystem	Ultrix
Hewlett-Packard Corp.	9000-7xx	HP-UX
Silicon Graphics	All	IRIX
Sun Microsystems, Inc.	All	SunOS
IBM	RS/6000	AIX

In a networked environment, the RPC requires a Port Mapper be running on both the sending and receiving machines. The RPC interface is currently not supported in the VAX/VMS environment.

Using RPC in a Network Environment

Since GLI and a RPC are separate programs, they do not necessarily have to run on the same computer in a networked environment. In networked environments, the user specifies the ‘service’ (host computer) that is running the application GLI will be using. A service consists of a hostname and a service number in the form of ‘[host] [: [number]]’. An example of using a service with GLI would first require the user to issue the following Unix commands on the command line on both client and server systems:

```
my_machine% glirpcd :1
your_machine% glirpcd :1
```

Note: You may omit the hostname if the RPC is running locally on your system.

This establishes a socket for communication between the two machines. Next, inside GLI, you must define a logical name for the RPC.

```
gli> define logical GLI_SERVICE “sender:1”
```

The use of the logical name GLI_SERVICE is mandatory, as is enclosing the name of the service within quotation marks (“ “). The logical GLI_SERVICE is only used in GLI for RPC’s and serves no other purpose at this time. Each RPC must also be assigned its own unique integer number which identifies it to GLI. RPC’s which are networked may share the same number, as long as the host names are different. To run a new RPC the logical GLI_SERVICE must be re-assigned to the new service before the new RPC can be used.

```
gli> define logical GLI_SERVICE “receiver:1”
```

After the logical has been defined, you can then begin to use the RPC by entering the RPC command at the GLI command line. The format for a call to RPC is:

```
gli> rpc “message” variable1, variable2, ...,
```

The ‘message’ is the command that controls the RPC and any parameters that the RPC might take. The message is not limited to one word, but it must be enclosed in quotation marks (“ “). The ‘message’ passed into the RPC may also be a GLI symbol.

The next parameter consists of one or more variables. A GLI remote procedure call may use up to eight variables. The variables passed to the RPC from GLI must be defined before they are used. You must determine the amount of memory allocated to each variable, and the memory allocated must be greater than or equal to the array size required by the RPC for that variable. RPC’s cannot increase memory for a GLI variable, but excess memory can be deleted.

The variable can be thought of as providing data and work space for the RPC. For example, if you are going to be reading in data from a file and assigning it to a variable, you must first define the variable and allocate the necessary memory before calling the RPC. If you have allocated more memory than required, the data array will be shortened to fit the amount of data. It is possible for returned arrays to be of different lengths depending on what happens inside the RPC itself.

RPC Message Handling

A user-written RPC must provide its own error handling routines. However, error messages can be passed back to GLI by using the message parameter in the RPC call. The message parameter passes the command controlling the RPC to the RPC, and it returns messages to GLI from the RPC.

A call to the C function *sprintf* with a question mark (?) as the first character of a text string is interpreted by GLI as an error message. An exclamation point (!) as the first character of a string indicates that it is being passed to GLI, and it is to be assigned to the symbol RPC_MESSAGE. This allows informational messages to be returned to GLI for use.

RPC Example

The GLI RPC command provides a convenient method to integrate user functions with GLI. The following example is also provided in your software distribution in the GLI directory. This example RPC, written in C, reads binary data from a file and assigns the data to a GLI variable, or it can write data from a GLI variable into a binary output file.

```
#include <stdio.h>
#include <string.h>

#define PATHLEN 100

int my_proc (message, argc, sizes, data)
 char *message;
 int argc;
 int *sizes;
 float **data;
{
 char path[PATHLEN], *str;
 FILE *file;
 int i, n;

 if (!strncmp(message, “read”, 4))
 {
 /* Read data from a binary file */
 if (str = strchr (message, ‘ ‘))
 {
 strcpy (path, ++str);
 if (file = fopen (path, “rb”))
 {
 n = 0;
 for (i = 0; i < argc; i ++)
 {
 sizes[i] = fread (data[i], sizeof (float), sizes[i], file);
 n += sizes[i];
 }
```

Multiple data items are handled by the RPC though the use of a second array subscript. For example, data[0][i] could be used to read in the first data item(s), and data[1][i] could read the second. Up to eight variables can be passed into and out of an RPC. There is no limit on the number of variables which are inside the RPC, and the size of the input variable array may be shortened by an RPC but not lengthened.

```
 fclose (file);
 sprintf (message, “%d value(s) read from file %s”, n, path);
 }
 else
 sprintf (message, “?Can’t open ‘%s’ for reading”, path);
```

Errors must be handled within the RPC itself. Error messages are denoted by a question mark at the beginning of the message string. Informational messages do not have question marks in front of them and are displayed on the GLI command line in the same way that other GLI messages are. If an informational message begins with an exclamation point (!) the string being returned by the RPC is assigned to the symbol RPC.MESSAGE, which is reserved by GLI for this purpose.

```
 sprintf (message, “?Missing file specification”);
 }

 else if (!strncmp(message, “write”, 5))
 {
 /* Write data to a binary file */
 if (str = strchr (message, ‘ ‘))
 {
 strcpy (path, ++str);
 if (file = fopen (path, “wb”))
 {
 n = 0;
 for (i = 0; i < argc; i ++)
 n += fwrite (data[i], sizeof (float), sizes[i], file);
 fclose (file);
 sprintf (message, “%d value(s) written to file %s”, n, path);
 }
 else
 sprintf (message, “?Can’t open ‘%s’ for writing”, path);
 }
 else
 sprintf (message, “?Missing file specification”);
 }
 else
 sprintf (message, “?Unknown command”, message);
}

main (argc, argv)
 int argc;
 char **argv;
{
 gli_registerrpc (argc, argv, my_proc);
}
```

The call to the function *gli_registerrpc* is necessary for running an RPC, as is passing into the function *gli_registerrpc* the arguments *argc* and *argv* in the proper order.

To compile and link an RPC with GLI, the following steps must be done.

```
%cc -c glirpcd.c
%cc -o glirpcd glirpcd.o $(GLI_HOME)/libgli.a
%glirpcd &
```

The first step compiles the RPC, and the next links the RPC with the GLI command library. The final command starts the RPC running as a background process on the computer. On some systems such as the Silicon Graphics running IRIX, it may be necessary to explicitly define the RPC library to the loader (ld), so the link command would look like:

```
%cc -o glirpcd glirpcd.o $(GLI_HOME)/libgli.a -lsun
```

on a Silicon Graphics workstation.

Glossary

attribute

A particular property that applies to a display element (output primitive). Examples: character spacing, polyline color index.

clipping

Removing parts of display elements that lie outside a given boundary, usually a window or viewport.

color table

A workstation dependent table, in which the entries specify the values of the red, green and blue intensities defining a particular color.

device coordinates

A coordinate expressed in a coordinate system that is device-dependent. Note: In GKS, DC units are meters on a device capable of a precisely scaled image, and appropriate workstation dependent units otherwise.

display element

A basic graphic element that can be used to construct a display image. Examples: polyline, polymarker, text. Also known as an output primitive.

fill area

A GKS output primitive consisting of a polygon (closed boundary) which may be hollow or filled with a uniform color, a pattern, or a hatch style.

locator device

A GKS logical input device providing a position in world coordinates.

marker

A glyph with a specific appearance which is used to identify a particular location. In GKS, markers may be scaled.

normalization transformation

A transformation that maps the boundary and interior of a window to the boundary and interior of a viewport. Note: In GKS this transformation maps positions in world coordinates to normalized device coordinates.

normalized device coordinates (NDC)

A coordinate specified in a device-independent intermediate coordinate system, normalized in some range, in GKS, typically 0 to 1. Note: In GKS, during an intermediate state the coordinates may lie outside the defined range, but associated clipping information insures that the output does not exceed the coordinate range [0,1]x[0,1].

output primitive

A display element. Output primitives in GKS are POLYLINE, POLYMARKER, TEXT, FILL AREA and PIXEL ARRAY.

polyline

A GKS output primitive consisting of a set of connected lines.

polymarker

A GKS output primitive consisting of a set of locations to be indicated by a marker.

text

A GKS output primitive consisting of a character string.

text extent rectangle

The boundary enclosing all the characters of a text string. It is formed by the topline of the largest ascending character, the bottomline of the largest descending character, the leftline of the leftmost character, and the rightline of the rightmost character.

text precision

An attribute describing the fidelity with which character position, character size, character orientation and character font of output matches that requested by the user. In order of increasing fidelity, the precisions are string, character, and stroke.

viewport

Part of the normalized device coordinate space. Note: In GKS, this definition is restricted to a rectangular region of normalized device coordinate space used in definition of the normalization transformation.

window

A predefined part of virtual space. Note: In GKS, this definition is restricted to a rectangular region of the world coordinate space used for the definition of the normalization transformation.

workstation

GKS is based on the concept of abstract graphical workstations which provide the logical interface through which the applications program controls physical devices.

workstation transformation

A transformation that maps the boundary and interior of a workstation window into the boundary and interior of a workstation viewport (part of display space), preserving aspect ratio. Note: In GKS, this transformation maps positions in normalized device coordinates to device coordinates. The effect of preserving aspect ratio is that the interior of the workstation window may not map to the whole workstation viewport.

workstation viewport

A portion of display space currently selected for graphics.

workstation window

A rectangular region within the normalized device coordinate system which is represented on a display space.

world coordinates (WC)

A device-independent Cartesian co-ordinate system used for specifying graphical input and output.