SAS® Technical Report P-246 SAS/TOOLKIT® Software: Graphics Capabilities

Release 6.08

SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS® Technical Report P-246, *SAS/TOOLKIT® Software: Graphics Capabilities, Release 6.08*, Cary, NC: SAS Institute Inc., 1992. 128 pp.

SAS® Technical Report P-246, SAS/TOOLKIT® Software: Graphics Capabilities, Release 6.08

ISBN 1-55544-524-1

1st printing, October 1992

Doc P19, 092992

Contents

Reference Aids

Tables

Using This Book

Purpose

SAS Technical Report P-246, *SAS/TOOLKIT Software: Graphics Capabilities, Release 6.08* documents the graphics capabilities of SAS/TOOLKIT software. The library of graphics routines described in this report are based on the graphics kernel system (GKS) standard.

"Using This Book" describes how you can best use this book. It describes the book's intended audience, the audience's prerequisite knowledge, the book's organization and its conventions, and the additional SAS System documentation that is available to you.

Audience and Prerequisites

Users must be familiar with SAS/TOOLKIT software and the SAS System. Users should also be familiar with the GKS standard. This book is intended for programmers who are experienced in programming in the C, PL/I, FORTRAN, or IBM 370 assembler languages.

Using SAS/TOOLKIT software to write graphics procedures also involves having

- SAS/TOOLKIT software, Release 6.08 or later.
- SAS/GRAPH software, Release 6.08 or later.
- base SAS software, Release 6.08 or later.
- one of these operating systems: AIX, CMS, HP-UX, MVS, OS/2, SunOS, or VMS.
- the appropriate language compiler for the language you are using. Supported compilers are
  - the SAS/C Compiler under MVS and CMS
  - the IBM Set/2 Compiler under OS/2 2.0
  - the native C compiler under AIX, HP-UX, SunOS, or VMS
  - the VS FORTRAN Version 3 compiler under MVS and CMS
  - the native FORTRAN compiler under AIX, HP-UX, SunOS, or VMS
  - the PL/I Optimizing Compiler under MVS and CMS
  - the native PL/I compiler under VMS
  - the IBM 370 Version 2 H-level Assembler.

How to Use This Book

This section gives an overview of the book's organization and content.

- Chapter 1: "Writing Graphics Procedures" explains the concepts used to create graphics output with SAS/TOOLKIT software.
- Chapter 2: "SAS_GKS Routines" provides reference information on all of the SAS_GKS routines in SAS/TOOLKIT software.

Conventions

This section covers the typographical conventions this book uses.

roman	is the basic type style used for most text.
UPPERCASE ROMAN	is used for references in the text to keywords of the SAS language, filenames, variable names, MVS JCL, CMS EXEC language, PL/I, FORTRAN, and IBM 370 assembler. Variable names from C language examples appear in uppercase in text only when they appear that way in the examples.
italic	is used to emphasize important information in text. Italic is also used to indicate variable values in examples and syntax.
`monospace`	is used to show examples of C or SAS programming code. In most cases, this book uses lowercase type for C programming statements and SAS code. Structure references and any variable names defined with the `#define` command are usually in uppercase monospace. Monospace is also used for C variable names that appear in text.

Using the SAS System

This book does not attempt to describe how to use the SAS System in detail. Note that once you have created a procedure or other SAS module, you can run your SAS module using any method of running the SAS System, including the SAS Display Manager System. For more information on running the SAS System, refer to the SAS companion for your operating system.

Additional Documentation

You may find the following documentation helpful when you are using SAS/TOOLKIT software and the SAS System.

SAS Documentation

There are many SAS System publications available. To receive a free *Publications Catalog* , write to the following address or call the following telephone number:

SAS Institute Inc.
Book Sales Department
SAS Campus Drive
Cary, NC 27513
919-677-8000

The books listed here should help you find answers to questions you may have about the SAS System in general or specific aspects of the SAS System.

- *SAS/TOOLKIT Software: Usage and Reference, Version 6, First Edition* (order #A56049) provides primary usage and reference information on using SAS/TOOLKIT software. This book includes appendices for the CMS, MVS, and VMS operating systems. It documents the basic steps and required routines for writing a SAS/TOOLKIT procedure. You must be familiar with this basic process before you can write a graphics procedure.
- SAS Technical Report P-245, *SAS/TOOLKIT Software: Changes and Enhancements, Releases 6.08 and 6.09* (order #A59162) updates *SAS/TOOLKIT Software: Usage and Reference, Version 6, First Edition.* It includes appendices for the OS/2 and UNIX operating systems.
- *SAS/GRAPH Software: Reference, Version 6, First Edition*, *Volume 1* and *Volume 2* (order #A56020) provides complete reference information on SAS/GRAPH procedures, statements, and options.
- *SAS/GRAPH Software: Usage, Version 6, First Edition* (order #A56021) documents how to use SAS/GRAPH software.

Chapter 1 Writing Graphics Procedures

Introduction

The User-Written Graphics Procedure Toolkit enables you to create graphics output within a user-written procedure. Through this toolkit, you can call the graphics routines used by SAS/GRAPH software to generate a custom graph. This toolkit may be used in conjunction with the User-Written Graphics Procedure Toolkit to access data in SAS data libraries for your graphs.

The User-Written Graphics Procedure Toolkit is based upon the Graphics Kernel System (GKS) standard, although it does not follow a strict interpretation, nor is it implemented on a particular level of GKS. GKS was used to provide a recognizable interface to the user. Because of its modularity, the standard allows for enhancements to the toolkit with the side effect of converting programs between versions of SAS/GRAPH software. These routines are those that are called by the DATA Step Graphics Interface (DSGI); you may notice many similarities between the two.

This chapter explains the concepts used to create graphics output with the graphics toolkit. The discussion provides an overview of the functions used. For complete details of each function, see Chapter 2, "SAS_GKS Routines."

Types of Functions

Query Functions

The graphics toolkit uses query functions to get information about the current operating environment. Because the graphics toolkit keeps track of the current settings of its attributes, you do not have to. Using the appropriate query functions yields the answer.

Setting Functions

The graphics toolkit uses setting functions to give values to graphics attributes. It remembers the values and uses them in appropriate drawing functions when they are issued.

Graph Management Functions

The graphics toolkit uses graph management functions to create, display, copy, or rename graphs in the current catalog.

Drawing Functions

The graphics toolkit uses drawing functions to draw various graphics primitives:

- arcs
- bars
- ellipses
- elliptical arcs

- lines
- markers
- pie slices
- polygons (filled areas)
- text

Creating Custom Graphs

You can produce custom graphs with the graphics toolkit either with or without using a data set to produce the graphics output.

To create custom graphs, you must provide the system with the following information:

- function calls to draw graphics elements
- the coordinates of the graphics elements in the output

In addition, you can specify the color, pattern, size, style, and position of these graphics elements.

Using the SAS_GKS Routines to Write Graphics Procedures

The following sections provide general information about using the graphics toolkit, including general steps for using it, how to produce and store graphs, how SAS/GRAPH global statements can be used with the graphics toolkit, and how to debug procedures. The sections also explain some of the basic concepts used in the graphics toolkit, including information about operating states and windowing systems.

To generate graphics output using the graphics toolkit, you generally follow these steps:

1. On a grid that matches the dimensions of the graphics output, sketch the output you want to produce.
2. Determine the coordinates of each graphics element.
3. Write the procedure to generate the graphics output. The basic steps are to

   a. load the graphics toolkit

   b. initialize the graphics toolkit

   c. open a graphics segment

   d. generate graphics elements

   e. close the graphics segment

   f. end the graphics toolkit

   g. unload the graphics toolkit

Note: The DISPLAY graphics option must be in effect for the graphics output to be displayed.

Producing and Storing Graphs

When you create or enhance graphs with the graphics toolkit, the graphics are displayed and stored as part of the graphics output. When you execute your procedure, the procedure creates a catalog entry using the name GKS as a catalog entry. By default, the catalog entry is stored in WORK.GSEG unless you specify another catalog with the SAS_GKSSCAT function.

If you generate another graph using a name that matches an existing catalog entry in the current catalog, the graphics toolkit uses the default naming conventions for the catalog entry.

If you want to store your output in a permanent library or in a different temporary catalog, you must use the SAS_GKSSCAT function. This function allows you to specify the libref and catalog name for the output catalog. Before you use the SAS_GKSSCAT function, you must allocate the libref using a LIBNAME statement before executing your procedure.

You can redisplay graphics output created by your procedure and stored in catalog entries using the GREPLAY procedure or the GRAPH window.

Using SAS/GRAPH Global Statements with User-Written Procedures

You can use some SAS/GRAPH global statements in conjunction with your user-written procedure. The graphics toolkit recognizes FOOTNOTE, GOPTIONS, and TITLE statements; however, it ignores AXIS, LEGEND, NOTE, PATTERN, and SYMBOL statements.

FOOTNOTE and TITLE statements affect user-written procedure graphics output the same way as they affect other SAS/GRAPH procedure output. When TITLE and FOOTNOTE statements are used, the output from the user-written procedure is placed in the procedure output area.

Some graphics toolkit functions override the graphics options. The following table lists the graphics toolkit functions that directly override graphics options.

Note: Some of these functions do not affect the GRAPH window because it is a separate task. If you want the graphics option to be global, use a GOPTION statement.

Function	Graphics Option That Is Overridden
SAS_GKSSCBA	CBACK=
SAS_GKSSCIX	COLORS=
SAS_GKSSDEV	DEVICE=
SAS_GKSSHPO	HPOS=
SAS_GKSSHSI	HSIZE=
SAS_GKSSVPO	VPOS=
SAS_GKSSVSI	VSIZE=
SAS_GKSSCTX	CTEXT=
SAS_GKSSFTX	FTEXT=
SAS_GKSSHTX	HTEXT=

The Current Window System

When the graphics toolkit draws graphics, it evaluates x and y coordinates in terms of the current window system, either a window you have defined or the default window system. Unless you define and activate a different window, the graphics toolkit uses the default window system.

The default window system assigns two arbitrary systems of units to the x and y axes. The default window guarantees a range of 0 through 100 in one direction (usually the y direction) and at least 0 through 100 in the other (usually the x direction). The ranges depend on the dimensions of your device. You can use the SAS_GKSQWNT function to determine the dimensions of your default window system.

You can define the x and y ranges to be any numeric range.For example, you can use -1000 to +2000 on the x axis and 30 to 35 on the yaxis. The units used are arbitrary.

Return Codes from SAS_GKS Routines

When a graphics toolkit function encounters an error, it returns a non-zero return code. The only exception are those function that return a status value. If you get a return code, you can refer to “Return Codes for Functions” in the Dictionary Chapter for a description of the error and why it might have occurred. The most common error is issuing a function while in an operating state that is not correct for the function.

Overview of SAS_GKS Routines

The following sections summarize the functions you can use to create graphics output with the user-written graphics procedure toolkit. These functions:

- initialize and terminate the graphics toolkit
- generate graphics elements
- control the appearance of graphics elements by setting attributes
- control the overall appearance of the graphics output
- perform management operations for the catalog

Table 1.1
Graphics Toolkit Functions

Associated Operation	Function	Function Description
Bundling Attributes	SAS_GKSSASF	set the aspect source flag of an attribute
	SAS_GKSSXFA SAS_GKSSXPL SAS_GKSSXPM SAS_GKSSXTX	select the bundle of attributes to use
	SAS_GKSSBFA SAS_GKSSBPL SAS_GKSSBPM SAS_GKSSBTX	assigns attributes to a bundle
Setting Attributes		

Associated Operation	**Function**	**Function Description**
color index	SAS_GKSSCIX	assigns a color name to color index
fill area	SAS_GKSSCFA	selects the color of the fill area
	SAS_GKSSSFA	selects the pattern when fill type is HATCH or PATTERN
	SAS_GKSSIFA	specifies the type of interior for the fill area
	SAS_GKSSPIX	sets a pattern for a pattern index
line	SAS_GKSSCPL	selects the color of the line
	SAS_GKSSTPL	sets the type (style) of the line
	SAS_GKSSWPL	specifies the width of the line
marker	SAS_GKSSCPM	selects the color of the marker
	SAS_GKSSSPM	determines the size of the marker
	SAS_GKSSTPM	sets the type of marker drawn
text	SAS_GKSSATX	specifies horizontal and vertical alignment of text
	SAS_GKSSCTX	selects the color of the text
	SAS_GKSSFTX	sets the font for the text
	SAS_GKSSHTX	selects the height of the text
	SAS_GKSSPTX	determines reading direction of text
	SAS_GKSSUTX	selects the angle of text
drawing order	SAS_GKSSPRI	selects drawing order (priority)
Setting Graph Attributes	SAS_GKSSASP	sets the aspect ratio
	SAS_GKSSCAT	selects the catalog to use
	SAS_GKSSCBA	selects the background color
	SAS_GKSSDEV	specifies the output device
	SAS_GKSSHPO	sets the number of columns in the graphics output area
	SAS_GKSSHSI	sets the width of the graphics output area in inches
	SAS_GKSSVPO	sets the number of rows in the graph output area
	SAS_GKSSVSI	sets the height of the graphics output area in inches
Managing Catalogs	SAS_GKSCSEG	copies a graph to another entry with the same catalog
	SAS_GKSDSEG	deletes a graph
	SAS_GKSISEG	inserts a previously created graph in the currently open segment.
	SAS_GKSRSEG	renames a graph

Associated Operation	**Function**	**Function Description**
	SAS_GKSPLAY	displays a graph
	SAS_GKSNAME	sets the name and description of a graph.
Drawing Graphics Elements		
arc	SAS_GKSDRAR	draws a circular arc
bar	SAS_GKSDRBA	draws a rectangle that can be filled
ellipse	SAS_GKSDREL	draws an ellipse that can be filled
elliptical arc	SAS_GKSDREA	draws an elliptical arc
fill area	SAS_GKSDRFA	draws a polygon that can be filled
line	SAS_GKSDRPL	draws a single line, a series of connected lines, or a dot
marker	SAS_GKSDRPM	draws one or more symbols
pie	SAS_GKSDRPI	draws a pie slice that can be filled
text	SAS_GKSDRTX	draws a character string
Initializing the Graphics Toolkit	SAS_GKSLOAD	loads image of graphics toolkit
	SAS_GKSOPKS	initializes graphics toolkit
	SAS_GKSOPWK	opens graphics toolkit
	SAS_GKSACWK	activates graphics toolkit
	SAS_GKSERWK	opens a segment to receive graphics primitives
Handling Messages	SAS_GKSDRMS	prints a message in the SAS log
Ending Graphics Toolkit	SAS_GKSUPWK	closes the currently open segment and optionally, displays it
	SAS_GKSDAWK	deactivates graphics toolkit
	SAS_GKSCLWK	closes graphics toolkit
	SAS_GKSCLKS	terminates graphics toolkit
	SAS_GKSLOAD	unloads image of graphics toolkit
Using Transformations	SAS_GKSSLNT	selects the transformation number of the viewport or window to use
Defining Viewports	SAS_GKSSVNT	sets the coordinates of the viewport and assigns it a transformation number
Defining Windows	SAS_GKSSWNT	sets the coordinates of the window and assigns it a transformation number
Defining Clip Area	SAS_GKSSCLP	sets the ability of a viewport to clip
Checking Attribute Bundles	SAS_GKSQASF	returns the aspect source flag of the attribute

Associated Operation	**Function**	**Function Description**
	SAS_GKSQXFA SAS_GKSQXPL SAS_GKSQXPM SAS_GKSQXTX	returns the index of the active bundle
	SAS_GKSQBFA SAS_GKSQBPL SAS_GKSQBPM SAS_GKSQBTX	returns information on the specified bundle
Checking Attributes		
color index	SAS_GKSQCNX	returns the color indices that currently have colors assigned to them
	SAS_GKSQCIX	returns the color name assigned to the color index
fill area	SAS_GKSQCFA	returns the color of the fill area
	SAS_GKSQSFA	returns the index of the pattern when the fill type is HATCH or PATTERN
	SAS_GKSQIFA	returns the index of the type of interior
	SAS_GKSQPIX	returns the patterns specified for a certain pattern index
line	SAS_GKSQCPL	returns the color index of the color the line
	SAS_GKSQTPL	returns the index of the type of line
	SAS_GKSQWPL	returns the width of the line
marker	SAS_GKSQCPM	returns the color index of the color markers
	SAS_GKSQSPM	returns the size of markers
	SAS_GKSQTPM	returns the type of marker drawn
text	SAS_GKSQATX	returns the horizontal and vertical alignment of text
	SAS_GKSQCTX	returns the color index of the color of text
	SAS_GKSQETX	returns the coordinates of text extent rectangle and the text concatenation point of the character string
	SAS_GKSQFTX	returns the text font
	SAS_GKSQHTX	returns the height of text
	SAS_GKSQPTX	returns the reading direction of text
	SAS_GKSQUTX	returns the character up vector in x vector and y vector
drawing order	SAS_GKSQPRI	returns the drawing order (priority)
Checking Graph Attributes	SAS_GKSQASP	returns the aspect ratio

Associated Operation	**Function**	**Function Description**
	SAS_GKSQCAT	returns the catalog to use
	SAS_GKSQCBA	returns the background color
	SAS_GKSQDEV	returns the output device
	SAS_GKSQHPO	returns the number of columns in the graphics output area
	SAS_GKSQHSI	returns the width of the graphics output area in inches
	SAS_GKSQVPO	returns the number of rows in the graphics output area
	SAS_GKSQVSI	returns the height of the graphics output area in inches
	SAS_GKSQMDS	returns the dimensions of maximum display area for the device in meters and pixels
Querying Catalogs	SAS_GKSQSEG	returns the names of graphs in the current catalog
	SAS_GKSQNSG	returns the number of graphs in the current catalog
	SAS_GKSQOSG	returns the name of the currently open graph
Checking System Status	SAS_GKSQOST	returns the current operating state
	SAS_GKSQAWK	returns whether or not the workstation is active
	SAS_GKSQOWK	returns whether or not the workstation is open
Checking Transformations	SAS_GKSQNNT	returns the active transformation number
Checking Viewports	SAS_GKSQVNT	returns the coordinates of the viewport assigned to the transformation number
Checking Windows	SAS_GKSQWNT	returns the coordinates of the window assigned to the transformation number

Creating Simple Graphics

Within any user-written graphics procedure, you need to follow these basic steps

1. Load the graphics toolkit. The function that loads the toolkit is SAS_GKSLOAD. It verifies that graphics are available and readies the bundle of graphics functions so that you can call them.
2. Initialize the graphics toolkit. The functions that initializes the toolkit are SAS_GKSOPKS, SAS_GKSOPWK, and SAS_GKSACWK. SAS_GKSOPKS loads the graphics sublibrary, SAS_GKSOPWK opens a workstation, and SAS_GKSACWK activates the workstation.
3. Open a graphics segment. Before you can submit graphics primitives, you must submit

the SAS_GKSERWK function. SAS_GKSERWK opens a graphic segment so that graphics primitives can be submitted.

4. Generate graphics elements. The toolkit can generate arcs, bars, ellipses, elliptical arcs, lines, markers, pie slices, polygons (fill areas), and text.These graphics elements are all produced with the drawing functions (that begin with SAS_GKSDR__). Drawing functions can only be submitted when a graphics segment is open. Therefore, they must be submitted between the SAS_GKSERWK and the SAS_GKSUPWK functions.
5. Close the graphics segment. Once the attribute and graphics statements have been entered, you must submit statements to close the graphics segment and output the graph. The SAS_GKSUPWK function closes the graphics segment currently open and, optionally, displays the graphics output.
6. End the graphics toolkit. The functions that end the graphics toolkit are SAS_GKSDAWK, SAS_GKSCLWK, and SAS_GKSCLKS. The SAS_GKSDAWK function deactivates the workstation, SAS_GKSCLWK closes the workstation, and SAS_GKSCLKS closes the graphics sublibrary and frees any memory allocated by the graphics toolkit.
7. Unload the graphics toolkit. The function that unloads the graphics toolkit is SAS_GKSUNLD. It unloads the graphics sublibrary and frees all resources used by the toolkit.

Notice that there are four pairs of functions that work together within a user-written procedure. The first pairs, SAS_GKSLOAD and SAS_GKSUNLD, load and unload the graphics toolkit. Within the first pair, SAS_GKSOPKS and SAS_GKSCLKS, begin and end the graphics toolkit. The next pair, SAS_GKSOPWK and SAS_GKSCLWK, open and close the workstation. The next pair, SAS_GKSACWK and SAS_GKSDAWK, activate and deactivate the workstation. The final pair, SAS_GKSERWK and SAS_GKSUPWK, begin and end a graphics segment. You can repeat these pairs throughout the procedure to produce multiple graphics output.

The order of these steps is controlled by operating states. Before any function can be submitted, the operating state in which that function can be submitted must be active. See "How Operating States Control the Order of Statements" later in this chapter.

Setting Attributes for Graphics Elements

The appearance of the graphics elements is determined by the settings of the attributes. Attributes control such aspects as height of text; text font; and color, size, and width of the graphics element. Attributes are set and reset with functions beginning with SAS_GKSS__. Functions beginning with SAS_GKSQ__ return the current setting of the attribute specified.

Each graphics primitive is associated with a particular set of attributes. Its appearance can only be altered by that set of attributes. This table lists the operators used with drawing functions to generate graphics elements and the attributes that control their appearance.

Table 1.2
Graphics Output Primitive Functions and Associated Attributes

Graphics Output Primitive	Functions	Associated Attributes
arc	SAS_GKSDRAR	line color line index line bundle line type line width
bar	SAS_GKSDRBA	fill color fill index fill bundle fill style fill type
ellipse	SAS_GKSDREL	fill color fill index fill bundle fill style fill type
elliptical arc	SAS_GKSDREA	line color line index line bundle line type line width
fill area	SAS_GKSDRFA	fill color fill index fill bundle fill style fill type
line	SAS_GKSDRPL	line color line index line bundle line type line width
marker	SAS_GKSDRPM	marker color marker index marker bundle marker size marker type
pie	SAS_GKSDRPI	fill color fill index fill bundle fill style fill type
text	SAS_GKSDRTX	text alignment text color text font text height text index text path text bundle text up vector

Attribute functions must precede the graphics primitive they control. Once an attribute is set, it controls any associated graphics primitives that follow. If you want to change the setting, you can issue another SAS_GKSS___ function with the new setting.

If you do not set an attribute before you submit a graphics primitive, the graphics toolkit uses the default value for the attribute. Refer to the dictionary chapter for the default values used for each attribute.

Operating States

The operating state of the graphics toolkit determines which functions may be issued at any point in the user-written procedure. You can only use a function when the operating state is appropriate for it. See the next section for a discussion of how functions should be ordered within the operating states.

The operating states defined by the graphics toolkit are

GKCL	facility closed, the initial state of the graphics toolkit. No graphical resources have been allocated.
GKOP	facility open. At this point, you may check the settings of the attributes.
SGOP	segment open. At this point, graphics output primitives may be generated.
WSAC	workstation active. When the workstation is active, it can receive drawing functions.
WSOP	workstation open. In this implementation, the graphics catalog, either the default or the one specified by the SAS_GKSSCAT function, is opened or created.

Refer to individual functions in the Dictionary for the operating states from which that function can be issued.

How Operating States Control the Order of Statements

Each function can only be submitted when certain operating states are active. This restriction affects the order of functions within the procedure. Generally, the operating states within a procedure follow this order:

```
GKCL -> GKOP -> WSOP -> WSAC -> SGOP -> WSAC -> WSOP -> GKOP -> GKCL
```

Functions That Change the Operating State

The functions described earlier actually control the changes to the operating state. For example, the SAS_GKSOPKS function must be submitted when the operating state is GKCL, the initial state of the graphics toolkit. SAS_GKSOPKS then changes the operating state to GKOP. The SAS_GKSOPWK function changes the operating state to WSOP, and the SAS_GKSACWK function changes the operating state to WSAC. The SAS_GKSERWK function must be submitted when the operating state is WSAC and before any graphics primitives are submitted. The reason it precedes graphics primitives is that it changes the operating state to SGOP, the operating state in which you can submit graphics primitives. The following list shows the change in the operating state due to specific functions:

```
SAS_GKSOPKS              GKCL -> GKOP
SAS_GKSOPWK              GKOP -> WSOP
SAS_GKSACWK              WSOP -> WSAC
SAS_GKSERWK              WSAC -> SGOP
SAS_GKSUPWK              SGOP -> WSAC
SAS_GKSDAWK              WSAC -> WSOP
SAS_GKSCLWK              WSOP -> GKOP
SAS_GKSCLKS              GKOP -> GKCL
```

Because these functions change the operating state, you must order all other functions so that the change in operating state is appropriate for the functions that follow. The following program statements show how the operating state changes from step to step in a typical graphics procedure. They also summarize the functions that can be submitted under each operating state. The functions that change the operating state are included as actual statements. Refer to the dictionary chapter for the operating states from which functions and routines can be submitted.

```
{
int rc;

UWPRCC(&proc);
SAS_XSPARSE(gramg(),NULL,&proc); /* parse the statements */

rc = SAS_GKSLOAD();
if (rc)
   SAS_XEXIT(XEXITERROR, 0);

/* GKCL - initial state; can execute:                      */
/*   1. setting functions that set attributes that affect the   */
/*      entire graphics output (some setting functions)        */
/*   2. some catalog management functions                      */

/* Step 1 - initialize                  */

rc = SAS_GKSOPKS();

/* GKOP - open state; can execute:                         */
/*   1. setting functions that set attributes that affect the   */
/*      entire graphics output (some setting functions)        */
/*   2. some catalog management functions                      */

/* Step 2 - open workstation            */

rc = SAS_GKSOPWK();

/* WSOP - open workstation state; can execute:             */
/*   1. setting functions that set attributes that affect the   */
/*      entire graphics output (some setting functions)        */
/*   2. some catalog management functions                      */

/* Step 3 - activate workstation      */

rc = SAS_GKSACWK();
```

```
/* WSAC - workstation is active; can execute:                  */
/*   1. most query routines                                    */
/*   2. some catalog management functions                      */
/*   3. setting functions that set attributes and bundles,     */
/*      viewports, windows, and transformations                */

/* Step 4 - open a graphics segment     */

rc = SAS_GKSERWK(NULL, 0);

/* SGOP - segment open; can execute:                           */
/* 1. any query routine                                        */
/* 2. any drawing function                                     */
/* 3. some catalog management functions                        */
/* 4. setting functions that set attributes and bundles,       */
/*    viewports, windows, and transformations                  */

/* Step 5 - execute graphics primitives */
rc = SAS_GKSDRBA(30.0, 30.0, 50.0, 50.0);

/* Step 6 - close the graphics segment  */

rc = SAS_GKSUPWK(1);

/* WSAC - workstation is active; can execute:                  */
/*  1. most query routines                                     */
/*  2. some catalog management functions                       */
/*  3. setting functions that set attributes and bundles,      */
/*     viewports, windows, and transformations                 */

/* Step 7 - deactivate workstation      */

rc = SAS_GKSDAWK();

/* WSOP - open workstation state; can execute:                 */
/*   1. setting functions that set attributes that affect the  */
/*      entire graphics output (some setting functions)        */
/*   2. some catalog management functions                      */

/* Step 8 - close workstation           */

rc = SAS_GKSCLWK();

/* GKOP - open state; can execute:                             */
/*   1. setting functions that set attributes that affect the  */
/*      entire graphics output (some setting functions)        */
/*   2. some catalog management functions                      */

/* Step 9 - terminate                   */

rc = SAS_GKSCLKS();

/* GKCL - initial state of graphics toolkit  */
```

```
rc = SAS_GKSUNLD();
SAS_XEXIT(XEXITNORMAL, 0);
}
```

Order of Functions

Functions within each operating state can technically be used in any order; however, once an attribute is set, it remains in effect until the SAS_GKSCLKS function is called or until you change its value. If you are producing multiple graphics output within the same procedure, the attributes for one output affect the ones that follow.

Notice that you can set attributes for the graphics primitives in several places. As long as the functions that set the attributes are executed before the graphics primitives, they will affect the graphics output. If you execute them after a graphics primitive, the primitive is not affected. See "Setting Attributes for Graphics Elements" earlier in this chapter.

The following program statements illustrate a more complex program. Notice that all attributes for a graphics primitive are executed before the graphics primitive. In addition, the SAS_GKSOPKS / SAS_GKSCLKS, SAS_GKSOPWK / SAS_GKSCLWK, and SAS_GKSACWK / SAS_GKSDAWK pairings are maintained.

```
{
int rc;
struct GPAIRF points[2];

SAS_XINIT();
rc = SAS_GKSLOAD();
if (rc)
   SAS_XEXIT(XEXITERROR, 0);

/* initialize */
rc = SAS_GKSOPKS();
rc = SAS_GKSOPWK();
rc = SAS_GKSACWK();

rc = SAS_GKSERWK(NULL, 0);

/* assign colors to color index */

rc = SAS_GKSSCIX(1, "BLUE    ");
rc = SAS_GKSSCIX(2, "RED     ");

/* define and display titles    */

rc = SAS_GKSSCTX(1);
rc = SAS_GKSSFTX("SWISSB   ");
rc = SAS_GKSSHTX(6.0);
rc = SAS_GKSDRTX(35.0, 93.0, 22, "Simple Graphics Output");

/* change the height and */
/* display second title  */
rc = SAS_GKSSHTX(4.0);
rc = SAS_GKSDRTX(53.0, 85.0, 20, "Created with UWPROC");
/* define and display footnotes */
/* using same text font and     */
/* color as defined for titles  */
rc = SAS_GKSSHTX(3.0);
rc = SAS_GKSDRTX(125.0, 1.0, 9, "GR20N03  ");

/* define and draw bar */
rc = SAS_GKSSCPL(2);
rc = SAS_GKSSWPL(5);

points[0].x =
points[1].x = 72.0;
points[0].y = 30.0;
points[1].y = 70.0;
rc = SAS_GKSDRPL(2, &points);

points[0].x = 52.0;
points[1].x = 92.0;
points[0].y =
points[1].y = 50.0;
rc = SAS_GKSDRPL(2, &points);

/* display graph and end graphics toolkit */
rc = SAS_GKSUPWK(1);
rc = SAS_GKSDAWK();
rc = SAS_GKSCLWK();
rc = SAS_GKSCLKS();

rc = SAS_GKSUNLD();
rc = SAS_XEXIT(XEXITNORMAL, 0);
}
```

Optional Features for User-Written Graphics Procedures

The following sections discuss optional features you can use with the graphics toolkit.

Bundling Attributes

The graphics toolkit allows you to bundle attributes. As a result, you can select a group of attribute values rather than having to select each one individually. This feature is useful if you use the same attribute settings over and over within the same procedure.

To use an attribute bundle, you assign the values of the attributes to a bundle index. When you want to use those attributes for a graphics primitive, you select the bundle rather than set each attribute separately.

Attributes That Can Be Bundled for Each Graphics Primitive

Each graphics primitive has a group of attributes associated with it that can be bundled. Only the attributes in that group can be assigned to the bundle. The next table shows the attributes that can be bundled for each graphics primitive.

You do not have to use attribute bundles for all graphics primitives if you use a bundle for one. You can define bundles for some graphics primitives and set the attributes individually for others. However, if the other graphics primitives are associated with the same attributes you have bundled and you do not want to use the same values, you can use other bundles to set the attributes, or you can set the attributes back to INDIVIDUAL.

Table 1.3
Attributes That Can Be Bundled for Each Graphics Primitive

Graphics Output Primitive	Associated Attributes That Can Be Bundled
arc (SAS_GKSDRAR)	line color
	line type
	line width
bar (SAS_GKSDRBA)	fill color
	fill style index
	fill interior type
ellarc (SAS_GKSDREA)	line color
	line type
	line width
ellipse (SAS_GKSDREL)	fill color
	fill style index
	fill interior type
fill area (SAS_GKSDRFA)	fill color
	fill style index
	fill interior type
polyline (SAS_GKSDRPL)	line color
	line type
	line width
polymarker (SAS_GKSDRPM)	marker color
	marker size
	marker type
pie (SAS_GKSDRPI)	fill color
	fill style index
	fill interior type

Graphics Output Primitive	**Associated Attributes That Can Be Bundled**
text (SAS_GKSDRTX)	text color
	text font

Assigning Attributes to a Bundle

To assign values of attributes to a bundle, you must

1. assign the values to a numeric bundle index with the SAS_GKSSB__ function. Each set of attributes that can be bundled uses a separate SAS_GKSSB__ function, where __ is the appropriate prefix for the set of attributes to be bundled. Valid for __ are fa, pl, pm, and tx.
2. set the aspect source flag (ASF) of the attributes to BUNDLED before you use the bundled attributes. You can use the SAS_GKSSASF function to set the ASF of an attribute. You need to execute a SAS_GKSSASF function for each attribute in the bundle.

The following example assigns the text attributes, color and font, to the bundle indexed by the number 1. As shown in the SAS_GKSSBTX function, the color for the bundle is green, the second color in the COLORS= graphics options. The font for the bundle is the ZAPF font.

```
{
int rc;

.
.    /* other procedure statements */
.
/* associate the bundle with the index 1 */

rc = SAS_GKSSBTX(1, 2, "ZAPF    ");
.
.    /* more statements */
.

/* assign the text attributes to a bundle */
rc = SAS_GKSSASF(TEXCOLASF, ASFBUND);
rc = SAS_GKSSASF(TEXFNTASF, ASFBUND);

/* draw the text */
rc = SAS_GKSDRTX(50.0, 50.0, 17, "Today is the day.");
```

The bundled attributes are used when an associated drawing function is executed. If the ASF of an attribute is not set to BUNDLED at the time a drawing function is executed, the graphics toolkit searches for a value to use in the following order:

1. the current value of the attribute

2. the default value of the attribute.

Selecting a Bundle

Once you have issued the SAS_GKSSASF and SAS_GKSSB__ functions, you can issue the SAS_GKSSX__ function to select the bundle. The following statement selects the bundle defined in the previous example:

```
/* invoke the bundle of text attributes */

rc = SAS_GKSSXTX(1);
```

The 1 in this example corresponds to the index number specified in the SAS_GKSSBTX function.

Defining Multiple Bundles for a Graphics Primitive

You can set up more than one bundle for graphics primitives by issuing another SAS_GKSSB__ function with a different index number. If you wanted to add a second attribute bundle for text to the previous example, you could issue the following statement:

```
/* define another attribute bundle for text */
rc = SAS_GKSSBTX(2, 3, "SWISS   ");
```

When you activate the second bundle, the graphics primitives for the text that follows will use the third color and the SWISS font.

When using a new bundle, you do not need to reissue the SAS_GKSSASF functions for the attributes that will be bundled. Once the ASF of an attribute has been set, the setting remains in effect until it is changed.

How the Toolkit Selects the Value of an Attribute to Use

Attributes that are bundled override any of the same attributes that are individually set. For example, you assign the line color green, the type 1, and the width 5 to a line bundle with the following statements:

```
rc = SAS_GKSSASF(LINCOLASF, ASFBUND);
rc = SAS_GKSSASF(LINTYPASF, ASFBUND);
rc = SAS_GKSSASF(LINWIDASF, ASFBUND);
rc = SAS_GKSSBPL(3, 2, 5, 1);
```

In subsequent statements, you activate the bundle, select other attributes for the line, and then draw a line:

```
/* activate the bundle */
rc = SAS_GKSSXPL(3);

/* select other attributes for the line */
rc = SAS_GKSSCPL(3);
rc = SAS_GKSSWPL(10);
rc = SAS_GKSSTPL(4);

/* draw a line from point (30,50) to (70,50) */
points[0].x = 30.0;
points[1].x = 70.0;
points[0].y =
points[1].y = 50.0;
rc = SAS_GKSDRPL(2, &points);
```

The color, type, and width associated with the line bundle are used rather than the attributes set just before the SAS_GKSDRPL function was executed. The line that is drawn is green (the second color from the colors list of the COLORS= graphics option), 5 units wide, and solid (line type 1).

During processing, the graphics toolkit chooses the value of an attribute using the following logic:

1. Get the index of the active line bundle.
2. Check the ASF of the line color attribute. If the ASF is INDIVIDUAL, the value selected with SAS_GKSSCPL is used; otherwise, the line color associated with the bundle index is used.
3. Check the ASF of the line type attribute. If the ASF is INDIVIDUAL, the value selected with SAS_GKSSTPL is used; otherwise, the line type associated with the bundle index is used.
4. Check the ASF of the line width attribute. If the ASF is INDIVIDUAL, the value selected with SAS_GKSSWPL is used; otherwise, the line width associated with the bundle index is used.
5. Draw the line using the appropriate color, type, and width for the line.

Disassociating an Attribute from a Bundle

To disassociate an attribute from a bundle, use the SAS_GKSSASF function to reset the ASF of the attribute to INDIVIDUAL. The following program statements demonstrate how to disassociate the attributes from the text bundle:

```
/* disassociate an attribute from a bundle */
rc = SAS_GKSSASF(TEXCOLASF, ASFINDIV);
rc = SAS_GKSSASF(TEXFNTASF, ASFINDIV);
```

Using Viewports and Windows

In the graphics toolkit, you can define viewports and windows. Viewports allow you to subdivide the graphics output area and insert existing graphs or draw graphics elements in smaller sections of the graphics output area. Windows define the coordinate system within a viewport and allow you to scale the graph or graphics elements drawn within the viewport.

The default viewport is defined as (0,0) to (1,1) with 1 being 100 percent of the graphics output area. If you do not define a viewport, graphics elements or graphs are drawn using the default.

The default window is defined so that a rectangle drawn from window coordinates (0,0) to (100,100) is square and fills the display in one dimension. The actual dimensions of the default window are device dependent. Use the SAS_GKSQWNT function to find the exact dimensions of your default window. You can define a window without defining a viewport. The coordinate system of the window is used with the default viewport.

If you define a viewport, you can position it anywhere in the graphics output area. You can define multiple viewports within the graphics output area so that more than one existing graph, part of a graph, or more than one graphics element can be inserted into the graphics output.

Transformations activate both a viewport and the associated window. The graphics toolkit maintains 21 (0 through 20) transformations. By default, transformation 0 is active. Transformation 0 always uses the entire graphics output area for the viewport and maps the window coordinates to fill the viewport. The definition of the viewport and window of transformation 0 may not be changed.

By default, the viewports and windows of all the other transformations (1 through 20) are set to the defaults for viewports and windows. If you want to define a different viewport or window, you must select a transformation number between 1 and 20.

You generally follow these steps when defining viewports or windows:

1. Define the viewport or window.
2. Activate the transformation so that the viewport or window is used for the output.

These steps can be used in any order; however, if you use a transformation you have not defined, the default viewport and window are used. Once you activate a transformation, the graphics elements drawn by the subsequent drawing functions are drawn in the viewport and window associated with that transformation.

Defining Viewports

You can define a viewport with the SAS_GKSSVNT(n) function, where n is the transformation number of the viewport you are defining. You can also use this function to define multiple viewports, each containing a portion of the graphics output area. You can then place a separate graph, part of a graph, or graphics elements within each viewport.

The following program statements divide the graphics output area into four subareas:

```
/* define the first viewport, indexed by 1 */
rc = SAS_GKSSVNT(1, .05, .05, .45, .45);

/* define the second viewport, indexed by 2 */
rc = SAS_GKSSVNT(2, .55, .05, .95, .45);

/* define the third viewport, indexed by 3 */
rc = SAS_GKSSVNT(3, .55, .55, .95, .95);

/* define the fourth viewport, indexed by 4 */
rc = SAS_GKSSVNT(4, .05, .55, .45, .95);
```

Once you define the viewports, you can insert existing graphs or draw graphics elements in each viewport by activating the transformation of that viewport.

Defining Windows

You can define a window by using the SAS_GKSSWNT(n) function, where n is the transformation number of the window you are defining. If you are defining a window for a viewport you have also defined, n must match the transformation of the viewport.

You can scale the x and y axes differently for a window. The following program statements scale the axes for each of the four viewports defined earlier in "Defining Viewports":

```
/* define the window for viewport 1 */
rc = SAS_GKSSWNT(1, 0.0, 50.0, 20.0, 100.0);

/* define the window for viewport 2 */
rc = SAS_GKSSWNT(2, 0.0, 40.0, 20.0, 90.0);

/* define the window for viewport 3 */
rc = SAS_GKSSWNT(3, 10.0, 25.0, 45.0, 100.0);

/* define the window for viewport 4 */
rc = SAS_GKSSWNT(4, 0.0, 0.0, 100.0, 100.0);
```

See "Scaling Graphs by Using Windows" later in this chapter for an example of using windows to scale graphs.

When you define a window for a viewport, the transformation numbers in the SAS_GKSSVNT and SAS_GKSSWNT functions must match in order for the graphics toolkit to activate them simultaneously.

Activating Transformations

Once you have defined a viewport or window, you must activate the transformation in order for the graphics toolkit to use the viewport or window. To activate the transformation, use the SAS_GKSSLNT(n) function where n has the same value as in SAS_GKSSVNT or SAS_GKSSWNT.

The following program statements illustrate how to activate the viewports and windows defined in the previous examples:

```
/* define the viewports */
.
.
.
/* define the windows */
.
.
.
/* activate the first transformation */
rc = SAS_GKSSLNT(1);
.
.  /* graphics primitive functions follow */
.
/* activate the second transformation */
rc = SAS_GKSSLNT(2);
```

```
.
.  /* graphics primitive functions follow */
.
/* activate the third transformation */
rc = SAS_GKSSLNT(3);
.
.  /* graphics primitive functions follow */
.
/* activate the fourth transformation */
rc = SAS_GKSSLNT(4);
.
.  /* graphics primitive functions follow */
.
```

When you activate these transformations, your display is logically divided into four subareas.

If you want to use the default viewport and window after selecting different ones, execute the SAS_GKSSLNT(0) function to reselect the default transformation.

Inserting Existing Graphs into Procedure Graphics Output

You can insert existing graphs into graphics output you are creating. The graph you insert must be in the same catalog in which you are currently working. Follow these steps to insert an existing graph:

1. Use the SAS_GKSSCAT function to set the output catalog to the catalog that contains the existing graph. You must have previously defined the libref in a LIBNAME statement or window before you run the procedure.
2. Define a viewport with the dimensions and position of the place in the graphics output where you want to insert the existing graph. SAS_GKSSVNT defines a viewport and SAS_GKSSWNT defines a window.
3. Define a window as (0,0) to (100,100) so that the inserted graph is not distorted. The graph must have a square area defined to avoid the distortion. If your device does not have a square graphics output area, the window defaults to the units of the device rather than (0,0) to (100,100) and may distort the graph.
4. Activate the transformation number n, as defined in the viewport function and the window function, using SAS_GKSSLNT(n).
5. Use the SAS_GKSISEG function with the name of the existing graph.

The following program statements provide an example of including an existing graph in the graphics output being created. The name of the the existing graph is MAP. LOCAL points to the library containing the catalog MAPCTLG. The coordinates of the viewport are percentages of the display.

```
.
.
.
/* select the output catalog to the */
/* catalog that contains 'map' */
rc = SAS_GKSSCAT("LOCAL   ", "MAPCTLG ");
```

```
.
.
.

/* define the viewport to contain the */
/* existing graph */
rc = SAS_GKSSVNT(1, .25, .45, .75, .9);
rc = SAS_GKSSWNT(1, 0, 0, 100, 100);

/* set the transformation number to the one */
/* defined in the viewport function */
rc = SAS_GKSSLNT(1);

/* insert the existing graph */
rc = SAS_GKSISEG("MAP     ");
```

These statements put the existing graph MAP in the upper half of the graphics output.

Generating Multiple Graphics Output in One Procedure

You can produce more than one graphics output within the same procedure. All statements between the SAS_GKSERWK and SAS_GKSUPWK functions will produce one graphics output. Each time the SAS_GKSUPWK function is executed, a graph is displayed. After the SAS_GKSCLKS function is executed, no more graphs are displayed. The SAS_GKSOPKS function must be executed again to produce more graphs.

Examples

The following examples show different uses for the graphics toolkit, illustrate some of its features such as defining viewports and windows, inserting existing graphs, angling text, using query routines, enlarging a segment of a graph, and scaling a graph.

Refer to the dictionary chapter for a complete description of each of the functions used in the examples.

Vertically Angling Text

This example generates a pie chart with text that changes its angle as you rotate around the pie. The functions position the text by aligning it differently depending on its location on the pie. In addition, the functions change the angle of the text so that it aligns with the spokes of the pie.

```
{
int rc;
struct GPAIRF points[2];

UWPRCC(&proc);
SAS_XSPARSE(gramg(),NULL,&proc); /* parse the statements */

/* prepare SAS/GRAPH software    */
/* to accept drawing statements  */

rc = SAS_GKSLOAD();
if (rc)
   SAS_XEXIT(XEXITERROR, 0);
rc = SAS_GKSOPKS();
rc = SAS_GKSOPWK();
rc = SAS_GKSACWK();

rc = SAS_GKSERWK(NULL, 0);

/* define and display arc  */
/* with intersecting lines */

rc = SAS_GKSSCPL(2);
rc = SAS_GKSSWPL(5);

rc = SAS_GKSDRAR(84.0, 50.0, 35.0, 0.0, 360.0);

points[0].x = 49.0;
points[1].x = 119.0;
points[0].y =
points[1].y = 51.0;
rc = SAS_GKSDRPL(2, &points);

points[0].x =
points[1].x = 84.0;
points[0].y = 15.0;
points[1].y = 85.0;
rc = SAS_GKSDRPL(2, &points);

/* define height of text */
rc = SAS_GKSSHTX(5.0);

/* mark 360 degrees on the arc */
/* using default align         */
rc = SAS_GKSDRTX(121.0, 50.0, 1, "0");

/* set text to align to the right and */
/* mark 180 degrees on the arc        */
rc = SAS_GKSSATX(RIGHTHORIZ, NORMVERT);
rc = SAS_GKSDRTX(47.0, 50.0, 3, "180");

/* set text to align to the center and */
/* mark 90 and 270 degrees on the arc  */
rc = SAS_GKSSATX(CENTHORIZ, NORMVERT);

rc = SAS_GKSDRTX(84.0, 87.0, 2, "90");
rc = SAS_GKSDRTX(84.0, 9.0, 3, "270");

/* reset text alignment to normal and    */
/* display coordinate values or quadrant */
rc = SAS_GKSSATX(NORMHORIZ, NORMVERT);
rc = SAS_GKSDRTX(85.0, 52.0, 11, "(0.0, +1.0)");

/* rotate text using text up vector and  */
/* display coordinate values or quadrant */
rc = SAS_GKSSUTX(1.0, 0.0);
rc = SAS_GKSDRTX(85.0, 49.0, 11, "(+1.0, 0.0)");

/* rotate text using text up vector and  */
/* display coordinate values or quadrant */
rc = SAS_GKSSUTX(0.0, -1.0);
rc = SAS_GKSDRTX(85.0, 50.0, 11, "(0.0, -1.0)");

/* rotate text using text up vector and  */
/* display coordinate values or quadrant */
rc = SAS_GKSSUTX(-1.0, 0.0);
rc = SAS_GKSDRTX(85.0, 52.0, 11, "(-1.0, 0.0)");

/* display graph and end graphics toolkit */
rc = SAS_GKSUPWK(1);
rc = SAS_GKSDAWK();
rc = SAS_GKSCLWK();
rc = SAS_GKSCLKS();
rc = SAS_GKSUNLD();

SAS_XEXIT(XEXITNORMAL, 0);
}
```

This example illustrates the following features:

- The series of functions, SAS_GKSLOAD, SAS_GKSOPKS, SAS_GKSOPWK, and SAS_GKSACWK begin the graphics toolkit.
- The SAS_GKSERWK function sets the graphics environment.
- The SAS_GKSSHTX, SAS_GKSSCPL, and SAS_GKSSWPL functions set attributes of the graphics primitives. See the SAS_GKSSCPL function in the dictionary chapter to see how this function chooses a color.
- The SAS_GKSDRAR function draws a empty pie. The arguments of the SAS_GKSDRAR function provide the coordinates of the starting point, the radius, and the beginning and ending angles of the arc.
- The SAS_GKSDRPL function draws a line. It provides the type of line, the coordinates of the beginning point, and the coordinates of the ending point.
- The SAS_GKSDRTX function draws the text. It sets the coordinates of the starting point of the text string as well as the text string to be written.
- The SAS_GKSSATX function aligns text to the center, left, or right of the starting point specified in the SAS_GKSDRTX function.
- The SAS_GKSSUTX function determines the angle at which the text is to be written.
- The SAS_GKSUPWK function closes the graphics segment.
- The series of functions SAS_GKSDAWK, SAS_GKSCLWK, SAS_GKSCLKS, and SAS_GKSUNLD ends the graphics toolkit.

Changing the Reading Direction of the Text

This example changes the reading direction of text.

```
{
int rc;

UWPRCC(&proc);
SAS_XSPARSE(gramg(),NULL,&proc); /* parse the statements */

/* prepare SAS/GRAPH software    */
/* to accept drawing statements  */
rc = SAS_GKSLOAD();
if (rc)
SAS_XEXIT(XEXITERROR, 0);
rc = SAS_GKSOPKS();
rc = SAS_GKSOPWK();
rc = SAS_GKSACWK();

rc = SAS_GKSERWK(NULL, 0);

/* define height of text */
rc = SAS_GKSSHTX(5.0);

/* display first text */
rc = SAS_GKSDRTX(105.0, 50.0, 5, "Right");

/* change text path so that text reads from */
/* right to left and display next text      */
rc = SAS_GKSSPTX(PATHLEFT);
rc = SAS_GKSDRTX(65.0, 50.0, 4, "Left");

/* change text path so that text reads up */
/* the display and display next text      */
rc = SAS_GKSSPTX(PATHUP);
rc = SAS_GKSDRTX(85.0, 60.0, 2, "Up");

/* change text path so that text reads down */
/* the display and display next text        */
rc = SAS_GKSSPTX(PATHDOWN);
rc = SAS_GKSDRTX(85.0, 40.0, 4, "Down");

/* display graph and end graphics toolkit */
rc = SAS_GKSUPWK(1);
rc = SAS_GKSDAWK();
rc = SAS_GKSCLWK();
rc = SAS_GKSCLKS();
rc = SAS_GKSUNLD();

SAS_XEXIT(XEXITNORMAL, 0);
}
```

Note: The SAS_GKSSPTX function changes the direction in which the text reads.

Using Viewports in a User-Written Procedure

This example uses the GCHART procedure to generate a graph, defines a viewport in which to display it, and inserts the GCHART graph into the graphics output being created by the graphics procedure.

These are the statements run in SAS to produce the GCHART graph:

```
/* create data set TOTALS */
data totals;
   length dept $ 7 site $ 8;
   do year=1984 to 1987;
      do dept='Parts','Repairs','Tools';
         do site='New York','Atlanta','Chicago','Seattle';
            sales=ranuni(97531)*10000+2000;
            output;
         end;
      end;
   end;
run;

/* define the footnote */
footnote h=3 j=r 'GR20N06  ';

/* generate pie chart from TOTALS */
/* and create catalog entry PIE   */
proc gchart data=totals;
   format sales dollar8.;
   pie site
      / type=sum
      sumvar=sales
      midpoints='New York' 'Chicago' 'Atlanta' 'Seattle'
      fill=solid
      cfill=green
      coutline=blue
      angle=45
      percent=inside
      value=inside
      slice=outside
      noheading
      name='pie';
run;
```

The following code shows how to insert a graph:

```
{
int rc;

UWPRCC(&proc);
SAS_XSPARSE(gramg(),NULL,&proc); /* parse the statements */
/* prepare SAS/GRAPH software    */
/* to accept drawing statements  */
rc = SAS_GKSLOAD();
if (rc)
SAS_XEXIT(XEXITERROR, 0);
rc = SAS_GKSOPKS();
rc = SAS_GKSOPWK();
rc = SAS_GKSACWK();

rc = SAS_GKSERWK(NULL, 0);

/* define and activate viewport for inserted graph */
rc = SAS_GKSSVNT(1, .15, .05, .85, .90);
rc = SAS_GKSSWNT(1, 0.0, 0.0, 100.0, 100.0);
rc = SAS_GKSSLNT(1);

/* insert graph created from GCHART procedure */
rc = SAS_GKSISEG("PIE     ");

/* display graph and end graphics toolkit */
rc = SAS_GKSUPWK(1);
rc = SAS_GKSDAWK();
rc = SAS_GKSCLWK();
rc = SAS_GKSCLKS();
rc = SAS_GKSUNLD();

SAS_XEXIT(XEXITNORMAL, 0);
}
```

Features not explained in previous examples are described here:

- □ A graph can be created by another SAS/GRAPH procedure and inserted into graphics output produced by your procedure. In this case, the NAME= option in the PIE statement of the GCHART procedure names the graph, PIE, to be inserted.
- □ The SAS_GKSSVNT function defines the section of the graphics output area into which PIE is inserted. The dimensional ratio of the viewport should match that of the entire graphics output area so that the inserted graph is not distorted.
- □ The SAS_GKSSWNT function defines the coordinate system to be used within the viewport. In this example, the coordinates (0.0,0.0) to (100.0,100.0) are used. These coordinates provide a square area to insert the graph and preserve the aspect ratio of the GCHART graph.
- □ The SAS_GKSSLNT function activates the transformation for the defined viewport and window.
- □ The SAS_GKSISEG function inserts the existing graph, PIE, into the one being created by your procedure. If no viewport has been explicitly defined, the default viewport, which is the entire graphics output area, is used.

Scaling Graphs by Using Windows

This example uses the GPLOT procedure to generate a plot of AMOUNT*MONTH and store the graph in a permanent catalog. Code in the user-written procedure then scales the graph by defining a window and inserting the GPLOT graph into that window.

This example also requires a LIBNAME statement, which allocates a libref to a permanent SAS data library.

These are the SAS statements to produce the GPLOT graph:

```
data earn;
   input month amount;
   cards;
1 2.1
2 3
3 5
4 6.4
5 9
6 7.2
7 6
8 9.8
9 4.4
10 2.5
11 5.75
12 4.35
;
run;

   /* define the footnote for the first graph */
footnote j=r 'GR20N07(a)  ';

   /* define axis and symbol characteristics */
axis1 label=(color=green 'Millions of Dollars')
      order=(1 to 10 by 1)
      value=(color=green);
axis2 label=(color=green 'Months')
      order=(1 to 12 by 1)
      value=(color=green Tick=1 'Jan' Tick=2 'Feb' Tick=3 'Mar'
            Tick=4 'Apr' Tick=5 'May' Tick=6 'Jun'
            Tick=7 'Jul' Tick=8 'Aug' Tick=9 'Sep'
            Tick=10 'Oct' Tick=11 'Nov' Tick=12 'Dec');

symbol value=M font=special height=8 interpol=join
       color=blue width=3;

   /* generate a plot of AMOUNT * MONTH,          */
   /* send output to permanent catalog PLOTS, and */
   /* store in member ANEARN                      */
proc gplot data=earn gout=sampsrc.plots;
   plot amount*month
      / haxis=axis2
      vaxis=axis1
      name='anearn';
```

```
run;
```

The following code shows how to insert and scale a graph:

```
{
int rc;

UWPRCC(&proc);
SAS_XSPARSE(gramg(),NULL,&proc); /* parse the statements */
/* prepare SAS/GRAPH software    */
/* to accept drawing statements  */
rc = SAS_GKSLOAD();
if (rc)
   SAS_XEXIT(XEXITERROR, 0);
rc = SAS_GKSSCAT('sampsrc', 'plots');

rc = SAS_GKSOPKS();
rc = SAS_GKSOPWK();
rc = SAS_GKSACWK();

rc = SAS_GKSERWK(NULL, 0);

/* define viewport and window for inserted graph */
rc = SAS_GKSSVNT(1, .20, .30, .90, .75);
rc = SAS_GKSSWNT(1, 15.0, 15.0, 95.0, 75.0);
rc = SAS_GKSSLNT(1);

/* insert graph previously created */
rc = SAS_GKSISEG("ANEARN  ");

/* display graph and end graphics toolkit */
rc = SAS_GKSUPWK(1);
rc = SAS_GKSDAWK();
rc = SAS_GKSCLWK();
rc = SAS_GKSCLKS();
rc = SAS_GKSUNLD();

SAS_XEXIT(XEXITNORMAL, 0);
}
```

Features not explained in previous examples are described here:

- The GOUT= option in the PROC GPLOT statement specifies the catalog in which to store the graph ANEARN. Since the graph is not stored in the default catalog, the catalog in which it is stored must be specified in a SAS_GKSSCAT function.
- The SAS_GKSSCAT function selects the catalog in which to store output. To insert an existing graph into user-written procedure graphics output, you must have stored the existing graph in the catalog specified in the SAS_GKSSCAT function. In this example, SAS_GKSSCAT selects the same catalog in which the GPLOT graph ANEARN is stored. If no SAS_GKSSCAT function were included, the default catalog, WORK.GSEG, would be used. In order to include ANEARN in the graph, you would also have to store it in WORK.GSEG.

□ The SAS_GKSSWNT function scales the plot with respect to the viewport that is defined. The x axis is scaled from 15 to 95, and the y axis is scaled from 15 to 75. If no viewport were explicitly defined, the window coordinates would be mapped to the default viewport, the entire graphics output area.

Enlarging an Area of a Graph by Using Windows

This example illustrates how you can enlarge a section of a graph by using windows. In the first section of code, the program statements generate graphics output that contains four pie charts. The second section defines a window that enlarges the bottom-left quadrant of the graphics output and inserts FOURPIES into that window.

```
{
int rc;

UWPRCC(&proc);
SAS_XSPARSE(gramg(),NULL,&proc); /* parse the statements */

/* prepare SAS/GRAPH software    */
/* to accept drawing statements  */
rc = SAS_GKSLOAD();
if (rc)
   SAS_XEXIT(XEXITERROR, 0);
rc = SAS_GKSOPKS();
rc = SAS_GKSOPWK();
rc = SAS_GKSACWK();

rc = SAS_GKSERWK(NULL, 0);

/* define and draw first pie chart */
rc = SAS_GKSSCFA(4);
rc = SAS_GKSSIFA(FILLSOL);
rc = SAS_GKSDRPI(30.0, 75.0, 22.0, 0.0, 360.0);

/* define and draw second pie chart */
rc = SAS_GKSSCFA(1);
rc = SAS_GKSSIFA(FILLSOL);
rc = SAS_GKSDRPI(30.0, 25.0, 22.0, 0.0, 360.0);

/* define and draw third pie chart */
rc = SAS_GKSSCFA(3);
rc = SAS_GKSSIFA(FILLSOL);
rc = SAS_GKSDRPI(90.0, 75.0, 22.0, 0.0, 360.0);

/* define and draw fourth pie chart */
rc = SAS_GKSSCFA(2);
rc = SAS_GKSSIFA(FILLSOL);
rc = SAS_GKSDRPI(90.0, 25.0, 22.0, 0.0, 360.0);

/* don't display graph and end graphics toolkit */
rc = SAS_GKSUPWK(0);
rc = SAS_GKSRSEG("GKS     ", "FOURPIES");
rc = SAS_GKSDAWK();
rc = SAS_GKSCLWK();
rc = SAS_GKSCLKS();
rc = SAS_GKSUNLD();

SAS_XEXIT(XEXITNORMAL, 0);
}
```

Now for the section of code that produces the zoomed picture:

```
{
int rc;

UWPRCC(&proc);
SAS_XSPARSE(gramg(),NULL,&proc); /* parse the statements */

/* prepare SAS/GRAPH software    */
/* to accept drawing statements  */
rc = SAS_GKSLOAD();
if (rc)
   SAS_XEXIT(XEXITERROR, 0);
rc = SAS_GKSOPKS();
rc = SAS_GKSOPWK();
rc = SAS_GKSACWK();

rc = SAS_GKSERWK(NULL, 0);

/* define and activate a window     */
/* that will enlarge the lower left */
/* quadrant of the graph            */
rc = SAS_GKSSWNT(1, 0.0, 0.0, 50.0, 50.0);
rc = SAS_GKSSLNT(1);

/* insert the previous graph into */
/* window 1                       */
rc = SAS_GKSISEG("FOURPIES");

/* display graph and end graphics toolkit */
rc = SAS_GKSUPWK(1);
rc = SAS_GKSDAWK();
rc = SAS_GKSCLWK();
rc = SAS_GKSCLKS();
rc = SAS_GKSUNLD();

SAS_XEXIT(XEXITNORMAL, 0);
}
```

Features not explained in previous examples are described here:

- The SAS_GKSSWNT function defines a window into which the graph is inserted. In this example, no viewport is defined, so the window coordinates map to the default viewport, which is the entire graphics output area. The result of using the default viewport is that only the portion of the graph enclosed by the coordinates of the window is displayed.

- The SAS_GKSISEG function inserts a graph that was previously generated in another procedure or section of code. If you want to insert output created by a user-written procedure, the output to be inserted must be closed.

Using Query Functions in a User-Written Procedure

This example illustrates how to invoke query functions and how to display the returned values in the SAS log.

This example assigns a predefined color to color index 2 and then invokes a query routine to get the name of the color associated with color index 2. The value returned from the query function is displayed in the log and written to a data set.

```
{
int   rc;
char8 color;

UWPRCC(&proc);
SAS_XSPARSE(gramg(),NULL,&proc); /* parse the statements */

/* prepare SAS/GRAPH software    */
/* to accept drawing statements  */
rc = SAS_GKSLOAD();
if (rc)
   SAS_XEXIT(XEXITERROR, 0);
rc = SAS_GKSOPKS();
rc = SAS_GKSOPWK();
rc = SAS_GKSACWK();

rc = SAS_GKSSCIX(2, "ORANGE  ");

/* check color associated with color index 2 and */
/* display the value in the LOG window           */
rc = SAS_GKSQCIX(2, color);
SAS_XPSLOG("Color name in 2nd color index is %8b.", color);

rc = SAS_GKSDAWK();
rc = SAS_GKSCLWK();
rc = SAS_GKSCLKS();
rc = SAS_GKSUNLD();

SAS_XEXIT(XEXITNORMAL, 0);
}
```

Features not explained in previous examples are described here:

- The SAS_GKSSCIX function assigns the predefined color ORANGE to the color index 2.
- The SAS_GKSQxxx functions check the current value of an attribute. In this example, the SAS_GKSQCIX functions returns the color associated with color index 2.

- A SAS_XPSLOG function displays the value of the color argument in the log.

Chapter 2 SAS_GKS Routines

Overview

This chapter contains detailed descriptions of each function that can be called in the User-written Graphics Procedure Toolkit.

The functions are discussed in the following order:

- utility functions
  - initializing graphics
  - printing error messages
  - terminating graphics
- query functions
- drawing functions
- graph management functions
- graphics attribute setting functions.

For each command, this chapter provides the statement syntax, other argument definitions, and notes about using the functions, operating states, and return codes. Operating states are summarized in “Operating States” later in this chapter. Values and structures used by the toolkit are included in uwproc.h, the header file supplied with the toolkit for use by the procedure writer.

The syntax for all functions contains the argument *rc*. This argument must be a variable of type int and can be a different variable name for each routine.

The *rc* argument is used to debug user-written graphics procedures. It contains the return code of the function call. If the return is any value other than 0 and the function is not a function querying a status value, the function did not execute properly.

Each function has a different set of possible return codes. The return codes are listed in the heading for the routine or function. Refer to "Return Codes for Functions" later in this chapter for an explanation of the return codes.

Operating States

This list summarizes the operating states. For a detailed discussion of operating states, refer to Chapter 1, "Writing Graphics Procedures."

GKCL	indicates the facility is closed. This is the initial state of the graphics interface.
GKOP	indicates the facility is open. You may check the settings of attributes.
SGOP	indicates the segment is open. Graphics output can be generated.
WSAC	indicates the workstation is active.
WSOP	indicates the workstation is open. The graphics catalog is opened or created.

Utility Functions

Utility functions enable you to initialize graphics, print error messages, and terminate graphics.

SAS_GKSLOAD

Loads the graphics interface for user-written procedures

Operating States: ——

Return Codes: 0, 1, 2, 3

Resulting Operating State: GKCL

Syntax

```
rc = SAS_GKSLOAD();
```

The SAS_GKSLOAD function loads the image that contains the graphics toolkit.

Argument Definitions

Variable	Type	Description
rc	int	returns the return code of the function call (0 if OK, 1 if memory could not be allocated, 2 if image not found, or 3 if the SAS/GRAPH product is not licensed). These return codes do not work in conjunction with SAS_GKSERR.

SAS_GKSOPKS

Initializes the graphics interface for user-written procedures

Operating States: GKCL

Return Codes: 0, 1, 307

Resulting Operating State: GKOP

Syntax

```
rc = SAS_GKSOPKS();
```

The SAS_GKSOPKS function readies the library that contains the SAS/GRAPH graphics routines. This function moves the operating state from GKCL to GKOP.

Argument Definitions

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSOPWK

Opens a workstation for the graphics interface

Operating States: GKOP

Return Codes: 0, 2, 26

Resulting Operating State: WSOP

Syntax

```
rc = SAS_GKSOPWK();
```

The SAS_GKSOPWK function opens a workstation. A workstation is a Graphics Kernel Standard (GKS) concept. GKS allows for multiple workstations to open at the same time; however, for these applications, you always use exactly one workstation. This function moves the operating state from GKOP to WSOP.

Argument Definitions

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSACWK

Activates the open workstation

Operating States: WSOP

Return Codes: 0, 7

Resulting Operating State: WSAC

Syntax

```
rc = SAS_GKSACWK();
```

The SAS_GKSACWK function activates a workstation. A workstation is a Graphics Kernel Standard (GKS) concept. GKS allows for multiple workstations to be open at the same time; however, for these applications, you always use exactly one workstation. This function moves the operating state from WSOP to WSAC.

Argument Definitions

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSERR

Prints the specified interface error message

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSERR(code);
```

The SAS_GKSERR function displays the message that corresponds to the error code entered. It works with all routines in the graphics toolkit except SAS_GKSLOAD and SAS_GKSUNLD.

Argument Definitions

Variable	Type	Description
code	int	should be the value of a return code received from some previous function.
rc	int	returns the return code of the function call.

SAS_GKSCLKS

Terminates the graphics interface for the user-written procedures

Operating States: GKOP

Return Codes: 0, 2

Resulting Operating State: GKCL

Syntax

```
rc = SAS_GKSCLKS();
```

The SAS_GKSCLKS function closes the library that contains SAS/GRAPH routines. This function should be issued to free memory allocated by previously called functions. This function moves the operating state from GKOP to GKCL.

Argument Definitions

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSCLWK

Closes the workstation

Operating States: WSOP

Return Codes: 0, 7

Resulting Operating State: GKOP

Syntax

```
rc = SAS_GKSCLWK();
```

The SAS_GKSCLWK function closes the workstation. This function moves the operating state from WSOP to GKOP.

Argument Definitions

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSDAWK

Deactivates the workstation

Operating States: WSAC

Return Codes: 0, 3

Resulting Operating State: WSOP

Syntax

```
rc = SAS_GKSDAWK();
```

The SAS_GKSDAWK function deactivates the workstation. This function moves the operating state from WSAC to WSOP.

Argument Definitions

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSUNLD

unloads the graphics interface for user-written procedures

Operating States: GKCL

Return Codes: 0

Resulting Operating State: ——

Syntax

```
rc = SAS_GKSUNLD();
```

The SAS_GKSUNLD function unloads the image that contains the graphics toolkit.

Argument Definitions

Variable	Type	Description
rc	int	returns the return code of the function call.

Query Functions

These functions allow you to check the current attribute settings. When you are using these functions, remember the following that many of the arguments return values and need not be initialized to any particular value.

SAS_GKSQASF

Queries the status of the aspect source flag (ASF) of a particular attribute

Operating States: GKOP, WSOP, WSAC, SGOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQASF(attribute, status);
```

The SAS_GKSQASF function returns the status of the aspect source flag (ASF) of a particular attribute. Possible ASF values are BUNDLED (associated with a bundle index) and INDIVIDUAL (separate from a bundle index). SAS_GKSQASF returns the default value INDIVIDUAL if you have not set the ASF for an attribute.

Argument Definitions

Variable	Type	Description
attribute	int	a value between 0 and 10, inclusive, indicative of which aspect source flag is queried. Use the #define values for the aspect source flag in the file uwproc.h or use one of the following values from the table:

```
                  Integer
  Value          Equivalent      Purpose
FILCOLASF            0         fill color ASF
FILSTYASF            1         fill style ASF
FILINTASF            2         interior style ASF
FILINCOLASF          3         line color ASF
LINTYPASF            4         line type ASF
LINWIDASF            5         line width ASF
MARCOLASF            6         marker color ASF
MARSIZASF            7         marker size ASF
MARTYPASF            8         marker type ASF
TEXCOLASF            9         text color ASF
TEXFNTASF           10         text font ASF
```

Variable	Type	Description
status	char16	returns either the value BUNDLED or INDIVIDUAL, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSQASP

Finds the value of the aspect ratio

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSQASP(aspect);
```

The SAS_GKSQASP function returns the current aspect ratio used to draw graphics output. SAS_GKSQASP searches for the current aspect ratio in the following order:

```
* the aspect ratio set with the SAS_GKSSASP function
* the ASPECT= graphics option
* the device's default aspect ratio found in the device entry.
```

Argument Definitions

Variable	Type	Description
aspect	ptr to a double	returns the aspect ratio.
rc	int	returns the return code of the function call.

SAS_GKSQATX

Finds the horizontal and vertical alignment of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQATX(halign, valign);
```

The SAS_GKSQATX function returns the current horizontal and vertical text alignment. If no values have been previously selected with the SAS_GKSSATX function, SAS_GKSQATX returns the default value NORMAL for both halign and valign.

Argument Definitions

Variable	Type	Description

halign	char8	returns the horizontal alignment as one of the following values, in upper case, padded with blanks:
		`* CENTER`
		`* LEFT`
		`* NORMAL`
		`* RIGHT`

valign	char8	returns the vertical alignment as one of the following values, in upper case, padded with blanks:
		`* BASE`
		`* BOTTOM`
		`* HALF`
		`* NORMAL`
		`* TOP`

rc	int	returns the return code of the function call.

SAS_GKSQAWK

Finds whether the interface is active

Operating States: All

Return Codes: 29, 30

Syntax

```
status = SAS_GKSQAWK();
```

The SAS_GKSQAWK function asks if the workstation is active. When the workstation is active, you can execute certain graphics toolkit functions.

Argument Definitions

Variable	Type	Description
status	int	returns the status of the workstation Use the #define values for the workstation status in the file uwproc.h or use one of the following values from the table:

```
                 Integer
  Value    Equivalent        Meaning
EWSISACT       29      workstation active
EWSNTACT       30      workstation inactive
```

SAS_GKSQBFA

Finds the fill area attributes associated with a bundle index

Operating States: GKOP, WSOP, WSAC, SGOP

Return Codes: 0, 8, 75, 76

Syntax

```
rc = SAS_GKSQBFA(index, color-index, interior,style-index)
```

The SAS_GKSQBFA function returns the color, type of interior, and fill pattern associated with a specific fill bundle. If the bundle indicated by index has not been previously defined with a SAS_GKSSBFA function, the error code EFILNOTP (75) is returned.

Argument Definitions

Variable	Type	Description
index	int	indicates the fill bundle to check. Valid values are 1 to 20, inclusive.

color-index	ptr to an int	returns the color index of the fill color associated with the bundle. The color index returned corresponds to a color specification in the following order:
		
		`a color index assigned to a color name with the SAS_GKSSCIX function`
		
		`the nth color in the colors list of the COLORS= graphics option`
		
		`the nth color in the device's default colors list found in the device entry.`

interior	char8	returns the style of the interior associated with the bundle index that is one of the following values, in upper case, padded with blanks:
		
		`HATCH`
		
		`HOLLOW`
		
		`PATTERN`
		
		`SOLID`

style-index	ptr to an int	returns the index of the fill pattern associated with the bundle. See the SAS_GKSSSFA function later in this chapter for the fill patterns represented by style-index.
rc	int	returns the return code of the function call.

SAS_GKSQBPL

Finds the bundle of line attributes associated with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 60, 61

Syntax

```
rc = SAS_GKSQBPL(index, color-index, width, type);
```

The SAS_GKSQBPL function returns the color, width, and line type associated with a specific line bundle. If the bundle indicated by index has not previously defined with a SAS_GKSSBPL function, the error code ENOLINEX (61) is returned.

Argument Definitions

Variable	Type	Description
index	int	indicates the fill bundle to check. Valid values are 1 to 20, inclusive.

color-index	ptr to an int	returns the color index of the line color associated with the bundle. The color index returned corresponds to a color specification in the following order:
		`* a color index assigned with the SAS_GKSSCIX function`
		`* the nth color in the colors list of the COLORS=graphics option`
		`* the nth color in the device’s default colors list.`

width	ptr to a int	returns the line width (in pixels) associated with the bundle.
type	ptr to a int	returns the index of the line type (style) associated with the bundle.
rc	int	returns the return code of the function call.

SAS_GKSQBPM

Finds the bundle of marker attributes associated with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 64, 65

SAS_GKSQBPM *continued*

Syntax

```
rc = SAS_GKSQBPM(index, color-index, size, type);
```

The SAS_GKSQBPM function returns the color, size, and type of marker associated with a specific marker bundle. If the bundle indicated by index has not been previously defined with the SAS_GKSSBPM function, the error code ENOMARKX (65) is returned.

Argument Definitions

Variable	Type	Description
index	int	indicates the index of the fill bundle to check. Valid values are 1 to 20, inclusive.

color-index	ptr to an int	returns the color index of the marker color associated with the bundle. The color index returned corresponds to a color specification in the following order:
		`* a color index assigned with the SAS_GKSSCIX function`
		`* the nth color in the colors list of the COLORS=graphics option`
		`* the nth color in the device's default colors list.`

size	ptr to an double	returns the marker size in units of the current window system.
type	ptr to an int	returns the index of the marker type associated with the bundle. See the SAS_GKSSTPM function for an explanation of the marker indices.
rc	int	returns the return code of the function call.

SAS_GKSQBTX

Finds the attribute settings associated with a text bundle

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 68, 69

Syntax

```
rc = SAS_GKSQBTX(index, color-index, font);
```

The SAS_GKSQBTX function returns the color and font associated with a specific text bundle. If the bundle indicated by index has not been previously defined with the SAS_GKSSBTX function, the error code ENOTEXTX (69) is returned.

Argument Definitions

Variable	Type	Description
index	int	indicates the fill bundle to check. Valid values are 1 to 20, inclusive.

color-index	ptr to an int	returns the color index of the text color associated with the bundle. The color index returned corresponds to a color specification in the following order:
		`* a color index assigned with the SAS_GKSSCIX function`
		`* the nth color in the colors list of the COLORS=graphics option`
		`* the nth color in the device's default colors list.`

font	char8	returns the text font associated with the bundle, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSQCAT

Finds the libref and the name of the current output catalog

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSQCAT(libref, memname);
```

The SAS_GKSQCAT function returns the libref and the name of the current output catalog. SAS_GKSQCAT returns the default catalog, WORK.GSEG, if no other catalog has been specified with the SAS_GKSSCAT function.

Argument Definitions

Variable	Type	Description
libref	char8	returns the libref of the library in which the current catalog is stored, in upper case, padded with blanks.
memname	char8	returns the name of the current output catalog, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSQCBA

Finds the current background color

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSQCBA(cback);
```

The SAS_GKSQCBA function returns the current background color. SAS_GKSQCBA searches for the current background color in the following order:

```
* the background color selected with the SAS_GKSSCBA function
* the CBACK= graphics option
* the default background color for the device found in the device
entry.
```

Argument Definitions

Variable	Type	Description
cback	char8	returns the background color name, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSQCFA

Finds the color index of the color to be used to draw fill areas

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQCFA(color-index);
```

The SAS_GKSQCFA function returns the current fill color. If a SAS_GKSSCFA function has not been previously submitted to initialize the fill color, SAS_GKSQCFA returns the default value, 1. The color index returned corresponds to a color specification in the following order:

```
* the color assigned to a color name with the SAS_GKSSCFA function
* the nth color in the colors list of the COLORS= graphics options
* the nth color in the device's default colors list found in the
```

```
device entry.
```

Argument Definitions

Variable	Type	Description
color-index	ptr to an int	returns the color index of the fill color currently selected.
rc	int	returns the return code of the function call.

SAS_GKSQCIX

Finds the color name associated with a color index

Operating States: SGOP

Return Codes: 0, 4, 86, 87

Syntax

```
rc = SAS_GKSQCIX(color-index, color);
```

The SAS_GKSQCIX routine returns the predefined SAS color name associated with a color index. SAS_GKSQCIX searches for the current color name assigned to a color index in the following order:

```
* the color set by the SAS_GKSSCIX function.
* the COLORS= graphics option. If color-index is 2, the function
returns the second color from the colors list of the COLORS=
graphics option.
* the device's default colors list found in the device entry.  If
color-index is 2, the routine returns the second color from the
default colors list.
```

Argument Definitions

Variable	Type	Description
color-index	int	indicates the color index for which you want to check the color. Valid values are 1 to 256, inclusive.
color	char8	returns the color name associated with color-index, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSQCLP

Finds whether clipping is on or off

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 55, 56

Syntax

```
status = SAS_GKSQCLP();
```

The SAS_GKSQCLP function returns a code telling whether clipping outside viewports is enabled or disabled. The default is disabled.

Argument Definitions

Variable	Type	Description
status	int	returns the current setting. Use the #define values for the clip status in the file uwproc.h or use one of the following values from the table:

```
           Integer
Value     Equivalent      Meaning
ECLPON        55      clipping enabled
ECLPOFF       56      clipping disabled
```

SAS_GKSQCNX

Finds the color indices that have colors associated with them

Operating States: SGOP

Return Codes: 0, 4, 86, 87

Syntax

```
rc = SAS_GKSQCNX(listn, listidx);
```

The SAS_GKSQCNX function returns the number of and the values of the color indices that currently have colors assigned to them.

Argument Definitions

Variable	Type	Description
listn	ptr to an int	must be initialized to the number of indices you want returned. If the number of assigned color indices is less than the number you requested, the value of listn is changed to the number of assigned color indices.
listidx	ptr to an int	points to an area of memory you have allocated that is formatted in an array of ints. The dimension of the array is determined by the number of color indices you want returned, and the number of indices actually returned is determined by the returned value of listn.
rc	int	returns the return code of the function call.

SAS_GKSQCPL

Finds the color index of the color to be used to draw lines

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQCPL(color-index);
```

The SAS_GKSQCPL function returns the current line color. If a SAS_GKSSCPL function has not been previously submitted, SAS_GKSQCPL returns the default value, 1. The color index returned corresponds to a color specification in the following order:

```
* the color specified in a SAS_GKSSCPL function
* the nth color in the colors list of the COLORS= graphics option
* the nth color in the device's default colors list.
```

Argument Definitions

Variable	Type	Description
color-index	ptr to an int	returns the color index of the current line color.
rc	int	returns the return code of the function call.

SAS_GKSQCPM

Finds the color index of the color to be used to draw markers

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQCPM(color-index);
```

The SAS_GKSQCPM function returns the current marker color.If a SAS_GKSSCPM function has not been previously submitted, SAS_GKSQCPM returns the default value, 1. The color index returned corresponds to a color specification in the following order:

```
* the color selected in a SAS_GKSSCIX function
* the nth color in the colors list of the COLORS= graphics option
* the nth color in the device's default colors list.
```

Argument Definitions

Variable	Type	Description
color-index	ptr to an int	returns the color index of current marker color.
rc	int	returns the return code of the function call.

SAS_GKSQCTX

Finds the color index of the color currently selected to draw text strings

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQCTX(color-index);
```

The SAS_GKSQCTX function returns the current text color. If a SAS_GKSSCTX function has not been previously submitted, SAS_GKSQCTX returns the default value, 1. The color index returned corresponds to a color specification in the following order:

```
* the color specified in a SAS_GKSSCIX function
* the nth color in the colors list of the COLORS= graphics option
 the nth color in the device's default colors list.
```

Argument Definitions

Variable	Type	Description
color-index	ptr to an int	returns the color index of the color used to draw text.
rc	int	returns the return code of the function call.

SAS_GKSQDEV

Finds the output graphics device

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSQDEV(device);
```

The SAS_GKSQDEV function returns the current device driver. This function return the device driver set by one of the following methods:

```
* the SAS_GKSSDEV function
* the DEVICE= graphics option
* the device driver you entered in the DEVICE prompt window
* the device driver you entered in the OPTIONS window.
```

There is no default value for a device driver unless the GWINDOW option is active and no GOPTIONS NODISPLAY is specified. To use the graphics user-written procedure toolkit, you must specify a device driver.

Argument Definitions

Variable	Type	Description
device	char8	returns the name of the device driver, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSQETX

Finds the text extent rectangle and concatenation point for a specified text string

Operating States: SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQETX(x, y, len, string, x-end,y-end, quad);
```

The SAS_GKSQETX function returns the text extent rectangle and text concatenation point for a specified text string. All text extent coordinates returned are in units of the current window system. If no text string is specified, SAS_GKSQETX does not return values for the other arguments.

The text attributes and bundles affect the values returned by this query.

See Figure for a diagram of the text extent rectangle.

```
(x3,y3) +-------------------------------------+ (x2,y2)
        |                                     |
        |                                     |
(x,y)   +------center line is baseline---------+ (xend,yend)
        |                                     |
(x0,y0) +-------------------------------------+ (x1,y1)
```

Argument Definitions

Variable	Type	Description
x	ptr to a double	coordinates are in units based on the current window system; returns x coordinate after justification. The variable used to specify x must be initialized.
y	ptr to a double	coordinates are in units based on the current window system; returns y coordinate after justification. The variable used to specify y must be initialized.
len	int	length of text string in characters
string	ptr	a set of characters for which the text extent rectangle and text concatenation point are calculated. Memory for this variable should be allocated by the caller.
x-end	ptr to a double	returns the x coordinate of the point at which the next text string may be concatenated.
y-end	ptr to a double	returns the y coordinate of the point at which the next text string may be concatenated.
quad	ptr to struct	GPAIRF (which is #defined in uwproc.h.h) returns the x,y coordinate pairs of the text extent rectangle. The caller should allocate 4 instances of struct GPAIRF either by memory allocation or by declaring an automatic variable: struct GPAIRF quad[4];

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSQFTX

Finds the font used to draw text strings

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQFTX(font);
```

The SAS_GKSQFTX function returns the current text font. SAS_GKSQFTX searches for the current font in the following order:

```
* the value selected in the SAS_GKSSFTX function, if specified
* the value of the FTEXT= graphics option, if specified
* the device's default hardware font if the device supports a hardware font
* the SIMULATE font.
```

Argument Definitions

Variable	Type	Description
font	char8	returns the font name, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSQHPO

Finds the number of columns

Operating States: All

Return Codes: 0

SAS_GKSQHPO *continued*

Syntax

```
rc = SAS_GKSQHPO(hpos);
```

The SAS_GKSQHPO function returns the number of columns currently in the graphics output area. SAS_GKSQHPO searches for the current number of columns in the following order:

```
* the value selected in the SAS_GKSSHPO function
* the value of the HPOS= graphics option
* the device's default HPOS value found in the device entry.
```

Argument Definitions

Variable	Type	Description
hpos	ptr to an int	returns the number of columns in the graphics output area.
rc	int	returns the return code of the function call.

SAS_GKSQHSI

Finds the horizontal dimension of the graphics output area

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSQHSI(hsize);
```

The SAS_GKSQHSI function returns the current horizontal dimension, in inches, of the graphics output area. SAS_GKSQHSI searches for the current horizontal dimension in the following order:

```
* the value selected in the SAS_GKSSHSI function
* the value of the HSIZE= graphics options
* the device's default HSIZE found in the device entry
```

Argument Definitions

Variable	Type	Description
hsize	ptr to a double	returns the size of the graphics output area in the x dimension (in inches)
rc	int	returns the return code of the function call.

SAS_GKSQHTX

Finds the character height of the text strings

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQHTX(height);
```

The SAS_GKSQHTX function returns the current text height. SAS_GKSQHTX searches for the current text height in the following order:

- □ the value selected in the SAS_GKSSHTX function, if specified
- □ the value of the HTEXT= graphics option, if specified
- □ the default text height, 1.

Argument Definitions

Variable	Type	Description
height	ptr to a double	returns the character height in units of the current window system.
rc	int	returns the return code of the function call.

SAS_GKSQIFA

Finds the type of the interior of the fill area

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

SAS_GKSQIFA *continued*

Syntax

```
rc = SAS_GKSQIFA(interior);
```

The SAS_GKSQIFA function returns the current fill type. If no fill type has been previously selected with the SAS_GKSSIFA function, SAS_GKSQIFA returns the default value, HOLLOW.

Argument Definitions

Variable	Type	Description
interior	char8	returns the fill type that is active and is one of the following values, in upper case, padded with blanks: * HATCH * HOLLOW * PATTERN * SOLID
rc	int	returns the return code of the function call.

SAS_GKSQMDS

Finds the maximum display area size

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQMDS(units, x-dim, y-dim,x-pixels,y-pixels);
```

The SAS_GKSQMDS function returns the dimensions of the maximum display area for the device. This routine is useful when you need to know the maximum display area in order to figure out how to scale a graph.

There is a difference between the maximum display size returned when the operating state is not SGOP and when it is SGOP. The full addressable display area is returned when the operating state is not SGOP, and the display area minus room for titles and footnotes is returned when the operating state is SGOP.

Argument Definitions

Variable	Type	Description
units	ptr to an int	returns a 1 to show that x-dim and y-dim are in meters.
x-dim	ptr to a double	returns the dimension, in meters, in the x direction.
y-dim	ptr to a double	returns the dimension, in meters, in the y direction.
x-pixels	ptr to an int	returns the number of pixels in the x direction.
y-pixels	ptr to an int	returns the number of pixels in the y direction.
rc	int	returns the return code of the function call.

SAS_GKSQNSG

Finds the number of graphs in the current catalog

Operating States: SGOP, WSAC, WSOP

Return Codes: 0, 7

Syntax

```
rc = SAS_GKSQNSG(n);
```

The SAS_GKSQNSG function returns how many graphs are in the current catalog. The catalog checked is the catalog selected in the SAS_GKSSCAT function, if specified; otherwise, it is the default catalog, WORK.GSEG.

Argument Definitions

Variable	Type	Description
n	ptr to an int	returns the number of graphs in the current catalog.
rc	int	returns the return code of the function call.

SAS_GKSQNNT

Finds the number of the transformation to be used

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQNNT(n);
```

The SAS_GKSQNNT function returns the current transformation. If a transformation has not been previously selected with the SAS_GKSSLNT function, SAS_GKSQNNT returns the number of the default transformation, 0.

Argument Definitions

Variable	Type	Description
n	ptr to an int	returns the number of the current transformation
rc	int	returns the return code of the function call.

SAS_GKSQOSG

Finds the name of the segment currently open

Operating States: SGOP

Return Codes: 0, 4

Syntax

```
rc = SAS_GKSQOSG(segment);
```

The SAS_GKSQOSG function returns the name of the graph currently open. The name returned is some form of GKS: for example, GKS, GKS1, and GKS2.

Argument Definitions

Variable	Type	Description
name	char8	returns the name of the graph that is currently open, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSQOST

Finds the current operating state of the graphics toolkit

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSQOST(state);
```

The SAS_GKSQOST function returns the current operating state of the Graphics User-Written Procedure Toolkit.

Argument Definitions

Variable	Type	Description
state	char8	return one of the following values, in upper case, padded with blanks: * GKCL * GKOP * SGOP * WSAC * WSOP
rc	int	returns the return code of the function call.

SAS_GKSQOWK

Finds whether the interface is open

Operating States: All

Return Codes: 24, 25

Syntax

```
status = SAS_GKSQOWK();
```

The SAS_GKSQOWK function asks if the workstation is open. If a workstation is open, the graphics catalog can be accessed.

SAS_GKSQOWK *continued*

Argument Definitions

Variable	Type	Description

status	int	returns the status of the workstation Use the #define values for the workstation status in the file uwproc.h or use one of the following values from the table:
		`Value / Integer Equivalent / Meaning`
		`EWSISOPN  24  workstation open`
		`EWSNOTOP  25  workstation closed`

SAS_GKSQPIX

show pattern names assigned to a style index.

Operating States: GKOP, WSOP, WSAC, or SGOP

Return Codes: 0, 8, 79

Syntax

```
rc = SAS_GKSQPIX(styindex, patname, hatchname);
```

This shows which pattern names are assigned to a style index.

Argument Definitions

Variable	Type	Description
styindex	int	must be initialized to a value in the range of 1 to 100.
patname	char8	returns the name of the pattern used when the interior style is PATTERN.
hatchname	char8	returns the name of the pattern used when the interior style is HATCH.
rc	int	returns the return code of the function call.

SAS_GKSQPRI

Finds the current priority

Operating States: SGOP

Return Codes: 0, 4

Syntax

```
rc = SAS_GKSQPRI(priority);
```

The SAS_GKSQPRI function returns the current priority (drawing order). If the priority has not been previously selected with the SAS_GKSSPRI function, SAS_GKSQPRI returns the default value, 4.

Argument Definitions

Variable	Type	Description
priority	int	returns a priority value from 0 to 7; 0 priority primitives are drawn first and 7 priority primitives are drawn last.
rc	int	returns the return code of the function call.

SAS_GKSQPTX

Finds the direction of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQPTX(path);
```

The SAS_GKSQPTX function returns the current text path (reading direction). If the text path has not been previously selected with the SAS_GKSSPTX function, SAS_GKSQPTX returns the default value, RIGHT. See the SAS_GKSSPTX function for an illustration of text paths.

SAS_GKSQPTX *continued*

Argument Definitions

Variable	Type	Description

path	char8	returns one of the following values,in upper case, padded with blanks:
		`* DOWN`
		`* LEFT`
		`* RIGHT`
		`* UP`

rc	int	returns the return code of the function call.

SAS_GKSQSEG

Finds the names of segments in the current catalog

Operating States: SGOP, WSAC, WSOP

Return Codes: 0, 7

Syntax

```
rc = SAS_GKSQSEG(n, name-array);
```

The SAS_GKSQSEG function lists the first n names of the graphs that are in the current catalog. If a catalog has not been previously specified with the SAS_GKSSCAT function, the routine returns names from the default catalog, WORK.GSEG.

The names returned are any of the following:

```
* some form of GKS: for example, GKS, GKS1, or GKS2.
* the name specified in the NAME= option of a graphics procedure
* graphs previously created by other graphics procedures and already in the catalog.
```

Argument Definitions

Variable	Type	Description
n	ptr to an int	must be initialized to the number of graph names you want returned. If the number of graph in the catalog is less than the number you requested, the value of n is changed to the number of graphs in the catalog.
name-array	ptr to a char8	points to an area of memory you have allocated that is formatted in an array of char8s. The dimension of the array is determined by the number of graph names you want returned, and the number of graph names actually returned is determined by the returned value of n. The names are in upper case and padded with blanks.

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSQSFA

Finds the style of the fill area when FILTYPE is PATTERN or HATCH

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQSFA(style-index);
```

The SAS_GKSQSFA function returns the current fill style of the interior when the interior style is PATTERN or HATCH. If no interior style has been previously selected with the SAS_GKSSSFA function, SAS_GKSQSFA returns the default value, 1

Argument Definitions

Variable	Type	Description
style-index	ptr to an int	returns the index of the fill pattern associated with the bundle. See the SAS_GKSSSFA function later in this chapter for the interior styles represented by style-index.
rc	int	returns the return code of the function call.

SAS_GKSQSPM

Finds the size of markers

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQSPM(size);
```

The SAS_GKSQSPM function returns the current marker size. If no marker size has been previously selected with the SAS_GKSSSPM function, SAS_GKSQSPM returns the default value, 1.

SAS_GKSQSPM *continued*

Argument Definitions

Variable	Type	Description
size	ptr to a double	returns the marker size in units of the current window system.
rc	int	returns the return code of the function call.

SAS_GKSQTPL

Finds the line type

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQTPL(type);
```

The SAS_GKSQTPL function returns the current line type. If no line type was previously selected with the SAS_GKSSTPL function, SAS_GKSQTPL returns the default value, 1.

Argument Definitions

Variable	Type	Description
type	ptr to an int	returns the index of the line type currently selected. See the SAS_GKSSTPL function for an explanation of the indices for lines.
rc	int	returns the return code of the function call.

SAS_GKSQTPM

Finds the kind of markers

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQTPM(type);
```

The SAS_GKSQTPM function returns the current marker type. If no marker type has been previously selected with the SAS_GKSSTPM function, SAS_GKSQTPM returns the default value, 1.

Argument Definitions

Variable	Type	Description
type	ptr to an int	returns the index of the marker type currently selected. See the SAS_GKSSTPM function for an explanation of the indices for markers.
rc	int	returns the return code of the function call.

SAS_GKSQUTX

Finds the orientation (angle) of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQUTX(up-x, up-y);
```

The SAS_GKSQUTX function returns the character up vector values. If the character up vector has not been previously selected with the SAS_GKSSUTX function, SAS_GKSQUTX returns the default values for up-x and up-y, 0 and 1. See the SAS_GKSSUTX function for an explanation of the vector values.

Argument Definitions

Variable	Type	Description
up-x	ptr to a double	returns the x component of the vector.
up-y	ptr to a double	returns the y component of the vector.
rc	int	returns the return code of the function call.

SAS_GKSQVNT

Finds coordinates of the viewport associated with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50

Syntax

```
rc = SAS_GKSQVNT(n, viewport);
```

The SAS_GKSQVNT function returns the coordinates of the viewport associated with the specified transformation. If a viewport has not been defined with the SAS_GKSSVNT function for the specified transformation, SAS_GKSQVNT returns the default coordinates for the viewport, (0,0) and (1,1).

Argument Definitions

Variable	Type	Description
n	int	indicates the transformation number assigned to the viewport to check. Valid values are 0 to 20, inclusive.
viewport	ptr to struct GWINDOW	returns the coordinates to the lower-left and upper-right corners of the viewport.
rc	int	returns the return code of the function call.

SAS_GKSQVPO

Finds the number of rows

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSQVPO(vpos);
```

The SAS_GKSQVPO function returns the current number of rows in the graphics output SAS_GKSQVPO searches for the current number of rows in the following order:

```
* the value selected in the SAS_GKSSVPO function
* the value of the VPOS= graphics option
* the device's default VPOS value found in the device entry
```

Argument Definitions

Variable	Type	Description
vpos	ptr to an int	returns the number of rows in the graphics output
rc	int	returns the return code of the function call.

SAS_GKSQVSI

Finds the vertical dimension of the graphics output area

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSQVSI(vsize);
```

The SAS_GKSQVSI function returns the current vertical dimension, in inches, of the graphics output area. SAS_GKSQVSI searches for the current vertical dimension in the following order:

```
* the value selected in the SAS_GKSSVSI function
* the value of the VSIZE= graphics option
* the device's default VSIZE found in the device entry.
```

Argument Definitions

Variable	Type	Description
vsize	ptr to a double	returns the size of the graphics output area in the y dimension (in inches).
rc	int	returns the return code of the function call.

SAS_GKSQWNT

Finds the coordinates of the window associated with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50

SAS_GKSQWNT *continued*

Syntax

```
rc = SAS_GKSQWNT(n, window);
```

The SAS_GKSQWNT function returns the coordinates of the window associated with the specified transformation number. If no window has been defined with the SAS_GKSSWNT function for transformation n, SAS_GKSQWNT returns the default window coordinates, which are device dependent.

Argument Definitions

Variable	**Type**	**Description**
n	int	indicates the transformation number assigned to the window to check. Valid values are 0 to 20, inclusive.
window	ptr to struct GWINDOW	returns the coordinates to the lower-left and upper-right corners of the window.
rc	int	returns the return code of the function call.

SAS_GKSQWPL

Finds the line thickness

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQWPL(width);
```

The SAS_GKSQWPL function returns the current line width (in pixels). If a line width has not been previously selected with the SAS_GKSSWPL function, SAS_GKSQWPL returns the default value, 1.

Argument Definitions

Variable	**Type**	**Description**
width	ptr to an int	returns the current line width (in units of pixels).
rc	int	returns the return code of the function call.

SAS_GKSQXFA

Finds the bundle of fill area attributes that is active

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQXFA(index);
```

The SAS_GKSQXFA function asks which fill bundle is active.If no fill bundles have been previously defined with a SAS_GKSSBFA function or activated with a SAS_GKSSXFA function, SAS_GKSQXFA returns the default value, 1.

Argument Definitions

Variable	Type	Description
index	ptr to an int	returns the index of the fill currently selected.
rc	int	returns the return code of the function call.

SAS_GKSQXPL

Finds the index of the bundle of line attributes

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQXPL(index);
```

The SAS_GKSQXPL function returns the current line bundle. If no line bundles have been previously defined with SAS_GKSSBPL or activatedwith SAS_GKSSXPL, SAS_GKSQXPL returns the default value, 1.

Argument Definitions

Variable	Type	Description
index	ptr to an int	returns the index of the current line bundle.
rc	int	returns the return code of the function call.

SAS_GKSQXPM

Finds the index of the bundle of marker attributes currently selected

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQXPM(index);
```

The SAS_GKSQXPM function returns the current marker bundle. If no marker bundles have been previously defined with SAS_GKSSBPM or activated with SAS_GKSSXPM, SAS_GKSQXPM returns the default value, 1.

Argument Definitions

Variable	Type	Description
index	ptr to an int	returns the index of the marker bundle currently selected.
rc	int	returns the return code of the function call.

SAS_GKSQXTX

Finds the index of the bundle of text attributes currently selected

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

```
rc = SAS_GKSQXTX(index);
```

The SAS_GKSQXTX function returns the current text bundle. If no text bundles have been previously defined with SAS_GKSSBTX or activated with SAS_GKSSXTX, SAS_GKSQXTX returns the default value, 1.

Argument Definitions

Variable	Type	Description
index	ptr to an int	returns the text bundle index.
rc	int	returns the return code of the function call.

Drawing Functions

Drawing functions create graphics elements. Each drawing operator is associated with a set of setting operators that control its attributes. For example, the color, height, and font for the SAS_GKSDRTX function are controlled by SAS_GKSSCTX, SAS_GKSSHTX, and SAS_GKSSFTX, respectively. The complete graph is displayed after the SAS_GKSUPWK function is submitted.

SAS_GKSDRAR

Draws a circular arc

Operating States: SGOP

Return Codes: 0, 4, 61, 86

Syntax

```
rc = SAS_GKSDRAR(x, y, radius, start, end);
```

The SAS_GKSDRAR function draws a circular arc. The line attributes and bundles affect the appearance of this primitive.

Argument Definitions

Variable	Type	Description
x	double	specifies the x coordinate of the position of the arc; the x coordinates are in units based on the current window system.
y	double	specifies the y coordinate of the position of the arc; the y coordinates are in units based on the current window system.
radius	double	the arc radius size is in units based on the current window system.
start	double	the starting angle of the arc is in degrees, with 0 degrees at 3 o'clock.
end	double	the ending angle of the arc is in degrees, with 0 degrees at 3 o'clock.
rc	int	returns the return code of the function call.

SAS_GKSDRBA

Draws a rectangle

Operating States: SGOP

Return Codes: 0, 4, 76, 79, 80, 86

Syntax

```
rc = SAS_GKSDRBA(x1, y1, x2, y2);
```

The SAS_GKSDRBA function draws a rectangular bar whose sides are parallel to the sides of the display area. The fill attributes and bundles affect the appearance of this graphics element.

Argument Definitions

Variable	Type	Description
x1	double	refers to the x coordinate of one corner of the bar; x coordinates are in units based on the current window system.
y1	double	refers to the y coordinate of one corner of the bar; y coordinates are in units based on the current window system.
x2	double	refers to the x coordinate of the corner of the bar that is diagonally opposite to the corner of x1; x coordinates are in units based on the current window system.
y2	double	refers to the x coordinate of the corner of the bar that is diagonally opposite to the corner of y1; y coordinates are in units based on the current window system.
rc	int	returns the return code of the function call.

SAS_GKSDREA

Draws an elliptical arc

Operating States: SGOP

Return Codes: 0, 4, 61, 86

Syntax

```
rc = SAS_GKSDREA(x, y, major, minor, start,end, angle);
```

The SAS_GKSDREA function draws a hollow section of an ellipse. The line attributes and bundles affect the appearance of this primitive.

Argument Definitions

Variable	Type	Description
x	double	x coordinate of the position of the elliptical arc; x coordinates are in units based on the current window system.
y	double	y coordinate of the position of the elliptical arc; y coordinates are in units based on the current window system.
major	double	the major axis length for the elliptical arc; length is in units based on the current window system.
minor	double	the minor axis length for the elliptical arc; length is in units based on the current window system.
start	double	the starting angle from the major axis, in degrees, for the elliptical arc, with 0 degrees beginning at 3 o'clock.
end	double	the ending angle from the major axis, in degrees, for the elliptical arc, with 0 degrees beginning at 3 o'clock.
angle	double	the angle that the major axis of the elliptical arc has to 0 degrees (with 0 degrees at 3 o'clock).
rc	int	returns the return code of the function call.

SAS_GKSDREL

Draws an ellipse

Operating States: SGOP

Return Codes: 0, 4, 76, 79, 80, 86

Syntax

```
rc = SAS_GKSDREL(x, y, major, minor, start,end, angle);
```

The SAS_GKSDREL function draws a filled section of an ellipse. The fill attributes and bundles affect the appearance of this primitive.

SAS_GKSDREL *continued*

Argument Definitions

Variable	Type	Description
x	double	the x coordinate of the position of the ellipse; x coordinates are in units based on the current window system.
y	double	the y coordinate of the position of the ellipse; y coordinates are in units based on the current window system.
major	double	the major axis length for the ellipse; length is in units based on the current window system.
minor	double	the minor axis length for the ellipse; length is in units based on the current window system.
start	double	the starting angle for the ellipse from the major axis, with 0 degrees beginning at the major axis.
end	double	the ending angle for the ellipse from the major axis, with 0 degrees beginning at the major axis.
angle	double	the angle that the major axis of the ellipse has to 0 degrees, with 0 degrees at 3 o'clock.
rc	int	returns the return code of the function call.

SAS_GKSDRFA

Draws a filled area

Operating States: SGOP

Return Codes: 0, 4, 76, 79, 80, 86, 100, 301

Syntax

```
rc = SAS_GKSDRFA(n, vertices, missing);
```

The SAS_GKSDRFA function draws a filled polygon. The fill attributes and bundles affect the appearance of this primitive.

Argument Definitions

Variable	Type	Description
n	int	the number of vertices in the polygon.
vertices	ptr to struct GPAIRF	list of coordinates for the vertices in units basedon the current window system. The caller of this routine must allocate the memory pointed to by this pointer.

Variable	Type	Description
missing	int	if non-zero, missing values are in the list of vertices to denote multiple boundary polygons (polygons with holes).
rc	int	returns the return code of the function call.

SAS_GKSDRMS

Prints a message in the SAS log

Operating States: All

Return Codes: 0

Syntax

```
rc = SAS_GKSDRMS(length, message);
```

The SAS_GKSDRMS function prints a message in the SAS log. This function may be used for debugging applications or for printing custom messages for your application.

Argument Definitions

Variable	Type	Description
length	int	length of message pointed to by the message variable
message	ptr	ptr to character string containing message to be printed to the log. The memory pointed to by message must be allocated by the caller.
rc	int	returns the return code of the function call.

SAS_GKSDRPI

Draws a filled circle or section of a filled circle

Operating States: SGOP

Return Codes: 0, 4, 76, 79, 80, 86

SAS_GKSDRPI *continued*

Syntax

```
rc = SAS_GKSDRPI(x, y, radius, start, end);
```

The SAS_GKSDRPI function draws a filled section of a circular arc. The fill attributes and bundles affect the appearance of this primitive.

Argument Definitions

Variable	Type	Description
x	double	the x coordinate of the position of the circle in units based on the current window system.
y	double	the y coordinate of the position of the circle units based on the current window system.
radius	double	the radius length for the circle; length is units based on the current window system.
start	double	the starting angle for the circle, with 0 degrees at 3 o'clock.
end	double	the ending angle for the circle, with 0 degrees at 3 o'clock.
rc	int	returns the return code of the function call.

SAS_GKSDRPL

Draws a polyline

Operating States: SGOP

Return Codes: 0, 4, 61, 86, 100, 301

Syntax

```
rc = SAS_GKSDRPL(n, vertices);
```

The SAS_GKSDRPL function draws one line, a series of connected lines, or a dot. The line attributes and bundles affect the appearance of this primitive.

Argument Definitions

Variable	Type	Description
n	int	the number of vertices in the polyline.
vertices	ptr to struct GPAIRF	list of coordinates for the vertices in units based on the current window system. The caller of this routine must allocate the memory pointed to by this pointer.

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSDRPM

Draws a polymarker

Operating States: SGOP

Return Codes: 0, 4, 65, 86, 100, 301

Syntax

```
rc = SAS_GKSDRPM(n, vertices);
```

The SAS_GKSDRPM function draws a series of symbols. The marker attributes and bundles affect the appearance of this primitive.

Argument Definitions

Variable	Type	Description
n	int	the number of vertices in the polymarker.
vertices	ptr to struct GPAIRF	list of coordinates for the vertices in units based on the current window system. The caller of this routine must allocate the memory pointed to by this pointer.
rc	int	returns the return code of the function call.

SAS_GKSDRTX

Draws a text string

Operating States: SGOP

Return Codes: 0, 4, 69, 86

Syntax

```
rc = SAS_GKSDRTX(x, y, len, string);
```

The SAS_GKSDRTX function draws a text string. The text attributes and bundles affect the appearance of this primitive. The justification point of the text string is dependent on text path and text alignment.

SAS_GKSDRTX *continued*

Argument Definitions

Variable	Type	Description
x	double	x coordinate of justification point of the text string; x coordinates are in units based on the current window system.
y	double	y coordinate of justification point of the text string; y coordinates are in units based on the current window system.
len	int	length of text string in characters.
string	ptr	the text string. Memory for this variable should be allocated by the caller.
rc	int	returns the return code of the function call.

Graph Management Functions

Graph management functions perform library management tasks from within the graphics toolkit. These functions can only be performed on one catalog at a time. They cannot be performed across catalogs. For example, you cannot copy a graph from one catalog to another.

SAS_GKSCSEG

Copies a graph

Operating States: WSOP, WSAC, SGOP

Return Codes: 0, 7, 307

Syntax

```
rc = SAS_GKSCSEG(name, new-name);
```

The SAS_GKSCSEG function copies a graph to another catalog entry. The graph to be copied must be closed and be in the current catalog. You cannot copy from one catalog to another. The new graph will also be in the current catalog.

Argument Definitions

Variable	Type	Description
name	char8	name of the graph to be copied, in upper case, padded with blanks.
new-name	char8	name of the graph to be created,in upper case, padded with blanks.

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSDSEG

Deletes a graph

Operating States: SGOP, WSAC, WSOP

Return Codes: 0, 4, 7, 307

Syntax

```
rc = SAS_GKSDSEG(name);
```

The SAS_GKSDSEG function deletes a graph in the current catalog. The graph does not have to be closed to be deleted.

Argument Definitions

Variable	Type	Description
name	char8	the name of the graph to delete, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSERWK

Opens a graphics segment for output

Operating States: WSAC

Return Codes: 0, 3, 301, 302

Resulting Operating State: SGOP

Syntax

```
rc = SAS_GKSERWK(byline, len);
```

The SAS_GKSERWK function opens a graphics segment for output in the current catalog. The value of byline is displayed in catalog listings and in catalog information in the GREPLAY procedure. This function moves the operating state from WSAC to SGOP.

SAS_GKSERWK *continued*

Argument Definitions

Variable	Type	Description
byline	ptr	a text string used for description of the graph. The memory referenced by byline must be allocated by the caller. If no byline is desired, replace the value with NULL.
len	int	length of byline in characters. If no byline is desired, replace the value with 0.
rc	int	returns the return code of the function call.

SAS_GKSISEG

Inserts a previously created segment into the currently open graph

Operating States: SGOP

Return Codes: 0, 4, 302, 307

Syntax

```
rc = SAS_GKSISEG(name);
```

The SAS_GKSISEG function inserts a graph into the currently open graph. The graph to be inserted must be closed and be in the current catalog.

Argument Definitions

Variable	Type	Description
name	char8	the name of a graph to be inserted, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSNAME

names and describes a graph

Operating States: SGOP

Return Codes: 0, 4

Syntax

```
rc = SAS_GKSNAME(name, descrip);
```

The SAS_GKSNAME function give a name and description a graph to be created.

Argument Definitions

Variable	Type	Description
name	char8	name of the graph to be created
descrip	char40	description of the graph
rc	int	returns the return code of the function call.

SAS_GKSPLAY

displays a graph

Operating States: SGOP, WSAC, WSOP, GKOP

Return Codes: 0, 8, 307

Syntax

```
rc = SAS_GKSPLAY(graph);
```

The SAS_GKSPLAY function displays the specified graph.

Argument Definitions

Variable	Type	Description
graph	char8	name of the graph to be display
rc	int	returns the return code of the function call.

SAS_GKSRSEG

Renames a graph

Operating States: SGOP, WSAC, WSOP

Return Codes: 0, 7, 307

SAS_GKSRSEG *continued*

Syntax

```
rc = SAS_GKSRSEG(name, new-name);
```

The SAS_GKSRSEG function renames a graph. The graph to be renamed must be in the current catalog and be closed.

Argument Definitions

Variable	Type	Description
name	char8	the name of the closed graph that is to be changed, in upper case, padded with blanks.
new-name	char8	the new name for the graph, in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSUPWK

Completes the currently open graph and (optionally) displays it

Operating States: SGOP

Return Codes: 0, 4

Resulting Operating State: WSAC

Syntax

```
rc = SAS_GKSUPWK(show);
```

The SAS_GKSUPWK function closes the graph currently open and displays it. The graphics toolkit operates in buffered mode, so the picture is never displayed until this function is called.

This function can be called only once for the currently open graph. Therefore, you cannot incrementally build a graph; however, you can close the currently open graph and later insert it into another graph. This function moves the operating state from SGOP to WSAC.

Argument Definitions

Variable	Type	Description
show	int	if non-zero, the graph will be shown. If zero, the graph is closed and not displayed.
rc	int	returns the return code of the function call.

Setting Functions

Setting functions allow you to set attributes for the graphics elements. Some setting functions set the attributes for a subset of graphics primitives. For example, fill attributes control the appearance of the graphics primitives bar, ellipse, fill area, and pie.

Some setting functions affect the appearance of the entire graphics output. For example, SAS_GKSSHPO and SAS_GKSSVPO set the number of columns and rows for the output. See each function for the aspect of the graphics output that it controls.

SAS_GKSSASF

Specifies an aspect source flag to bundle or separate attributes

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Default Value: INDIVIDUAL

Syntax

```
rc = SAS_GKSSASF(flag, status);
```

The SAS_GKSSASF function sets an attribute's aspect source flag (ASF) so that it can be used in a bundle (BUNDLED) or individually (INDIVIDUAL). If an attribute's ASF is set to BUNDLED, it cannot be used outside of a bundle. It must be defined in a SAS_GKSSB__ function and activated with a SAS_GKSSX__ function, where __ can have one of the following values: fa, pl, pm, tx. If an attribute's ASF is set to INDIVIDUAL, it cannot be used with a bundle.

Argument Definitions

Variable	Type	Description
flag	int	value between 0 and 10, inclusive, indicative of which aspect source flag is queried. Use the #define values for the aspect source flag in the file uwproc.h or use one of the following values from the table:

Value	Integer Equivalent	Purpose
FILCOLASF	0	fill color ASF
FILSTYASF	1	fill style ASF
FILINTASF	2	interior style ASF
LINCOLASF	3	line color ASF
LINTYPASF	4	line type ASF
LINWIDASF	5	line width ASF
MARCOLASF	6	marker color ASF
MARSIZASF	7	marker size ASF
MARTYPASF	8	marker type ASF
TEXCOLASF	9	text color ASF
TEXFNTASF	10	text font ASF

status	int	the value to set the ASF to. Use the #define values for the ASF flag value in the file uwproc.h or use one of the following values from the table:

SAS_GKSSASF *continued*

Variable	Type	Description
		```
		                 Integer
		  Value    Equivalent        Meaning
		ASFBUND        0      attribute is bundled
		ASFINDIV       1      attribute is individual
		```
rc	int	returns the return code of the function call.

SAS_GKSSASP

Specifies the aspect ratio

Operating States: GKCL

Return Codes: 0, 1, 90, 307

Default Value: 0.0

Syntax

```
rc = SAS_GKSSASP(aspect);
```

The SAS_GKSSASP function sets the aspect ratio used to draw graphics output. SAS_GKSSASP affects only pies and arcs.

Argument Definitions

Variable	Type	Description
aspect	double	specifies the aspect ratio and cannot be less than 0.
rc	int	returns the return code of the function call.

SAS_GKSSATX

Specifies the horizontal and vertical alignment of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Default values: halign=NORMHORIZ, valign=NORMVERT

Syntax

```
rc = SAS_GKSSATX(halign, valign);
```

The SAS_GKSSATX function sets a particular type of horizontal and vertical alignment for text strings.

Argument Definitions

Variable	Type	Description
halign	int	specifies horizontal alignment of text strings. Use the #define values for the horizontal alignment value in the file uwproc.h or use one of the following values from the table:

	Integer	
Value	Equivalent	Meaning
CENTHORIZ	0	text is centered horizontally
LEFTHORIZ	1	text is left-justified
NORMHORIZ	2	text follows normal horizontal alignment; left if the text path is RIGHT, right if the text path is LEFT, and center if the text path is UP or DOWN.
RIGHTHORIZ	3	text is right-justified.

Variable	Type	Description
valign	int	specifies vertical alignment of text strings. Use the #define values for the vertical alignment value in the file uwproc.h or use one of the following values from the table:

	Integer	
Value	Equivalent	Meaning
BASEVERT	0	text aligns to character baseline
BOTVERT	1	text aligns to character bottom
HALFVERT	2	text aligns to 1/2 character heigh
NORMVERT	3	text follows normal vertical alignment; base if text path is LEFT or RIGHT, bottom if text path is UP, top if text path is DOWN.
TOPVERT	4	text aligns to character top.

Variable	Type	Description
rc	int	returns the return code of the function call.

SAS_GKSSBFA

Associates a bundle of fill attributes with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 75, 78, 85

Default Value: none

SAS_GKSSBFA *continued*

Syntax

```
rc = SAS_GKSSBFA(index, color-index, interior,style-index);
```

The SAS_GKSSBFA function assigns a color, type of interior, and style of the interior to a specific fill bundle. The aspect source flags for fill color, fill type, and interior style must be set to bundled before the associated drawing function is executed if you want the bundled values to be used when the affected graphics element is drawn.

Argument Definitions

Variable	Type	Description
index	int	indicates the index to be used with the bundle. Valid values are 1 to 20, inclusive.

color-index	int	indicates the index of the color to be used. Valid values are 1 to 256, inclusive. The color index should represent one of the following:
		`* a color index assigned with the SAS_GKSSCIX function`
		`* the nth color in the colors list of the COLORS= graphics option`
		`* the nth color in the device's default colors list.`

interior	int	indicates the type of interior. Use the #define values for the ASF flag value in the file uwproc.h or use one of the following values from the table:
		`Value / Integer Equivalent / Meaning`
		`FILLHATCH  0  hatch`
		`FILLHOLL  1  empty`
		`FILLPAT  2  pattern`
		`FILLSOL  3  solid`

style-index	int	indicates the index of the style to be used. Valid values are 1 to 100. If interior is HOLLOW or SOLID, style- index is ignored.
rc	int	returns the return code of the function call.

SAS_GKSSBPL

Associates a bundle of line attributes with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 60, 62, 85, 90

Default Value: none

Syntax

```
rc = SAS_GKSSBPL(index, color-index, width, type);
```

The SAS_GKSSBPL function assigns a color, width, and line type to a specific line bundle. The aspect source flags (ASF) for line color, line width, and line type must be set to BUNDLED before the associated drawing function is executed if you want the bundled values to be used when the affected graphics element is drawn.

Argument Definitions

Variable	Type	Description
index	int	indicates the number for the bundle to use as an index. Valid values are 1 and 20, inclusive.

color-index	int	specifies the index of the color to use. Valid values are 1 to 256, inclusive. The color index should represent one of the following:
		`* a color index assigned with the SAS_GKSSCIX function`
		`* the nth color in the colors list of the COLORS= graphics option`
		`* the nth color in the device's default colors list.`

width	int	indicates the width of the line in pixels; must be greater than 0.
type	int	indicates the type of line. Valid values are 1 to 46, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSBPM

Associates a bundle of marker attributes with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 64, 66, 85, 90

Default Value: none

Syntax

```
rc = SAS_GKSQBPM(index, color-index, size, type);
```

The SAS_GKSSBPM function assigns a color, size, and type of marker to a specific marker bundle. The aspect source flag (ASF) of marker color, marker size, and marker type must be set to BUNDLED before the SAS_GKSDRPM function is executed if you want the bundled values to be used when the marker is drawn.

SAS_GKSSBPM *continued*

Argument Definitions

Variable	Type	Description
index	int	defines the bundle index number. Valid values are 1 to 20, inclusive.

color-index	int	indicates the color index of the color to use. Valid values are 1 to 256, inclusive. The color index should represent one of the following:
		`* a color index assigned to a color name with the SAS_GKSSCIX function`
		`* the nth color in the colors list of the COLORS= graphics option`
		`* the nth color in the device's default colors list.`

size	double	indicates the size of the marker in units of the current window system; must be greater than 0.
type	int	specifies the type of marker to use; valid values are 1 to 67, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSBTX

Associates a bundle of text attributes with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 68, 85

Default Value: none

Syntax

```
rc = SAS_GKSSBTX(index, color-index, font);
```

The SAS_GKSSBTX function assigns a color and font to a particular text bundle. The aspect source flags (ASF) of TEXCOLOR and TEXFONT must be set to BUNDLED before the SAS_GKSDRTX function is executed if you want the bundled values to be used when the text is drawn.

Argument Definitions

Variable	Type	Description
index	int	specifies the number to use as an index for the bundle; valid values are 1 to 20, inclusive.
color-index	int	indicates the color to use; valid values are 1 to 256, inclusive. The color index should represent one of the following:

Variable	Type	Description
		`* a color index assigned with the SAS_GKSSCIX function` `* the nth color in the colors list of the COLORS= graphics option` `* the nth color in the device's default colors list.`
font	char8	names the font to use with the bundle; must be in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSSCAT

Specifies the catalog for the graphs

Operating States: GKCL

Return Codes: 0, 1

Default Values: libref = WORK, catalog-name = GSEG

Syntax

```
rc = SAS_GKSSCAT(libref, catalog-name);
```

The SAS_GKSSCAT function makes the specified catalog the current catalog in which to store graphs generated with the graphics toolkit. SAS_GKSSCAT creates the catalog if it does not exist. The value of libref and catalog-name cannot exceed eight characters. The number of characters allowed for a catalog name varies across operating systems; see the SAS companion for your operating system. Libref should have been defined through the LIBNAME statement.

Argument Definitions

Variable	Type	Description
libref	char8	points to the library that contains the catalog; must be in upper case, padded with blanks.
catalog-name	char8	specifies the catalog name to be used; must be in upper case, padded with blanks.
rc	int	returns the return code of the function call

SAS_GKSSCBA

Specifies the background color

Operating States: GKCL

Return Codes: 0, 1

Default Value: 1. CBACK=graphics option, if specified

2. device's default background color.

Syntax

```
rc = SAS_GKSSCBA(cback);
```

The SAS_GKSSCBA function sets the background color. SAS_GKSSCBA has the same effect as the CBACK= graphics option.

Argument Definitions

Variable	Type	Description
cback	char8	specifies the background using any predefined SAS color name; must be in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSSCFA

Specifies the color index of the color used to draw fill areas

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 85

Default Value: 1

Syntax

```
rc = SAS_GKSSCFA(color-index);
```

The SAS_GKSSCFA function selects the color index of the color used to draw fill areas. The aspect source flag (ASF) of FILCOLASF must be set to individual for this attribute to be used outside of a fill bundle. The graphics toolkit searches for a color to assign to the index in the following order:

* the color specified for the index in a SAS_GKSSCIX function
* the nth color in the colors list of the COLORS= graphics option
* the nth color in the device's default colors list found in the device entry.

Argument Definitions

Variable	Type	Description
color-index	int	a number from 1 to 256 that identifies a color.
rc	int	returns the return code of the function call.

SAS_GKSSCIX

Associates a color name with a certain color index

Operating States: SGOP

Return Codes: 0, 4, 86

Default Values: 1. colors list of COLORS= graphics option 2.device’s default colors list

Syntax

```
rc = SAS_GKSSCIX(color-index, color);
```

The SAS_GKSSCIX function associates a predefined SAS color name with a color index. Many other toolkit functions use color-index as an argument. If this function is not used, the graphics toolkit searches for a color specification in the following order:

```
* the nth color in the colors list of the COLORS= graphics option
* the nth color in the device's default colors list.
```

Argument Definitions

Variable	Type	Description
color-index	int	a number from 1 to 256 that identifies a color.
color	char8	a predefined SAS color name; must be in upper case, padded with blanks.
rc	int	returns the return code of the function call.

SAS_GKSSCLP

enables and disables viewport clipping

Operating States: WSOP, WSAC

Return Codes: 0, 6

SAS_GKSSCLP *continued*

Syntax

```
rc = SAS_GKSSCLP(flag);
```

The SAS_GKSSCLP function enables and disables viewport clipping.

Argument Definitions

Variable	Type	Description
flag	int	if set to 0, clipping is disabled. Any non-zero value enables clipping.
rc	int	returns the return code of the function call:

SAS_GKSSCPL

Specifies the color index of the color used to draw lines

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 85

Default Value: 1

Syntax

```
rc = SAS_GKSSCPL(color-index);
```

The SAS_GKSSCPL function selects the index of the color used to draw lines. The aspect source flag (ASF) for line color must be set to INDIVIDUAL for this attribute to be used outside of a line bundle. The graphics toolkit searches for a color specification in the following order:

```
* the color specified for the index in a SAS_GKSSCPL function
* the nth color in the colors list of the COLORS= graphics option
* the nth color in the device's default colors list found in the device entry.
```

Argument Definitions

Variable	Type	Description
color-index	int	indicates the index of the color to use. Valid values are 1 to 256, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSCPM

Specifies the color index of the color used to draw markers

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 85

Default Value: 1

Syntax

```
rc = SAS_GKSSCPM(color-index);
```

The SAS_GKSSCPM function selects the color index of the color used to draw markers. The aspect source flag (ASF) of marker color must be set to INDIVIDUAL for this attribute to be used outside of a marker bundle. The graphics toolkit searches for a color specification in the following order:

```
* the color specified for the index in a SAS_GKSSCIX function
* the nth color in the colors list of the COLORS= graphics option
* the nth color in the device's default colors list found in the
device entry.
```

Argument Definitions

Variable	Type	Description
color-index	int	indicates the index of the color to use. Valid values are 1 to 256, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSCTX

Specifies the color index of the color used to draw text strings

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 85

Default Value: 1

Syntax

```
rc = SAS_GKSSCTX(color-index);
```

The SAS_GKSSCTX function selects the color for text. The aspect source flag (ASF) of text color must be set to INDIVIDUAL for this attribute to be used outside of a text bundle. The graphics toolkit searches for a color specification in the following order:

SAS_GKSSCTX *continued*

```
* the color specified for the index in a SAS_GKSSCIX function
* the nth color from the COLORS= graphics options
* the nth color in the device's default colors list found in
the device entry.
```

Argument Definitions

Variable	Type	Description
color-index	int	indicates the color index of the color to be used. Valid values are 1 to 256, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSDEV

Specifies the output graphics device

Operating States: GKCL

Return Codes: 0, 1

Default Value: 1. DEVICE= graphics option, if specified

2. value entered in DEVICE prompt window

3. value entered in OPTIONS window

Syntax

```
rc = SAS_GKSSDEV(device);
```

The SAS_GKSSDEV function selects the device driver.

Argument Definitions

Variable	Type	Description
device	char8	the name of the driver you will be using; must be in upper case, padded with blanks. Device must match one of the device entries in the SASHELP.DEVICES catalog or one of your personal device catalogs, GDEVICE0.DEVICES through GDEVICE9.DEVICES.
rc	int	returns the return code of the function call.

SAS_GKSSFTX

Specifies the font used to draw text strings

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Default values: 1. FTEXT= graphics option, if specified

2. hardware font, if possible

3. SIMULATE font

Syntax

```
rc = SAS_GKSSFTX(font);
```

The SAS_GKSSFTX function selects a SAS/GRAPH font for the text. The aspect source flag (ASF) for text font must be set to INDIVIDUAL for this attribute to be used outside of a text bundle.

Argument Definitions

Variable	Type	Description
font	char8	the name of a font that can be accessed by SAS/GRAPH software; must be in upper case, padded with blanks. If you want to use the hardware font, use blank for this argument.
rc	int	returns the return code of the function call.

SAS_GKSSHPO

Specifies the number of columns

Operating States: GKCL

Return Codes: 0, 1, 90, 307

Default Value: 1. HPOS= graphics option, if specified

2. device's default HPOS setting

Syntax

```
rc = SAS_GKSSHPO(hpos);
```

The SAS_GKSSHPO function sets the number of columns in the graphics output area. SAS_GKSSHPO has the same effect as the HPOS= graphics option. You can reset the HPOS value by submitting one of the following statements:

SAS_GKSSHPO *continued*

```
* goptions reset=goptions;
* goptions reset=all;
* goptions hpos=0;
```

Argument Definitions

Variable	Type	Description
hpos	int	specifies the number of horizontal columns; must be 0 or greater. If set to 0, the default value will be used.
rc	int	returns the return code of the function call.

SAS_GKSSHSI

Specifies the horizontal dimension of the graphics output area

Operating States: GKCL

Return Codes: 0, 1, 90, 307

Default Value: 1. HSIZE= graphics option, if specified

2. HSIZE device parameter

Syntax

```
rc = SAS_GKSSHSI(hsize);
```

The SAS_GKSSHSI function sets the horizontal dimension, in inches, of the graphics output area. SAS_GKSSHSI affects the dimensions of the default window. You can reset the HSIZE value by submitting one of the following statements:

```
* goptions reset=goptions;
* goptions reset=all;
* goptions hsize=0;
```

Argument Definitions

Variable	Type	Description
hsize	double	specifies the horizontal dimension, in inches, of the graphics output area; must be 0 or greater. If set to 0, the default value will be used.
rc	int	returns the return code of the function call.

SAS_GKSSHTX

Specifies the character height of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 73

Default Value: 1. HTEXT= graphics option, if specified

2. 1 unit

Syntax

```
rc = SAS_GKSSHTX(height);
```

The SAS_GKSSHTX function sets the height for text. SAS_GKSSHTX affects text the same way as the HTEXT= graphics option.

Argument Definitions

Variable	Type	Description
height	double	indicates height in units based on the current window system; must be greater than 0.
rc	int	returns the return code of the function call.

SAS_GKSSIFA

Specifies the type of the interior of the fill area

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 78

Default Value: FILHOLLOW

Syntax

```
rc = SAS_GKSSIFA(interior);
```

The SAS_GKSSIFA function selects a particular type of interior fill. If fill type is set to HATCH or PATTERN, the SAS_GKSSSFA function determines the type of hatch or pattern fill used. The aspect source flag (ASF) for fill type must be set to individual for this attribute to be used outside of a fill bundle.

SAS_GKSSIFA *continued*

Argument Definitions

Variable	Type	Description
interior	int	indicates the type of interior fill. Use the #define values for the interior style value in the file uwproc.h or use one of the following values from the table:
rc	int	returns the return code of the function call.

```
                 Integer
  Value     Equivalent  Meaning
FILLHATCH       0       hatch
FILLHOLL        1       empty
FILLPAT         2       pattern
FILLSOL         3       solid
```

SAS_GKSSLNT

Specifies the number of the transformation to be used

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50

Default Value: 0

Syntax

```
rc = SAS_GKSSLNT(n);
```

The SAS_GKSSLNT function activates the viewport and/or window you have defined for the specified transformation number. If you have not defined both a viewport and window for a transformation, the default is used for the one missing. You can select 0 as the active transformation, but you cannot define a viewport or window for that transformation number. A transformation of 0 activates the default viewport, (0,0) to (1,1), and window, which is device dependent.

Argument Definitions

Variable	Type	Description
n	int	indicates the viewport and/or window to activate; should correspond to the n used in the SAS_GKSSVNT and SAS_GKSSWNT functions. Valid values are 0 to 20, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSPIX

sets pattern names for a particular style type

Operating States: GKOP, WSOP, WSAC, or SGOP

Return Codes: 0, 8, 79

Syntax

```
rc = SAS_GKSSPIX(index, patname, stytype);
```

This routine sets a pattern name to a specified pattern index for a particular style type.

Argument Definitions

Variable	Type	Description
index	int	must be initialized to a value in the range of 1 to 100.
patname	char8	name of the pattern.

stytype	int	indicates which type of interior style inherits the pattern name:
		`Value     Meaning`
		`  0       both PATTERN and HATCH`
		`  1       HATCH only`
		`  2       PATTERN only`

rc	int	returns the return code of the function call.

SAS_GKSSPRI

Specifies the current priority for primitives

Operating States: SGOP

Return Codes: 0, 4

Default Value: 4

Syntax

```
rc = SAS_GKSSPRI(priority);
```

The SAS_GKSSPRI function selects a priority (drawing order) for primitives. The lower values means a primitive is drawn earlier, and higher values means the primitive is drawn later.

SAS_GKSSPRI *continued*

Argument Definitions

Variable	Type	Description
priority	int	specifies the priority. Valid are 0 to 7, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSPTX

Specifies the direction of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Default Value: PATHRIGHT

Syntax

```
rc = SAS_GKSSPTX(path);
```

The SAS_GKSSPTX function selects a particular type of text path. Text path determines the direction in which the text string reads.

Argument Definitions

Variable	Type	Description
path	int	specifies the direction in which the text will read. Use the #define values for the text path value in the file uwproc.h or use one of the following values from the table:
rc	int	returns the return code of the function call.

Value	Integer Equivalent	Meaning
PATHDOWN	0	text reads from up to down
PATHLEFT	1	text reads from right to left
PATHRIGHT	2	text reads from left to right
PATHUP	3	text reads from down to up

SAS_GKSSSFA

Specifies interior fill-area style when interior style is PATTERN or HATCH

Operating State: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 78

Default Value: 1

Syntax

```
rc = SAS_GKSSSFA(style-index);
```

The SAS_GKSSSFA function activates a particular fill pattern when fill interior style is specified as either PATTERN or HATCH. The aspect source flag (ASF) must be set to individual for this attribute to be used outside of a fill bundle.

Argument Definitions

Variable	Type	Description
style-index	int	specifies the fill pattern. Valid values are 1 to 100. When the interior style is PATTERN, you can specify the values 1 to 15. When the interior style is HATCH, you can use the values 1 to 60. Refer to **Table 2.1** for the explanation of these values. The values greater than these ranges, up to 100, can be used to access hardware patterns available on the device.
rc	int	returns the return code of the function call.

SAS_GKSSSFA *continued*

Table 2.1
Style Index Table

Value	PATTERN	HATCH	Value	PATTERN	HATCH
1	X1	M1X	31		M3N045
2	X2	M1X030	32		M3N060
3	X3	M1X045	33		M3N090
4	X4	M1X060	34		M3N120
5	X5	M1N	35		M3N135
6	L1	M1N030	36		M3N150
7	L2	M1N045	37		M4X
8	L3	M1N060	38		M4X030
9	L4	M1N090	39		M4X045
10	L5	M1N120	40		M4X060
11	R1	M1N135	41		M4N
12	R2	M1N150	42		M4N030
13	R3	M2X	43		M4N045
14	R4	M2X030	44		M4N060
15	R5	M2X045	45		M4N090
16		M2X060	46		M4N120
17		M2N	47		M4N135
18		M2N030	48		M4N150
19		M2N045	49		M5X
20		M2N060	50		M5X030
21		M2N090	51		M5X045
22		M2N120	52		M5X060
23		M2N135	53		M5N
24		M2N150	54		M5N030
25		M3X	55		M5N045
26		M3X030	56		M5N060
27		M3X045	57		M5N090
28		M3X060	58		M5N120
29		M3N	59		M5N135
30		M3N030	60		M5N150

SAS_GKSSSPM

Selects the size of markers

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 90

Default Value: 1

Syntax

```
rc = SAS_GKSSSPM(size);
```

The SAS_GKSSSPM function sets the marker size in units of the current window system. The aspect source flag (ASF) of marker size must be set to INDIVIDUAL for this attribute to be used outside of a marker bundle.

Argument Definitions

Variable	Type	Description
size	double	indicates the size of the marker in units of the current window system; must be greater than 0.
rc	int	returns the return code of the function call.

SAS_GKSSTPL

Specifies the line type

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 62

Default Value: 1

Syntax

```
rc = SAS_GKSSTPL(type);
```

The SAS_GKSSTPL function selects a line type. The aspect source flag (ASF) for line type must be set to INDIVIDUAL for this attribute to be used outside of a line bundle.

SAS_GKSSTPL *continued*

Argument Definitions

Variable	Type	Description
type	int	indicates the type of line to use. Valid values are 1 to 46, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSTPM

Selects the kind of markers

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 66

Default Value: 1

Syntax

```
rc = SAS_GKSSTPM(type);
```

The SAS_GKSSTPM function determines the type of marker drawn. The aspect source flag (ASF) of marker type must be set to INDIVIDUAL for this attribute to be used outside of a marker bundle.

Argument Definitions

Variable	Type	Description
type	int	indicates the index of the marker to draw. Valid values are 1 to 67, inclusive.
rc	int	returns the return code of the function call.

Table 2.2
Symbol Indexes Used with the Graphics Toolkit

Index	Symbol	Index	Symbol	Index	Symbol
1	plus	24	K	46	9
2	x	25	L	47	lozenge
3	star	26	M	48	spade
4	square	27	N	49	heart
5	diamond	28	O	50	diamond
6	triangle	29	P	51	club
7	hash	30	Q	52	shamrock
8	Y	31	R	53	fleur-de-lis
9	Z	32	S	54	star
10	paw	33	T	55	sun
11	point	34	U	56	Mercury
12	dot	35	V	57	Venus
13	circle	36	W	58	Earth
14	A	37	0	59	Mars
15	B	38	1	60	Jupiter
16	C	39	2	61	Saturn
17	D	40	3	62	Uranus
18	E	41	4	63	Neptune
19	F	42	5	64	Pluto
20	G	43	6	65	moon
21	H	44	7	66	comet
22	I	45	8	67	asterisk
23	J				

SAS_GKSSUTX

Specifies the orientation (angle) of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 74

Default Values: upx=0, upy=1

Syntax

```
rc = SAS_GKSSUTX(upx, upy);
```

The SAS_GKSSUTX function sets the angle of the text string. The graphics toolkit uses the values of character up vectors to determine the angle of a text string. The character up vector has two components, upx and upy, that describe the angle at which the text string is placed. The angle is calculated with the following formula:

```
angle=atan(-upx/upy)
```

Effectively, when the toolkit is calculating the angle for the text, it uses upx and upy as forces that are pushing the string toward an angle. The natural angle of text in upx direction is toward the 6 o’clock position. In the upy direction, text naturally angles at the 3 o’clock position. If upx is greater than upy, the text is angled toward 6 o’clock. If upy is greater than upx, the text is angled toward 3 o’clock.

Argument Definitions

Variable	Type	Description
upx	double	x component of character up vector. If upy is 0, upx cannot be 0.
upy	double	y component of character up vector. If upx is 0, upy cannot be 0.
rc	int	returns the return code of the function call.

SAS_GKSSVNT

Associates a viewport with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50, 51, 52

Default Values: xleft=0, ybottom=0, xleft=1, yright=1

Syntax

```
rc = SAS_GKSSVNT(n, viewport);
```

The SAS_GKSSVNT function defines a viewport and associates it with the transformation number n. See the SAS_GKSSLNT function for information on how to activate the viewport. See the SAS_GKSSWNT function for information on how to define a window to be used within the viewport.

Argument Definitions

Variable	Type	Description
n	int	specifies the transformation number of the viewport. Valid values are 1 to 20, inclusive.
viewport	ptr to struct GWINDOW	specifies the coordinates to the lower-left and upper-right corners of the viewport. xleft and ybottom cannot be lower than 0, and xright and ytop cannot exceed 1; units are based on percent of the graphics output area.
rc	int	returns the return code of the function call.

SAS_GKSSVPO

Specifies the number of rows

Operating States: GKCL

Return Codes: 0, 1, 90, 307

Default Values: 1. VPOS=graphics option, if specified;

2. device's default VPOS value

Syntax

```
rc = SAS_GKSSVPO(vpos);
```

The SAS_GKSSVPO function sets the number of rows in the graphics output area. SAS_GKSSVPO has the same effect on graphics output as the VPOS= graphics option. You can reset the VPOS value by submitting one of the following statements:

```
* goptions reset=goptions;
* goptions reset=all;
* goptions vpos=0;
```

SAS_GKSSVPO *continued*

Argument Definitions

Variable	Type	Description
vpos	int	specifies the number of rows in the graphics output area; must be 0 or greater. If set to 0, the default value will be used.
rc	int	returns the return code of the function call.

SAS_GKSSVSI

Specifies the vertical dimension of the graphics output area

Operating States: GKCL

Return Codes: 0, 1, 90, 307

Default Values: 1. VSIZE= graphics option, if specified;

2. device's default VSIZE value

Syntax

```
rc = SAS_GKSSVSI(vsize);
```

The SAS_GKSSVSI function sets the vertical dimension, in inches, of the graphics output area. SAS_GKSSVSI affects the dimensions of the default window. You can reset the VSIZE value by submitting one of the following statements:

```
* goptions reset=goptions;
* goptions reset=all;
* goptions vsize=0;
```

Argument Definitions

Variable	Type	Description
vsize	double	indicates the vertical dimension for the graph in inches; must be 0 or greater. If set to 0, the default value will be used.
rc	int	returns the return code of the function call.

SAS_GKSSWNT

Associates a window with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50, 51

Default Values: xleft=0, ybottom=0; xright and ytop are device dependent

Syntax

```
rc = SAS_GKSSWNT(n, window);
```

The SAS_GKSSWNT function defines a window and associates it with a transformation number. See the SAS_GKSSLNT function for information on how to activate a window. See the SAS_GKSSVNT function for information on how to define a viewport for a window.

Argument Definitions

Variable	Type	Description
n	int	specifies the transformation number of the window. Valid values are 1 to 20, inclusive.
window	ptr to struct GWINDOW	specifies the coordinates to the lower-left and upper-right corners of the window.
rc	int	returns the return code of the function call.

SAS_GKSSWPL

Specifies the thickness of the line

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 90

Default Value: 1

Syntax

```
rc = SAS_GKSSWPL(width);
```

The SAS_GKSSWPL function selects a line width in units of pixels. The aspect source flag (ASF) for line width must be set to INDIVIDUAL for this attribute to be used outside of a line bundle.

SAS_GKSSWPL *continued*

Argument Definitions

Variable	Type	Description
width	int	specifies the width of the line in pixels; must be greater than 0.
rc	int	returns the return code of the function call.

SAS_GKSSXFA

Specifies the index of the bundle of fill area attributes

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 75

Default Value: 1

Syntax

```
rc = SAS_GKSSXFA(index);
```

The SAS_GKSSXFA function activates a particular fill bundle. The aspect source flag (ASF) for fill color, fill style, and fill type must be set to bundled before the associated drawing function is executed if you want the bundled values to be used when the affected graphics element is drawn.

Argument Definitions

Variable	Type	Description
index	int	specifies the index number of the fill bundle. Valid values are 1 to 20, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSXPL

Specifies the index of the bundle of line attributes

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 60

Default Value: 1

Syntax

```
rc = SAS_GKSSXPL(index);
```

The SAS_GKSSXPL function activates a particular line bundle. The aspect source flags (ASF) of line color, line type, and line width must be set to BUNDLED before the associated drawing function is executed if you want the bundled values to be used when the affected graphics element is drawn.

Argument Definitions

Variable	Type	Description
index	int	indicates the index of the bundle to activate. Valid values are 1 to 20, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSXPM

Specifies the index of the bundle of marker attributes

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 64

Default Value: 1

Syntax

```
rc = SAS_GKSSXPM(index);
```

The SAS_GKSSXPM function activates the marker bundle indicated by index. The aspect source flag (ASF) for marker color, marker type, and marker size must be set to BUNDLED before the SAS_GKSDRPM function is executed if you want the bundled values to be used when the marker is drawn.

Argument Definitions

Variable	Type	Description
index	int	indicates the number of the bundle to activate. Valid values are 1 to 20, inclusive.
rc	int	returns the return code of the function call.

SAS_GKSSXTX

Specifies the index of the bundle of text attributes

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 68

Default Value: 1

Syntax

```
rc = SAS_GKSSXTX(index);
```

The SAS_GKSSXTX function activates the text bundle indicated by index. The aspect source flag (ASF) for TEXCOLOR and TEXFONT must be set to BUNDLED before the SAS_GKSDRTX function is executed if you want the bundled values to be used when the text is drawn.

Argument Definitions

Variable	Type	Description
index	int	indicates the number of the bundle to activate. Valid values are 1 to 20, inclusive.
rc	int	returns the return code of the function call.

Return Codes for Functions

Return Code	Description
0	Function completed successfully.
1	Should be in GKCL state; the statement is out of place.
2	Should be in GKOP state
3	Should be in WSAC state; the statement is out of place.
4	Should be in SGOP state; the statement is out of place.
7	Should be in WSOP, WSAC, or SGOP state; the statement is out of place.
8	should be in GKOP, WSOP, WSAC, or SGOP state; the statement is out of place.
24	Workstation is open.
25	Workstation is not open.
26	Workstation cannot be opened.
29	Workstation is active.

Return Code	Description
30	Workstation is not active.
50	Invalid transformation number; transformation numbers must be in the range 0 to 20; viewports and windows cannot be defined for transformation 0.
51	Transformation is not a well-defined rectangle; transformations must have coordinates for four vertices.
52	Viewport coordinates are out of range; coordinates must be within dimensions of graphics output area for the device.
55	Clipping is on.
56	Clipping is off.
60	Bad line index; index number must be in the range 1 to 20.
61	No bundle defined for the line index; a SAS_GKSSBPL function has not been submitted for the referenced line index.
62	Line type is less than or equal to 0 or greater than 46; type must be in the range 1 to 46.
64	Invalid marker index; index number must be in the range 1 to 20.
65	No bundle defined for the polymarker index; a SAS_GKSSBPM function has not been submitted for the referenced marker index.
66	Marker type is less than or equal to 0 or greater than 67; type must be in the range 1 to 67.
68	Invalid text index; index numbers must be in the range 1 to 20.
69	No bundle defined for the text index; a SAS_GKSSBTX function has not been submitted for the referenced text index.
73	Character height is less than or equal to 0; height must be greater than 0.
74	Both components of the character up vector are 0; both X and Y of a character up vector cannot be 0.
75	Invalid fill index; index numbers must be in the range 1 to 20.
76	No bundle defined for the fill index; a SAS_GKSSBFA function has not been submitted for the referenced fill index.
78	Style index is less than or equal to 0 or greater than 100; style indexes must be in the range of 1 to 100.
79	Specified pattern index is invalid.
80	Specified hatch style is not supported on this workstation.
85	Color index is less than 1.
86	Invalid color index; color index is out of the range 1 to 256 or is not numeric.
87	No color name defined for the color index.

Return Code	Description
90	Value is less than or equal to 0; value must be greater than 0.
100	Number of points is invalid.
301	Out of memory; your workstation does not have enough memory to generate the graph.
302	Out of room for graph; your device cannot display the size of the graph.
307	Error occurred in program library management; a graph management function did not execute properly.

References

Enderle, G.; Kansy, K.; and Pfaff, G. (1985), *Computer Graphics Programming: GKS The Graphics Standard*, New York: Springer-Verlag New York, Inc.