CHAPTER 10

The Annotate Data Set

Overview

The Annotate facility enables you to generate a special data set of graphics commands from which you can produce graphics output. This data set is referred to as an *Annotate data set*. You can use it to generate custom graphics or to enhance graphics output from many SAS/GRAPH procedures, including GCHART, GCONTOUR, GMAP, GPLOT, GPRINT, GSLIDE, and G3D.

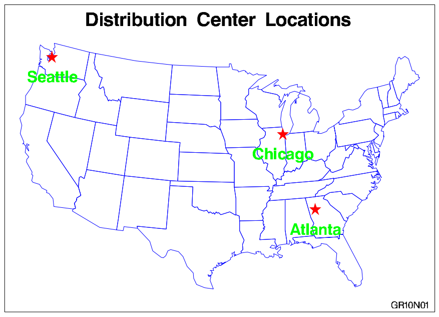
Enhancing Existing Graphs

The Annotate facility enhances output from SAS/GRAPH procedures by adding graphics elements to the output. For example, you can

- □ label points on a map using map coordinates
- □ label bars on horizontal and vertical bar charts
- □ label points on a plot
- □ create a legend for a three-dimensional graph.

Figure 10.1 on page 404 shows GMAP procedure output annotated with stars and labels at selected cities.

Figure 10.1 Annotate Graphics Applied to a Map from the GMAP Procedure (GR10N01)



The program that creates this output is in “Example 1: Labeling Cities on a Map” on page 419.

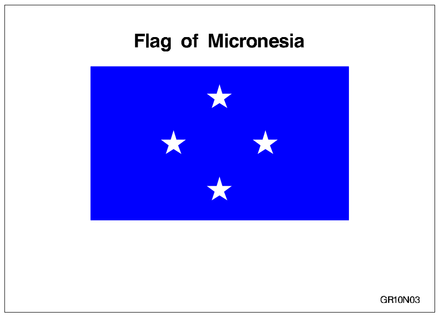
Creating Custom Graphs

You can also use an Annotate data set to create custom graphics. For example, you can use Annotate graphics commands to

- □ create various types of graphs (including pie charts, bar charts, and plots)
- □ draw graphics elements such as lines, polygons, arcs, symbols, and text.

Figure 10.2 on page 404 is an example of a custom graph that uses Annotate commands to draw the graphic elements.

Figure 10.2 Custom Graphics Using Only Annotate Commands (GR10N03)



The program that creates this output is in “Example 3: Drawing a Circle of Stars” on page 424.

Note: The Annotate facility provides many of the same features as the DATA Step Graphics Interface (DSGI). Use the following guidelines to help you decide whether to use DSGI or the Annotate facility when creating custom graphs or enhancing SAS/GRAPH output: △

- Use DSGI for creating custom graphics without using graphics procedures
- Use the Annotate facility for enhancing the output of graphics procedures.

For more information on DSGI, see *Data Step Graphics User's Guide*.

Creating Annotate Graphics

In order to create and use Annotate graphics, you must first understand the structure and functioning of the Annotate data set. For this information see “About the Annotate Data Set” on page 405. Once you understand the way the data set works, you can follow these three steps to create Annotate graphics:

1. Determine what you want to draw, and where (location) and how (coordinate system) you want to position it on the graphics output. (See “About Annotate Graphics” on page 410.)
2. Build an Annotate data set of graphics commands using the Annotate variables and functions. (See “Creating an Annotate Data Set” on page 414.)
3. Submit a SAS/GRAPH procedure to produce the graphics output. (See “Producing Graphics Output from Annotate Data Sets” on page 416.)

About the Annotate Data Set

In an Annotate data set, each observation represents a command to draw a graphics element or to perform an action. The graphic elements drawn by these commands can be added to SAS/GRAPH output or displayed with the GANNO or GSLIDE procedure as a custom graphic.

The observations in an Annotate data set use a set of predefined Annotate variables. The values of the variables in the observation determine what is done and how it is done. To create these observations, you assign values to the variables either explicitly with a DATA step or implicitly with Annotate macros. See “Creating an Annotate Data Set” on page 414.

The following sections describe the items in an Annotate data set and explain how SAS/GRAPH software uses the commands in an Annotate data set to create graphics elements.

Structure of An Annotate Data Set

Output 10.1 on page 406 is an example of an Annotate data set called TRIANGLE. The observations in this data set contain the commands that create a text label, move to a point in the output, and draw a triangle. (The DATA step that creates TRIANGLE is shown in “Using the DATA Step” on page 415.)

Output 10.1 Listing of the Annotate Data Set TRIANGLE

```
OBS   FUNCTION   X    Y   HSYS   XSYS   YSYS    STYLE    COLOR   POSITION  SIZE    LINE    TEXT

  1    label     20   85    3      3      3      swissb   green       6        6.0     .     Sample Annotate Graphics
  2    move      28   30    3      3      3      swissb   green       6        6.0     .     Sample Annotate Graphics
  3    draw      68   30    3      3      3      swissb   red         6        0.8     1     Sample Annotate Graphics
  4    draw      48   70    3      3      3      swissb   red         6        0.8     1     Sample Annotate Graphics
  5    draw      28   30    3      3      3      swissb   red         6        0.8     1     Sample Annotate Graphics
```

Note: A blank denotes a missing values for a character variable. A '.' denotes a missing value for a numeric variable. △

Each observation in this data set contains complete instructions for drawing a graphic or moving to a position to draw a graphic. The value of the FUNCTION variable determines what the observation does. Other variables control how the function is performed. This list describes each observation in the TRIANGLE and the task it performs:

1 Create a label. This instruction draws a blue label at position 20,85 (in X,Y coordinates). The value of the FUNCTION variable (LABEL) tells the program *what* to do. The values of the coordinate variables X and Y combined with the values of the coordinate system variables HSYS, XSYS, and YSYS tell *where* to do it. The values of the attribute variables STYLE, COLOR, TEXT, POSITION, and SIZE tell *how* to do it. These variables specify the font (SWISSB), the color and text of the label, the position of the label in relation to X and Y (centered on the point), and the size of the text.

2 Go to the starting point for the triangle. The value of the FUNCTION variable (MOVE) tells the program to go to the point specified by X and Y. This is the only instruction in the observation. Notice that the values of the variables specified for the first observation persist but are not used because they have no effect on the MOVE function.

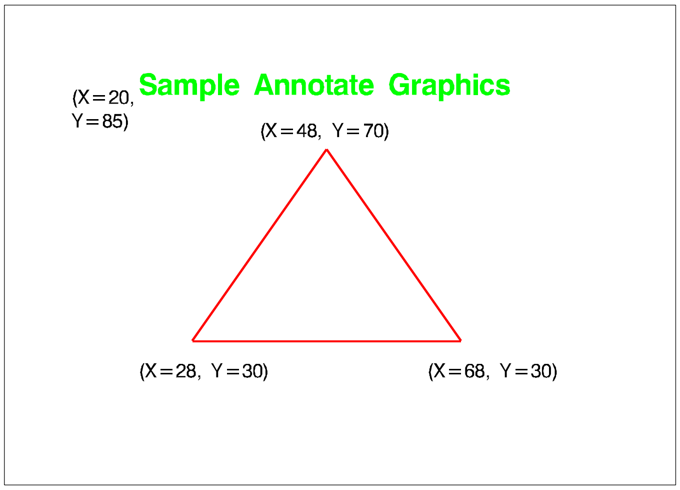
3 Draw the first line of the triangle. The value of the FUNCTION variable (DRAW) tells the program to draw a line from the current point (the one specified by MOVE in the second observation to the new point specified by X and Y. The value of the COLOR variable changes to red.

4 Draw the second line of the triangle.

5 Draw the third line of the triangle.

Figure 10.3 on page 407 shows the green title and the red triangle produced by the TRIANGLE data set and displayed with the GANNO procedure. Notes on the figure in black contain the X and Y coordinates of the graphics elements.

Figure 10.3 Annotate Output from the TRIANGLE Data Set



Annotate Variables

Annotate variables have predefined names. In each observation, the Annotate facility looks only for variables with those names. Other variables can be present, but they are ignored. Conceptually, there are three types of variables:

an action variable	tells *what* to do. The only action variable is FUNCTION, which specifies what graphics element to draw (graphics primitive) or what action to take (programming function).
positioning variables	tell *where* to do it. The positioning variables specify the point at which to draw the graphics element.
attribute variables	tell *how* to do it. The attribute variables specify the characteristics of the graphics element (for example, color, size, line style, text font).

There is also an HTML variabale, which provides linking information when you want to use the annotate data set to generate a drill-down graph that can be viewed in a Web browser.

Table 10.1 on page 407 lists all Annotate variables, grouped by task, and briefly describes each one. See “Annotate Variables” on page 455 for a complete description of each variable.

Table 10.1 Summary of Annotate Variables

Task Group	Variable	Description
Variable that defines an action	FUNCTION	specifies a drawing or programming action; Table 10.2 on page 409 describes these actions.
Positioning variables that determine coordinate values	GROUP	uses the value of the GCHART GROUP= option in place of X or Y
	MIDPOINT	uses the value of the GCHART MIDPOINT= option in place of X or Y

Task Group	Variable	Description
	SUBGROUP	uses the value of the GCHART SUBGROUP= option in place of X or Y
	X	specifies a numeric horizontal coordinate
	Y	specifies a numeric vertical coordinate
	Z	specifies a numeric third dimensional coordinate; used with G3D procedure only
	XC	specifies a horizontal character coordinate; only used with data coordinate systems 1, 2, 7, 8
	YC	specifies a vertical character coordinate; only used with data coordinate systems 1, 2, 7, 8
Positioning variables that contain internal coordinates	XLAST, YLAST	contain the X and Y coordinates of the last nontext function
	XLSTT, YLSTT	contain the X and Y coordinates of the last text function
Positioning variables that specify coordinate systems	HSYS	specifies type of units for the SIZE variable
	XSYS	specifies coordinate system for X or XC coordinates
	YSYS	specifies coordinate system for Y or YC coordinates
	ZSYS	specifies coordinate system for Z coordinate (G3D procedure only)
Attribute variables	ANGLE	angle of text label or starting angle of a pie slice
	CBORDER	colored border around text or symbol
	CBOX	colored box behind text or symbol
	COLOR	color of a graphics primitive
	LINE	line type to use in drawing or special control over pies and bars
	POSITION	placement and alignment for text strings
	ROTATE	angle at which to place individual characters in a text string or the delta angle (sweep) of a pie slice
	SIZE	size of an aspect of a graphics primitive; depends on FUNCTION variable (for TEXT, height of characters; for PIE, pie slice radius; for DRAW, line thickness; and so on)
	STYLE	font or pattern for a graphics element, depends on the FUNCTION variable
	TEXT	text to use in a label, symbol, or comment
	WHEN	whether a graphics element is drawn before or after procedure graphics output
Web variable	HTML	specifies link information for a drill-down graph

Annotate Functions

The FUNCTION variable accepts a set of predefined values (functions) that perform both graphics tasks and programming tasks.

The graphics functions draw the graphics elements that are illustrated in “Graphics Elements” on page 410.

The programming functions control the internal coordinates, manipulate the LIFO stack, and help you debug an Annotate data set. These programming functions are discussed in “Internal Coordinates” on page 413, “Using the LIFO Stack” on page 418, and “Debugging” on page 419.

Table 10.2 on page 409 summarizes the tasks that are performed by the Annotate functions. See “Annotate Functions” on page 429 for a complete description of the FUNCTION variable and its values.

Table 10.2 Summary of Graphics Tasks Performed by Annotate Functions

Task Group	If you want to...	Use this function...
Graphics tasks	begin to draw a polygon (starting point) and, optionally, specify a fill color and pattern	POLY
	continue drawing a polygon (additional vertex) and, optionally, specify an outline color of the polygon	POLYCONT
	draw a line from the current (X,Y) position (see MOVE and TXT2CNTL)	DRAW
	draw a point	POINT
	draw a rectangle from the current (X,Y) position (see MOVE and TXT2CNTL); optionally, fill with a pattern	BAR
	draw a symbol	SYMBOL
	draw line from (XLAST, YLAST) coordinates to (XLSTT, YLSTT) coordinates	DRAW2TXT
	draw pie slice, circle, or arc	PIE
	draw text	LABEL
	move to the specified point (X,Y)	MOVE
	put a frame around the area defined by XSYS and YSYS, optionally, fill with a pattern	FRAME
Programming tasks	insert a comment in the data set (no action); documentation aid	COMMENT
	copy (XLAST, YLAST) coordinates to (XLSTT, YLSTT) coordinates	CNTL2TXT
	copy (XLSTT, YLSTT) coordinates to (XLAST, YLAST) coordinates	TXT2CNTL
	exchange LSTT and LAST coordinates	SWAP
	get coordinates of a point on a pie slice outline	PIEXY
	get values for LAST and LSTT coordinates from LIFO stack	POP
	put current values of LAST and LSTT coordinates onto LIFO stack	PUSH

Task Group	If you want to...	Use this function...
	set pie radius and coordinates for center; does not draw a pie	PIECNTR
	turn on trace of previous values and LIFO stack	DEBUG

About Annotate Graphics

When you create Annotate graphics, you specify these things:

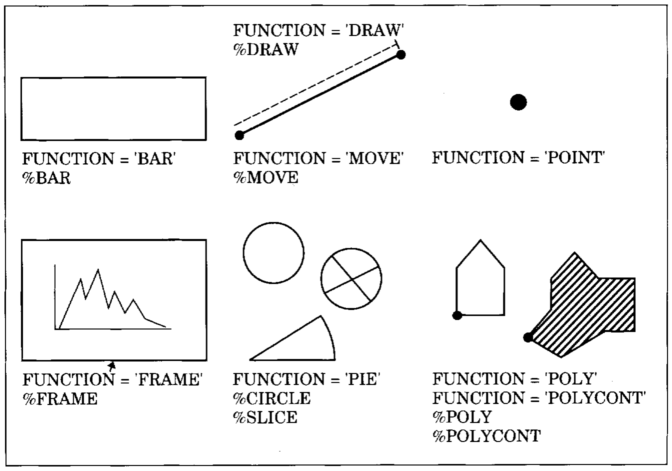
- □ what to draw (graphics elements)
- □ where to draw those elements (the coordinates of the position on the output)
- □ how to draw (characteristics of the element such as size or color).

The following sections describe these components of the graphics output that are produced by an Annotate data set.

Graphics Elements

In an Annotate data set, the FUNCTION variable determines the graphics element that is drawn. The particular graphics elements that you can draw are shown in Figure 10.4 on page 410 along with the value of the FUNCTION variable or Annotate macro that draws them.

Figure 10.4 Annotate Graphics Elements



You can control the position of graphics elements in the following ways:

- □ explicitly, using coordinates that you supply.
- □ dependently, based on the location of features in the SAS/GRAPH output. For example, when you use the GCHART procedure, you can label the parts of a subgrouped vertical bar chart by using the SUBGROUP variable in your Annotate data set. The Annotate facility enables you to label subgroups without having to specify the actual coordinates of the subgroup bar.

- dependently, based on values that are supplied from other data sets. For example, you can label the ending point of a plot line in the GPLOT procedure by extracting the value of the last point in the sorted input data set.

Coordinates

Coordinates specify where to put graphics elements. These variables can contain coordinate values:

- X, Y, and Z are used for numeric coordinates. (Use Z to annotate graphics output from the G3D procedure only.)
- XC and YC are used for character coordinates.
- GROUP, MIDPOINT, and SUBGROUP can be used when you annotate output from the GCHART procedure. Use these variables to specify coordinates for horizontal or vertical bar charts.

Coordinates are interpreted in terms of a coordinate system in order to identify a precise location in the graphics output.

Coordinate Systems

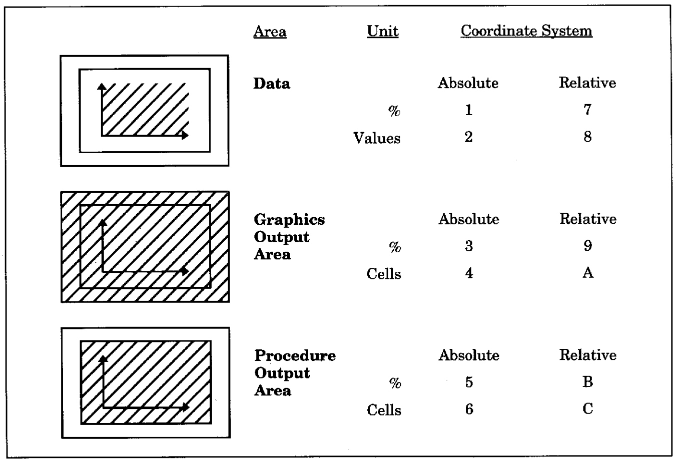
A coordinate system determines how coordinates are interpreted. You specify a coordinate system to use for each dimension, using the XSYS, YSYS, and ZSYS variables (for X, Y, and Z, respectively). Use ZSYS to annotate graphics output only from the G3D procedure.

You also specify a coordinate system for the SIZE variable using the HSYS variable. HSYS takes the same kinds of values as XSYS, YSYS, and ZSYS. The SIZE variable specifies the size of a graphics element, such as the width of lines (for example, FRAME), the radius of pie slices (for example, PIE, PIECNTR, and PIEXY), or the height of text (for example, LABEL and SYMBOL).

These are the important components of the Annotate coordinate systems:

- *Area:* Each coordinate system refers to one of three drawing areas: data area, procedure output area, and graphics output area. Coordinates are measured from a different origin for each area; they also have different limits. Figure 10.5 on page 412 shows the areas on the graphics output and the coordinate systems that use them.

Figure 10.5 Areas and Their Coordinate Systems



Area	Unit	Coordinate System	
Data		Absolute	Relative
	%	1	7
	Values	2	8
Graphics Output Area		Absolute	Relative
	%	3	9
	Cells	4	A
Procedure Output Area		Absolute	Relative
	%	5	B
	Cells	6	C

- □ *Units:* The units for a coordinate system are based on one of the following:
  - □ data values (for data coordinate systems). The range of values depends on the range of data expressed along the axes of the graph.
  - □ cells (for coordinate systems for the procedure output area or graphics output area). The range of values depends on the type of area. See "Ranges for Cells" on page 413.
  - □ percentages of the total area available, that is, percent of the data area, or percent of the procedure output area, or percent of the graphics output area.
- □ *Placement:* The placement of a coordinate can be absolute or relative. Absolute coordinates name the exact location for a graphics element in the graphics output. Relative coordinates name the location with respect to another graphics element in the output.

Table 10.3 on page 412 describes the coordinate system values for the XSYS, YSYS, ZSYS, and HSYS variables.

Table 10.3 Coordinate System Values for XSYS, YSYS, ZSYS, and HSYS Variables

Type of Coordinates	Area	Units	Range	Value for XSYS, YSYS, ZSYS, HSYS
Absolute	data	%	0-100% of axis	1' *
	data	values	minimum to maximum of axis	2' *
	graphics output area	%	0-100% of graphics output area	3'
	graphics output area	cells	0 to limit of graphics output area	4'
	procedure output area	%	0-100% of procedure output area	5'
	procedure output area	cells	0 to limit of procedure output area	6'
Relative	data	%	0-100% of axis	7' *

Type of Coordinates	Area	Units	Range	Value for XSYS, YSYS, ZSYS, HSYS
	data	values	minimum to maximum of axis	8' *
	graphics output area	%	0-100% of graphics output area	9'
	graphics output area	cells	0 to limit of graphics output area	A'
	procedure output area	%	0–100% of procedure output area	B'
	procedure output area	cells	0 to limit of procedure output area	C'

*Coordinate systems 1, 2, 7, and 8 are not valid with block, pie or star charts in the GCHART procedure or surface, prism or block maps with the GMAP procedure.

Ranges for Cells

The available range for coordinate systems that are measured in cells differs by area:

graphics output area
: The range of cells that are available for the graphics output area depends on the device and the number of rows and columns that are set by the HPOS= and VPOS= graphics options or by the PCOLS and LCOLS device parameters.

procedure output area
: As with the graphics output area, the range of cells available for the procedure output area depends on the device and the number of rows and columns set by the HPOS= and VPOS= graphics options or by the PCOLS and LCOLS device parameters. However, the procedure output area is sized *after* areas for titles and footnotes are allocated and is reduced accordingly. If you specify that the legend appear outside of the axis area, the procedure output area also decreases by the size of the legend.

See “Procedure Output and the Graphics Output Area” on page 29 for descriptions of the procedure output area and the graphics output area.

Internal Coordinates

The Annotate facility maintains two pairs of internal coordinates that are stored in internal variables:

- coordinates of the last graphics element drawn or the coordinates from the last move are stored in the variables XLAST and YLAST
- coordinates of the last text drawn are stored in the variables XLSTT and YLSTT.

Many functions use these internal coordinates as a starting point, relying on the coordinates that are specified with the function as an ending point. For example, in the BAR function, the (XLAST, YLAST) coordinate pair is used for the lower left corner; the position defined by the X and Y variables is used for the upper-right corner. (See the BAR function on “BAR Function” on page 431 for details.) These internal variables can also provide default coordinates if X, XC, Y, or YC contains a missing value.

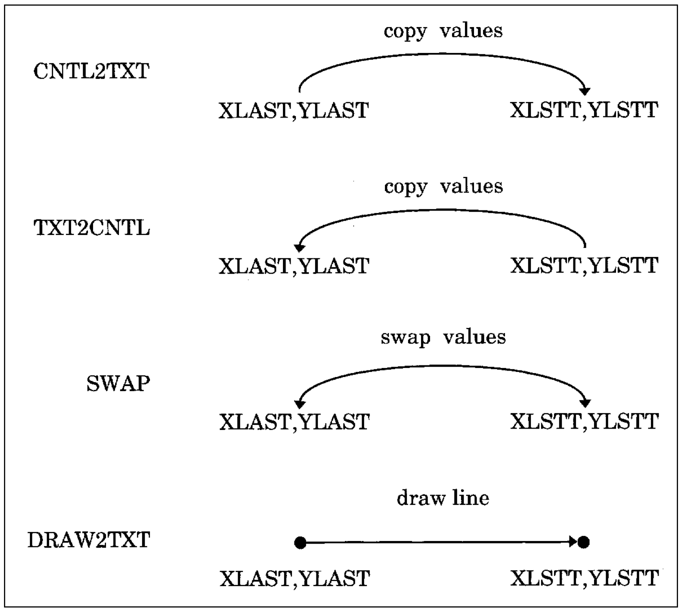
The internal coordinates are automatically updated by some of the Annotate functions. The text functions, LABEL and SYMBOL, update the (XLSTT,YLSTT) variables. The BAR, DRAW, MOVE, PIE, and POINT functions update the (XLAST,YLAST) variables.

You cannot explicitly assign a value to XLAST, YLAST, XLSTT, or YLSTT because they are internal variables. For example, you cannot make this assignment:

```
xlast=50;
```

However, you can use several functions to directly manipulate the values of the internal coordinates. The functions are shown in Figure 10.6 on page 414.

Figure 10.6 Programming Functions That Manipulate System Variables



For a complete description, see “Annotate Internal Coordinates” on page 483.

Attributes

Attribute variables control the appearance of the graphics elements. Each function uses only a subset of these variables. See Table 10.1 on page 407 for a list of attribute variables.

What an attribute variable controls often depends on the graphics element to which it applies. For example, the SIZE variable controls the width of a line when it is used with FUNCTION='DRAW', but it controls the text height when it is used with FUNCTION='LABEL'.

For a complete description of the attribute variables and the aspect of the graphics elements that they controls, see “Annotate Variables” on page 455.

Creating an Annotate Data Set

Once you have determined what you are going to draw and how you want it to appear in the output, you need to build an Annotate data set that contains the observations that are needed to generate the output. Although there are many ways to create SAS data sets, the most commonly used method for creating Annotate data sets is with a DATA step that uses either

- □ assignment statements that you explicitly output as separate observations
- □ Annotate macros, which implicitly assign values to Annotate variables.

Most of the examples in this book use a DATA step with assignment statements. For more information on creating SAS data sets, see *SAS Language Reference: Concepts* and *SAS Language Reference: Dictionary*.

Using the DATA Step

When you use the SAS DATA step with assignment statements, each statement provides a value for an Annotate variable. After you have assigned all of the variable values for an observation, you must use an OUTPUT statement to write the observation to the data set. For example, the following statements create the TRIANGLE data set shown in Output 10.1 on page 406:

```
data triangle;

      /* declare variables */
   length function style color $ 8 text $ 25;
   retain hsys xsys ysys '3';

      /* create observation to draw the title */
   function='label'; x=20; y=85; position='6';
      text='Sample Annotate Graphics';
      style='swissb'; color='green'; size=6;
      output;

      /* create observations to draw the triangle */
   function='move'; x=28; y=30; output;
   function='draw'; x=68; y=30; size=.8; line=1;
      color='red'; output;
   function='draw'; x=48; y=70; output;
   function='draw'; x=28; y=30; output;
run;

proc ganno data=triangle;
run;
quit;
```

Notice that a RETAIN statement sets the values of the HSYS, XSYS, and YSYS variables. RETAIN statements are useful when you want to select the values for variables that are required for many functions and the value is the same for all of them.

The SIZE, LINE, and COLOR variables are included with only the first DRAW function. Using this method to create the data set, the values set in the first DRAW function carry over to subsequent DRAW functions.

The PROC GANNO step creates the output shown in Figure 10.3 on page 407.

Using Annotate Macros in the DATA Step

A set of Annotate macros is provided in the SAS sample library. You can use macro calls in a DATA step to create observations in an Annotate data set. You can also use Annotate macros and explicit variable assignments together in the same DATA step. For complete information, see “Annotate Macros” on page 484.

Effect of Missing Values

Annotate data sets follow the same rules for missing values as any other SAS data set. (See *SAS Language Reference: Concepts* for information on the effect of missing values in a data set.)

Variables that have a missing value use a default value. For example, if the COLOR variable has a missing value, then the first color in either the colors list that is defined by the COLORS= graphics option, if specified, or the device's default colors list is used. If the FUNCTION variable has a missing value, LABEL is used. If the X variable is missing, the value of the XLSTT internal coordinate is used for text functions and the XLAST internal coordinate is used for nontext functions. See "Annotate Variables" on page 455 for the default value of each Annotate variable.

You probably should not depend on this effect when you create an Annotate data set. If the data set is structured so that observations depend on prior observations setting attributes for them, then you may have extra work to do if you change the order of observations later.

Sometimes missing values are required to produce the desired results. If you have calculated the coordinates of a point and have the values stored in (XLAST,YLAST) or (XLSTT,YLSTT), you can force Annotate to use the internal coordinates by supplying missing values for the X and Y variables. See "Annotate Internal Coordinates" on page 483 for details on using the (XLAST,YLAST) and (XLSTT,YLSTT) internal coordinates.

Producing Graphics Output from Annotate Data Sets

You can display Annotate graphics in two ways:

- annotate output from a SAS/GRAPH procedure by assigning the Annotate data set to the PROC statement or the action statement, or both.
- display only the Annotate graphics by assigning the Annotate data set to either the GANNO or GSLIDE procedure.

Including Annotate Graphics with Procedure Output

To annotate SAS/GRAPH procedure output, you must include the ANNOTATE= option in the appropriate statement in the procedure. ANNOTATE= must name the Annotate data set that you have already created. If you want the Annotate graphics to apply to all graphs produced by a procedure, you should include ANNOTATE= in the PROC statement. If you want the Annotate graphics to apply only to the graph produced by an action statement within the procedure, include ANNOTATE= in the action statement. You can specify Annotate data sets in both places.

When you annotate a SAS/GRAPH procedure, the Annotate graphics are displayed and stored as part of the graphics output that the procedure produces.

Producing Only Annotate Graphics Output

To produce Annotate graphics with other procedure output, use the GANNO procedure or the GSLIDE procedure:

- The GANNO procedure produces graphics output consisting only of Annotate graphics. See Chapter 12, "The GANNO Procedure," on page 503 for information on displaying or storing Annotate graphics.

- The GSLIDE procedure can also produce graphics output consisting only of Annotate graphics. In addition, you can enhance the graphics output with TITLE, NOTE, and FOOTNOTE statements. See Chapter 27, "The GSLIDE Procedure," on page 959 for details.

Processing Details

Order in Which Graphics Elements Are Drawn

When a procedure uses an Annotate data set, it reads and interprets the observations one at a time, starting with the first observation and proceeding to the last. The order of the observations in the data set determines the order in which the graphics elements are generated. If the coordinates of two graphics elements overlap, the graphics element produced by an earlier observation can be overwritten by any graphics elements that are produced by subsequent observations. As a result, graphics elements can overlay each other and they can also overlay or be overlaid by procedure output.

CAUTION:
Overlay behavior is device-dependent. Most terminals, cameras, and some printers demonstrate overlay behavior because the process of drawing updates pixels as each graphics element is drawn. Plotters do not overlay the graphics elements internally before plotting; they draw graphics elements on top of each other on the paper. The area where graphics elements overlap shows one color bleeding through the color that overlays it. To ensure that one graphics element overlays another, use the WHEN variable. △

Controlling the Processing with the WHEN Variable

The WHEN variable determines the order in which observations in an Annotate data set are processed. It determines if observations are processed *before* or *after* other observations and before or after output that is produced by a SAS/GRAPH procedure. This means that Annotate graphics can be overlaid by procedure output or can overlay procedure output. By default, Annotate graphics are drawn before the procedure output.

In effect, you can have two sets of Annotate graphics elements that are generated for the same output:

- Annotate graphics drawn before procedure output (the default, WHEN='B').
- Annotate graphics drawn after procedure output (WHEN='A').

Within each set, graphics elements are drawn in the order that they appear in the Annotate data set and overlay each other as appropriate (on devices that demonstrate overlay behavior). For details, see the description of the WHEN variable on "WHEN Variable" on page 474.

Using BY-Group Processing with the Annotate Facility

You can use the Annotate facility with procedures that use BY statements to annotate each graph that is generated with a BY statement. The Annotate graphics for each graph are generated depending on the value of the BY variable. To use BY-group processing with the Annotate facility, your program must meet the following conditions:

- Both the input data set for the procedure and the Annotate data set must contain the same BY variable.
- The BY variable must be defined as the same type (character or numeric) and length in both data sets.
- If a label or format is associated with a BY variable in one data set, the same label or format has to be associated with it in the other data set.
- Both data sets must be sorted by the BY variable.
- The ANNOTATE= option must be specified in an action statement in the procedure. If you specify the ANNOTATE= option in the PROC statement, the Annotate graphics are used for all graphs that are generated by the procedure rather than for unique values of the BY variable.

See "BY Statement" on page 177 for details.

Using the LIFO Stack

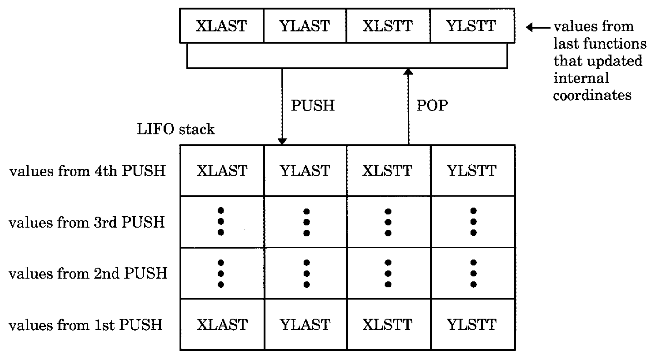
The FUNCTION variable supports several programming functions that manipulate the internal coordinates and provide other utility operations. Several of these functions use the LIFO stack to track and set variable values.

The *LIFO* (last-in-first-out) *stack* is a storage area where you can keep internal coordinate values for later use. It is useful when you want to save the current values of (XLAST,YLAST) and (XLSTT,YLSTT) and use them with functions later in the DATA step.

You store and retrieve values from the stack using the PUSH and POP functions. The PUSH function copies the current values of XLAST, YLAST, XLSTT, and YLSTT onto the stack. The POP function copies values from the stack into XLAST, YLAST, XLSTT, and YLSTT.

LIFO stacks manage the stored data so that the last data stored in the stack is the first data removed from the stack. This means that a POP function retrieves the values *most recently* stored with a PUSH function. Figure 10.7 on page 418 illustrates how PUSH and POP functions work together.

Figure 10.7 Using PUSH and POP to Store and Retrieve Coordinate Values



See also "Internal Coordinates" on page 413.

Debugging

You can print your Annotate data set with the PRINT procedure. This is an easy way to examine the Annotation that you have specified or to debug your program. For example, a listing such as the one in Output 10.1 on page 406 provides complete information about the value that you specify for each variable in every observation.

For more complex problems, the DEBUG function enables you to display the values of Annotate variables and internal coordinates before and after a function is submitted. The values are written to the SAS log.

If there is an error in your Annotate data set, one or more diagnostic messages are printed in the SAS log:

- If an error is found in preprocessing, this message appears:

```
NOTE: ERROR DETECTED IN ANNOTATE= libref.dataset
```

- If an error is found as an observation is being read, this message appears:

```
PROBLEM IN OBSERVATION number-message
```

  where *message* is the text of the error message.

- If the error limit of 20 errors is reached at any point during processing of the data set, a termination message similar to this one appears:

```
ERROR LIMIT REACHED IN ANNOTATE PROCESS

                20 TOTAL ERRORS
```

For an explanation of common diagnostic messages, refer to the Help facility.

Examples

The following examples show how to annotate graphics that are created with SAS/GRAPH procedures and how to build custom graphics.

"Example 1: Labeling Cities on a Map" on page 419

"Example 2: Labeling Subgroups in a Vertical Bar Chart" on page 422

"Example 3: Drawing a Circle of Stars" on page 424

Other examples that use Annotate data sets are in Chapter 12, "The GANNO Procedure," on page 503 (Example 1 on page 507 and Example 2 on page 509), and Chapter 27, "The GSLIDE Procedure," on page 959 (Example 2 on page 965).

For additional examples of the Annotate facility, see *SAS/GRAPH Software: Examples*.

Example 1: Labeling Cities on a Map

Features:

Annotate function:	LABEL
Annotate variables:	HSYS
	POSITION

SIZE

STYLE

TEXT

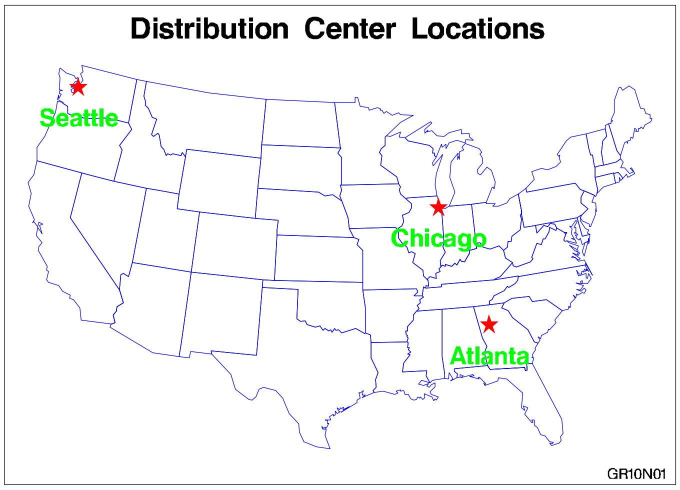
WHEN

X and Y

XSYS

YSYS

Sample library member:	GR10N01



This example labels a map of the continental United States with the location and names of three cities. The GMAP procedure draws a map of the U.S. and an Annotate data set adds the stars and labels.

The DATA step that creates the Annotate data set gets the *x* and *y* coordinates of the cities to be labeled from the MAPS.USCITY data set. Because MAPS.USCITY stores projected coordinates in the X and Y variables, the DATA step does not need to reassign the variable values. Also because X and Y contain data values (the map data set coordinates), the XSYS and YSYS variables specify coordinate system 2, absolute data values. However, the HSYS variable that controls text height uses coordinate system 3, percent of the graphics output area.

See Example 4 on page 891 for an example of labeling a map using map coordinates in units of latitude and longitude.

See Chapter 19, "The GMAP Procedure," on page 731 for more information on using map data sets.

Note: If the libref MAPS is automatically assigned at your site to the SAS data library containing the Institute-supplied map data sets, you can omit the LIBNAME statement. △

Assign the libref MAPS, if necessary, and set the graphics environment.

```
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
```

```
          colors=(black blue green red)
          ftext=swissb htitle=6 htext=3;
```

Subset the U.S. map data set by omitting Alaska and Hawaii.

```
data lower48;
   set maps.us;
   if state ne stfips('AK');
   if state ne stfips('HI');
run;
```

Create the Annotate data set, CITYSTAR. CITYSTAR contains the commands that draw a star and a label at each of the three cities. Because the RETAIN statement sets the value of FUNCTION to LABEL and no other function is specified, every observation uses LABEL. Setting WHEN to A draws the annotation after the map.

```
data citystar;
   length function style color $ 8 position $ 1
          text $ 20;
   retain function 'label' xsys ysys '2' hsys '3'
          when 'a';
```

Include the values of selected variables from MAPS.USCITY. X and Y contain projected coordinates; CITY contains names; STATE contains FIPS codes. Because there are several Atlantas, a STATE value is necessary.

```
   set maps.uscity(keep=x y city state);
   if (city='Atlanta' and state=13)
       or city='Chicago'
       or city='Seattle';
```

Create the observation that draws the star. The text string M is the character code for the star figure in the SPECIAL font assigned by the STYLE variable. A POSITION value of 5 centers the star on the city location.

```
   style='special'; text='M'; color='red'; size=7;
      position='5'; output;
```

Create the observation that labels the city. TEXT is assigned the value of CITY. The font is SWISSB. SIZE uses the units assigned by HSYS so text height is 7 percent of the height of the graphics output area. POSITION 8 places the label directly below the city location.

```
   style='swissb'; text=city; color='green';
      size=5; position='8'; output;
run;
```

Define the title and footnote for the map.

```
title 'Distribution Center Locations';
footnote font=swiss j=r 'GR10N101 ';
```

Define patterns for the map areas. MEMPTY colors only the state borders.

```
pattern value=mempty color=blue repeat=49;
```

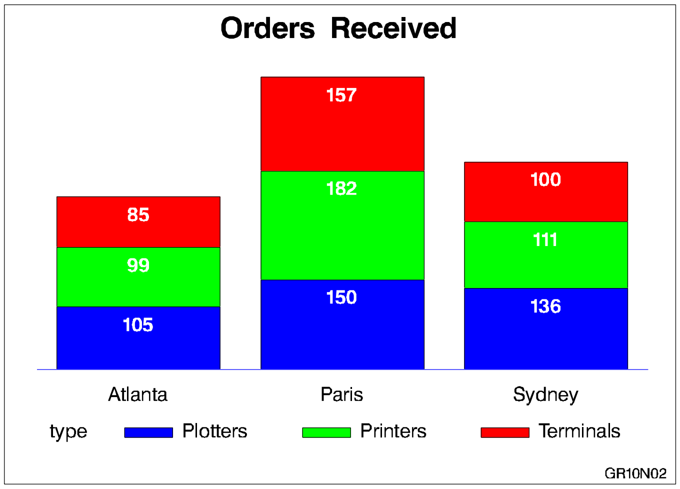
Generate the map and assign the annotate data set to the CHORO statement.

```
proc gmap data=lower48 map=lower48;
   id state;
   choro state / annotate=citystar discrete nolegend;
run;
quit;
```

Example 2: Labeling Subgroups in a Vertical Bar Chart

Features:

Annotate function:	LABEL (default)
Annotate variables:	MIDPOINT
	POSITION
	SUBGROUP
Sample library member:	GR10N02



This example shows how to label subgroups in a vertical bar chart that is generated by the GCHART procedure. Each bar represents total orders for a city and is subgrouped by the type of order. The Annotate facility labels each subgroup with the number of orders for that category. The coordinates that position the subgroup labels are derived from the values of the GCHART procedure variables CITY (the chart (or midpoint) variable) and TYPE (the subgroup variable). These variables are assigned to the corresponding Annotate variable.

See Chapter 13, “The GCHART Procedure,” on page 519 for more information on creating bar charts.

Assign the libref and set the graphics environment.

```
libname reflib 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(blue green red) ctext=black htitle=6
         ftitle=swissb htext=4 ftext=swiss;
```

Create the data set SOLD.

```
data sold;
   length type $ 10;
   input city $ units type $ ;
   datalines;
Atlanta  99 Printers
Atlanta 105 Plotters
...more data lines...
Sydney  136 Plotters
Sydney  100 Terminals
;
run;
```

Create the Annotate data set, BARLABEL. The MIDPOINT variable uses the values of the chart variable CITY to provide the X coordinate for the subgroup labels. The SUBGROUP variable uses the values of the variable TYPE to provide the Y coordinate that vertically positions the labels in the bar. Because no function is specified, the data set uses the default function, LABEL. The POSITION value **E** places the labels just below the top of each subgroup bar.

```
data barlabel;
   length color style $ 8;
   retain color 'white' when 'a' style 'swissb'
      xsys ysys '2' position 'E' size 4 hsys '3';
   set sold;
   midpoint=city;
   subgroup=type;
   text=left(put(units,5.));
run;
```

Define the title and footnote.

```
title 'Orders Received';
footnote h=3 j=r 'GR10N02 ';
```

Define axis characteristics. AXIS1 suppresses the vertical axis. AXIS2 drops the midpoint axis label.

```
axis1 label=none major=none minor=none style=0
      value=none;
```

```
axis2 label=none;
```

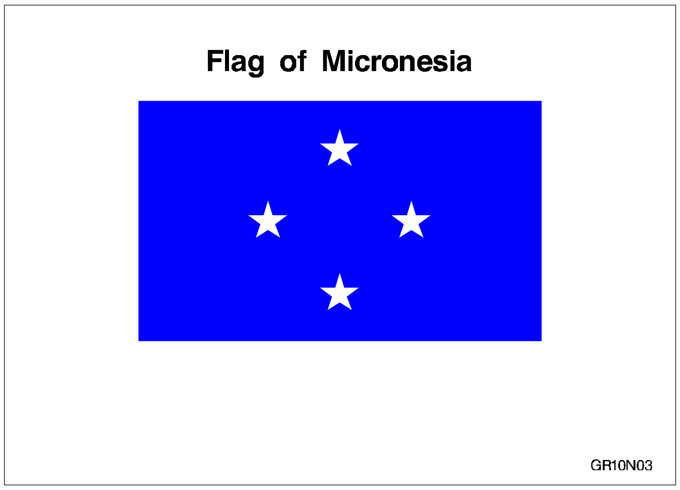
Generate a vertical bar chart and assign the Annotate data set to the VBAR statement.

```
proc gchart data=sold;
   vbar city / type=sum
               sumvar=units
               subgroup=type
               width=17
               raxis=axis1
               maxis=axis2
               annotate=barlabel;
run;
quit;
```

Example 3: Drawing a Circle of Stars

Features:

Annotate function:	BAR
	CNTL2TXT
	FRAME
	LABEL
	MOVE
	PIECNTR
	PIEXY
Annotate variables:	COLOR
	HSYS, XSYS, YSYS
	LINE
	STYLE
	TEXT
	X and Y
	XLAST and YLAST
	XLSTT and YLSTT
Sample library member:	GR10N03



This example shows how to use an Annotate data set to draw a flag that is composed of a rectangle and four stars. The stars are positioned by placing them on an imaginary circle. The program uses the PIECNTR and PIEXY functions to find the points on the circle and the CNTL2TXT programming function to transfer coordinate values. It also processes Annotate assignment statements in a DO loop. The GANNO procedure displays the Annotate graphics.

Set the graphics environment.

```
goptions reset=global cback=white colors=(black);
```

Create the Annotate data set, FLAG. XSYS, YSYS, and HSYS specify coordinate system 3, absolute size of the graphics output area.

```
data flag;
   length function style color $ 8 text $ 30;
   retain xsys ysys hsys '3';
```

Draw a frame. The FRAME function uses the default color BLACK to draw a frame around the graphics output area specified by the XSYS and YSYS variables.

```
   function='frame'; output;
```

Draw the footnote. The LABEL function draws the text specified in the TEXT variable. X and Y explicitly position the footnote on the graphics output area.

```
   function='label'; x=92; y=5; text='GR10N03';
      style='swiss'; size=3; position='5'; output;
```

Draw the title. The values of FUNCTION, POSITION, and COLOR remain the same because no new values are assigned.

```
   x=50; y=90; text='Flag of Micronesia';
      style='swissb'; size=6; output;
```

Draw the background. MOVE specifies the lower left corner of the rectangle that forms the flag. BAR draws the rectangle using the values of X and Y for the upper right corner. The LINE value of 3 fills the figure with the specified color.

```
   function='move'; x=20; y=30; output;
   function='bar'; x=80; y=80; color='blue';
      line=3; style='solid'; output;
```

Draw the circle of stars. The DO loop repeats the processing instructions defined by the nested assignment statements, placing a star every 90 degrees around the circle. To increase the number of stars, reduce the size of the angle between them and adjust the ending angle.

```
do star_ang=0 to 270 by 90;
```

The PIECNTR function is set to the center of the rectangle. PIEXY calculates a point on the arc based on the value of STAR_ANG and updates the internal coordinates XLAST and YLAST.

```
   function='piecntr'; x=50; y=55; size=15; output;
   function='piexy'; size=1; angle=star_ang; output;
```

The programming function CNTL2TXT copies the values of XLAST and YLAST to the text-handling coordinates XLSTT and YLSTT. Assigning missing values to X and Y forces the LABEL function to use the values of XLSTT and YLSTT to position the star. The text string V is the character code for the star figure in the MARKER font assigned by the STYLE variable.

```
   function='cntl2txt'; output;
   function='label'; style='marker'; text='V';
      angle=0; color='white'; size=10; x=.; y=.;
      output;
   end;
run;
```

Use the GANNO procedure to process the Annotate data set and generate the graphics output.

```
proc ganno annotate=flag;
run;
quit;
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/GRAPH® Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/GRAPH® Software: Reference, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–525–6

All rights reserved. Printed in the United States of America.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999