SAS/GRAPH® 9.4: Reference, Fifth Edition

SAS® Documentation
October 2, 2023

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS/GRAPH® 9.4: Reference, Fifth Edition*. Cary, NC: SAS Institute Inc.

SAS/GRAPH® 9.4: Reference, Fifth Edition

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

October 2023

9.4-P9:graphref

Contents

PART 4 Controlling the Appearance of Your Output 319

PART 5 Statements and Options Used by SAS/GRAPH 395

About This Book

Using This Document

Prerequisites

This document is written for users who are experienced in using SAS. You should understand the concepts of programming in the SAS language. The following table summarizes the SAS concepts that you need to understand in order to use SAS/GRAPH.

To learn how to	Refer to
invoke SAS at your site	instructions provided by the on-site SAS support personnel
use Base SAS software use the DATA step to create and manipulate SAS data sets use SAS Studio, the SAS windowing environment, or SAS Enterprise Guide to enter, edit, and submit program code	Base SAS documentation: ■ *SAS Programmer's Guide: Essentials* ■ *SAS Data Set Options: Reference* ■ *SAS Formats and Informats: Reference* ■ *SAS Functions and CALL Routines: Reference* ■ *SAS DATA Step Statements: Reference* ■ *SAS System Options: Reference* ■ *Base SAS Utilities: Reference*
allocate SAS libraries and assign librefs create external files and assign filerefs	documentation for using SAS in your operating environment: ■ *SAS Companion for Windows* ■ *SAS Companion for UNIX Environments* ■ *SAS Companion for z/OS*

To learn how to	Refer to
manipulate SAS data sets using SAS procedures	*Base SAS Procedures Guide*

Particular fonts have special meanings when used in the presentation of SAS/GRAPH syntax in this document. For example, items presented in italics identify arguments or values that you supply. Angle brackets (< >) indicate optional arguments. The conventions used in this document are the same conventions used in Base SAS documentation. For a complete explanation, see the Base SAS documentation listed above.

Mapping Procedures and Map Data Sets

Starting with SAS 9.4M4, information about creating maps was moved from this SAS/GRAPH Reference to a separate document, *SAS/GRAPH and Base SAS: Mapping Reference*. That separate document includes the GEOCODE, GINSIDE, GMAP, GPROJECT, GREDUCE, GREMOVE, and MAPIMPORT procedures, and their supporting documentation. What remains in this reference document is the remaining SAS/GRAPH procedures and their related subject matter.

To draw maps, you need to know how to access the map data sets that are stored on your system. Depending on your SAS/GRAPH installation, the map data set supplied by SAS might automatically be assigned a libref, such as MAPSGFK, MAPSSAS, or MAPS. By default, the MAPS libref is set equal to the MAPSSAS libref, sharing the same physical name (path). MAPS and MAPSSAS contain the non-vector-based map data sets. The MAPSGFK map library contains vector-based maps data sets that SAS has licensed from GfK GeoMarketing GmbH. SAS updates this map library as it receives updates from GfK, and provides the library for use with SAS/GRAPH. Typically, a SAS program specifying these librefs runs without the need to know where the map data sets reside. If necessary, ask your on-site SAS support personnel or system administrator where the map data sets are stored for your site.

SAS/GRAPH procedures that work with map data sets include the GINSIDE, GMAP, GREDUCE, and GREMOVE procedures. Starting with SAS 9.4M5, the GEOCODE, GPROJECT, and MAPIMPORT procedures, along with a new SGMAP procedure, are offered with Base SAS. The SGMAP procedure works without the MAPSGFK map data set to display OpenStreetMaps and EsriMaps with plot overlays. If your site has SAS/GRAPH installed, you have access to the map data set MAPSGFK. Map libraries MAPSSAS and MAPS can be installed separately from http://support.sas.com/rnd/datavisualization/mapsonline/index.html Mapping procedures also work with map data sets from third-party sources or maps that you provide.

Annotate Macros SAS Programs

To use Annotate macros, you need to know where the Annotate macro SAS program (annomac.sas) is stored on your system. Depending on your installation, the Annotate macro data set might automatically be assigned a fileref. Ask your on-

site SAS support personnel or system administrator where the Annotate macro data set is stored for your site.

About Examples, Their Output, and Sample Data Sets

Most of the chapters in this document include examples that illustrate some of the features of a procedure or its statements. The output that is shown for the examples was generated in a Windows operating environment. If you are using a different operating environment, you might need to make some minor adjustments to the example programs.

In most cases, the output was sent to the HTML destination and generated using the default style and device for that destination. Exceptions are noted in the text.

The dimensions of the graphics output area vary across devices and when using the GRAPH windows. The dimensions can affect aspects of the graphics output – for example, the appearance of axes or the position of graphics elements that use explicit coordinates in units other than percent. You might need to adjust the dimensions of your graphics output area or the size of graphics elements to correct any differences that you see. Most of the images of output in this document were generated with a GOPTIONS statement. This statement generally specified a size approximately equal 5.5 inches by 4.2 inches. However, some images might be larger, if necessary, to accommodate the content of the graph.

```
goptions hsize=5.5in vsize=4.2in;
```

These HSIZE= and VSIZE= settings are not shown in the example code and are not necessary for generating the output. However, you might want to use similar settings if your output looks significantly different from the output that is shown in the document.

Most examples specify the following graphics options in the GOPTION statement:

RESET=ALL
: sets all graphics options to default values and cancels all global statements.

BORDER
: draws a border around the graphics output area.

Many examples process sample data contained in a SAS data set stored in the SASHELP library. The documentation for each example provides the data set name used. To see a brief description of any SAS data set in the SASHELP library, as well as output displaying the first five observations in each data set, please refer to SASHELP Data Sets.

Syntax Conventions for the SAS Language

Overview of Syntax Conventions for the SAS Language

SAS uses standard conventions in the documentation of syntax for SAS language elements. These conventions enable you to easily identify the components of SAS syntax. The conventions can be divided into these parts:

- syntax components
- style conventions
- special characters
- references to SAS libraries and external files

Syntax Components

The components of the syntax for most language elements include a keyword and arguments. For some language elements, only a keyword is necessary. For other language elements, the keyword is followed by an equal sign (=). The syntax for arguments has multiple forms in order to demonstrate the syntax of multiple arguments, with and without punctuation.

keyword
specifies the name of the SAS language element that you use when you write your program. Keyword is a literal that is usually the first word in the syntax. In a CALL routine, the first two words are keywords.

In these examples of SAS syntax, the keywords are bold:

CHAR (*string, position*)
CALL RANBIN (*seed, n, p, x*);
ALTER (*alter-password*)
BEST *w*.
REMOVE <*data-set-name*>

In this example, the first two words of the CALL routine are the keywords:

CALL RANBIN(*seed*, *n*, *p*, *x*)

The syntax of some SAS statements consists of a single keyword without arguments:

DO;
... *SAS code* ...
END;

Some system options require that one of two keyword values be specified:

DUPLEX | **NODUPLEX**

Some procedure statements have multiple keywords throughout the statement syntax:

CREATE <UNIQUE> **INDEX** *index-name* **ON** *table-name* (*column-1* <, *column-2*, …>)

argument
specifies a numeric or character constant, variable, or expression. Arguments follow the keyword or an equal sign after the keyword. The arguments are used by SAS to process the language element. Arguments can be required or optional. In the syntax, optional arguments are enclosed in angle brackets (< >).

In this example, *string* and *position* follow the keyword CHAR. These arguments are required arguments for the CHAR function:

CHAR (*string, position*)

Each argument has a value. In this example of SAS code, the argument *string* has a value of 'summer', and the argument *position* has a value of 4:

```
x=char('summer', 4);
```

In this example, *string* and *substring* are required arguments, whereas *modifiers* and *startpos* are optional.

FIND(*string, substring* <, *modifiers*> <, *startpos*>

argument(s)
specifies that one argument is required and that multiple arguments are allowed. Separate arguments with a space. Punctuation, such as a comma (,) is not required between arguments.

The MISSING statement is an example of this form of multiple arguments:

MISSING *character(s)*;

<LITERAL_ARGUMENT> *argument-1* <<LITERAL_ARGUMENT> *argument-2* ... >
specifies that one argument is required and that a literal argument can be associated with the argument. You can specify multiple literals and argument pairs. No punctuation is required between the literal and argument pairs. The ellipsis (...) indicates that additional literals and arguments are allowed.

The BY statement is an example of this argument:

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …>;

argument-1 <*options*> <*argument-2* <*options*> ...>
specifies that one argument is required and that one or more options can be associated with the argument. You can specify multiple arguments and associated options. No punctuation is required between the argument and the option. The ellipsis (...) indicates that additional arguments with an associated option are allowed.

The FORMAT procedure PICTURE statement is an example of this form of multiple arguments:

PICTURE name <(*format-options*)>
<*value-range-set-1* <(*picture-1-options*)>
<*value-range-set-2* <(*picture-2-options*)> …>>;

argument-1=value-1 <argument-2=value-2 ...>
specifies that the argument must be assigned a value and that you can specify multiple arguments. The ellipsis (...) indicates that additional arguments are allowed. No punctuation is required between arguments.

The LABEL statement is an example of this form of multiple arguments:

LABEL *variable-1=label-1 <variable-2=label-2* …>;

argument-1 <, *argument-2*, ...>
specifies that one argument is required and that you can specify multiple arguments that are separated by a comma or other punctuation. The ellipsis (...) indicates a continuation of the arguments, separated by a comma. Both forms are used in the SAS documentation.

Here are examples of this form of multiple arguments:

AUTHPROVIDERDOMAIN (*provider-1:domain-1* <, *provider-2:domain-2*, …>

INTO :*macro-variable-specification-1* <, :*macro-variable-specification-2*, …>

Note: In most cases, example code in SAS documentation is written in lowercase with a monospace font. You can use uppercase, lowercase, or mixed case in the code that you write.

Style Conventions

The style conventions that are used in documenting SAS syntax include uppercase bold, uppercase, and italic:

UPPERCASE BOLD
identifies SAS keywords such as the names of functions or statements. In this example, the keyword ERROR is written in uppercase bold:

ERROR <*message*>;

UPPERCASE
identifies arguments that are literals.

In this example of the CMPMODEL= system option, the literals include BOTH, CATALOG, and XML:

CMPMODEL=BOTH | CATALOG | XML |

italic
identifies arguments or values that you supply. Items in italic represent user-supplied values that are either one of the following:

- nonliteral arguments. In this example of the LINK statement, the argument *label* is a user-supplied value and therefore appears in italic:

  LINK *label*;

- nonliteral values that are assigned to an argument.

  In this example of the FORMAT statement, the argument DEFAULT is assigned the variable *default-format*:

  FORMAT *variable(s)* <*format* > <DEFAULT = *default-format*>;

Special Characters

The syntax of SAS language elements can contain the following special characters:

=
an equal sign identifies a value for a literal in some language elements such as system options.

In this example of the MAPS system option, the equal sign sets the value of MAPS:

MAPS=*location-of-maps*

< >
angle brackets identify optional arguments. A required argument is not enclosed in angle brackets.

In this example of the CAT function, at least one item is required:

CAT (*item-1* <, *item-2*, …>)

|
a vertical bar indicates that you can choose one value from a group of values. Values that are separated by the vertical bar are mutually exclusive.

In this example of the CMPMODEL= system option, you can choose only one of the arguments:

CMPMODEL=BOTH | CATALOG | XML

...
an ellipsis indicates that the argument can be repeated. If an argument and the ellipsis are enclosed in angle brackets, then the argument is optional. The repeated argument must contain punctuation if it appears before or after the argument.

In this example of the CAT function, multiple *item* arguments are allowed, and they must be separated by a comma:

CAT (*item-1* <, *item-2*, …>)

'*value*' or "*value*"
indicates that an argument that is enclosed in single or double quotation marks must have a value that is also enclosed in single or double quotation marks.

In this example of the FOOTNOTE statement, the argument *text* is enclosed in quotation marks:

FOOTNOTE <*n*> <*ods-format-options* '*text*' | "*text*">;

;
a semicolon indicates the end of a statement or CALL routine.

In this example, each statement ends with a semicolon:

```
data namegame;
   length color name $8;
   color = 'black';
   name = 'jack';
   game = trim(color) || name;
run;
```

References to SAS Libraries and External Files

Many SAS statements and other language elements refer to SAS libraries and external files. You can choose whether to make the reference through a logical name (a libref or fileref) or use the physical filename enclosed in quotation marks.

If you use a logical name, you typically have a choice of using a SAS statement (LIBNAME or FILENAME) or the operating environment's control language to make the reference. Several methods of referring to SAS libraries and external files are available, and some of these methods depend on your operating environment.

In the examples that use external files, SAS documentation uses the italicized phrase *file-specification*. In the examples that use SAS libraries, SAS documentation uses the italicized phrase *SAS-library* enclosed in quotation marks:

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```

Recommended Reading

Here is the recommended reading list for this title:

- *Producing High-Quality Figures Using SAS/GRAPH® and ODS Graphics Procedures*
- *SAS/GRAPH and Base SAS: Mapping Reference*

What's New in SAS/GRAPH 9.4: Reference

Overview

Note: SAS 9.4M6: This What's New chapter covers non-mapping SAS/GRAPH procedure updates. See "What's New in SAS/GRAPH and Base SAS 9.4: Mapping Reference" in *SAS/GRAPH and Base SAS: Mapping Reference* for SAS/GRAPH and Base SAS mapping procedure updates.

There are many changes and enhancements for SAS/GRAPH 9.4. Highlights include the following:

- Many procedures have significant enhancements and new options. See "Procedures" on page xxv for a complete list.
- There are three new Universal Printer shortcut devices: EMFDUAL, TIFF, and TIFFK.
- The SVG and GIF devices now support animation. See "Devices and Universal Printers" on page xxiii for more information.
- The Annotate Facility has several enhancements. See "The Annotate Facility" on page xxiv for a complete list.
- The Tile Chart applet supports adding one or more custom menu items to the tile chart pop-up menu. See "The Tile Chart Applet" on page xxviii.
- The Treeview applet supports assigning multiple drill-down links for each node in the diagram. See "The Treeview Applet" on page xxviii.
- SAS 9.4M2:
  - TITLE and FOOTNOTE statements now support the ALT=*"text-string"* option. See "Global Statements" on page xxix.
  - Information about the use of the JAVA and ActiveX devices was moved from the SAS/GRAPH Reference document to a separate document, *SAS/GRAPH: Java Applets and ActiveX Control User's Guide*. Information about the level of support that SAS/GRAPH statements and options have for Java and ActiveX devices remain, including the Appendix Appendix 1, "Summary of ActiveX and Java Support," on page 1581.
  - The first and second parts of this reference were reorganized to consolidate device and output topics.

- SAS 9.4M3:
  - A topic in the introduction consolidates links for examples and various resources on the SAS website.
  - Two charts of the predefined colors in SAS are added to the appendix. Each chart lists the color name, the CX color code, the HLS color code, and a sample color swatch for each of the predefined colors. The colors are sorted by name in the first chart and by hue in the second chart.
- SAS 9.4M4: Information about creating maps was moved from the SAS/GRAPH Reference to a separate document, *SAS/GRAPH and Base SAS: Mapping Reference*. This document includes the GEOCODE, GINSIDE, GMAP, GPROJECT, GREDUCE, GREMOVE, and MAPIMPORT procedures, and their supporting documentation.
- SAS 9.4M5:
  - Information about using SAS/GRAPH to plot data that is processed in SAS Cloud Analytic Services is added to the SAS/GRAPH Reference. See Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 153.
  - SAS/GRAPH Network Visualization Workshop is discontinued and is no longer supported.
- SAS 9.4M7:
  - The SAS/GRAPH Java applets are deprecated. This includes the following:
    - JAVA and JAVAMETA graphics devices
    - %DS2CONST and %DS2TREE macros
    - Constellation applet
    - Contour applet
    - Graph applet
    - Map applet
    - Tile Chart applet
    - Treeview applet

    Existing programs that use these items continue to work. However, these items are no longer supported, and they might be removed in a future release.
  - The document *SAS/GRAPH: Java Applets and ActiveX Control User’s Guide* is discontinued. For information about generating interactive ActiveX output in SAS/GRAPH, see Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

The following changes to Base SAS are of interest to SAS/GRAPH users.

- SAS 9.4M1: Descriptions of each data set in the SASHELP library are available online. To see these descriptions, as well as output displaying the first five observations in each data set, see SASHELP Data Sets.
- SAS 9.4M3:
  - The DEVICE=, GSTYLE, and GWINDOW system options are now documented in the *SAS/GRAPH 9.4: Reference*. The MAPS= system option is now documented in the *SAS/GRAPH 9.4: Mapping Reference*.

  - HTMLBlue is the default style for the HTML ODS destination when using SAS both in batch mode and in the window environment. Previously, HTMLBlue was the default style only when using SAS in the window environment.
  - The following ODS styles have been removed from suggested use: Astronomy, Banker, Curve, Education, Electronics, Gears, Magnify, Money, RSVP, Science, Sketch, Solutions, Torn, and Watercolor. These styles still function when used in SAS programs, but they no longer appear in the list of available styles.
- SAS 9.4M5: The BlockPrint ODS style is removed from suggested use. This style still functions when used in SAS programs, but it no longer appears in the list of available styles.
- SAS 9.4M6: The %CENTROID macro is available with Base SAS. The macro is no longer part of SAS/GRAPH, nor is SAS/GRAPH required to be installed to use this macro. This macro is useful when adding labels to a map, or when computing distances with the Base SAS GEODIST function. See the "%CENTROID Macro Autocall Macro" on page 859, and the "GEODIST Function" in *SAS Functions and CALL Routines: Reference* for more information.

Devices and Universal Printers

- The **SVG device** now supports animation.
- The GIF device is now a **Universal Printer shortcut device** for the GIF Universal Printer. As a result, the GIF device now supports RGBA color mode (see **Support For Transparency**) and anti-aliasing. Anti-aliasing improves the appearance of text, angled plot lines, map borders, and so on.

  Also, the GIF device now supports animation, and using this device is the preferred method for doing **GIF animations**. Documentation for the SAS/GRAPH GIFANIM device has been removed from *SAS/GRAPH: Reference*.
- The new TIFF and TIFFK devices produce TIFF images and support the RGBA and CMYK color schemes, respectively.
- The **EMF device** is now a **Universal Printer shortcut device** for the EMF Universal Printer. The default format for the EMF Universal Printer is now Enhanced Metafile Format Plus Extensions (EMF Plus). It supports RGBA colors (transparency).
- The new **EMFDUAL device** is a Universal Printer shortcut device that produces graphics file containing both EMF and EMF Plus records. It supports RGBA colors (transparency).
- The new **%SHORTCUT autocall macro** creates a shortcut device that is linked to the existing Universal Printers of the same name. It can also define a new Universal Printer and shortcut device and link the device to the Universal Printer.
- SAS 9.4M2:
  - The ZPNG device is disabled.
  - The SVG, SVGT, and SVGView devices now add the ROLE and ARIA-LABEL attributes in the HTML output. The attributes are available when

custom data tips are specified for hotspots and the ODS HTML5 destination is used. These attributes make these data tips accessible to iPad users who are using the Voiceover screen reader.

- The PowerPoint destination supports both the JAVAIMG device and the ACTXIMG device.

- SAS 9.4M5: The IMGPNG and IMGGIF devices are available. These devices can be used to improve the performance of SAS jobs that generate a large number of graphs. When used to render a large number of graphs, the performance of these devices is better than their corresponding Universal Printer shortcut devices, PNG and GIF.
- SAS 9.4M7: The JAVA and JAVAMETA graphics devices are deprecated. Existing programs that use these devices continue to work. However, these devices are no longer supported, and they might be removed in a future release.
- SAS 9.4M8:
  - The ACTIVEX and ACTXIMG graphics devices are obsolete. In existing programs, if DEVICE=ACTIVEX or DEVICE=ACTXIMG is specified, the JAVAIMG device is used instead.
  - The JAVA and JAVAMETA graphics devices are obsolete. They no longer ship with SAS. In existing programs, if DEVICE=JAVA is specified, the JAVAIMG device is used instead, and if DEVICE=JAVAMETA is specified, the PNG device is used instead. In both cases, the graph that is generated is a static image.
  - In the z/OS operating environment, the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices are no longer supported. If you attempt to use one of these devices on z/OS, an error results.

Support for Transparency

The GIF and EMF devices and the new EMFDUAL device support RGBA color mode. RGBA colors enable you to specify transparency. (The RGBA color mode supports alpha channel blending. Overlapping colors of varying opacity are blended together.) See “Using Transparency” on page 367 for more information.

The Annotate Facility

- The HTML= option has been added to the **ARROW function**. The drill-down link or tooltip is applied to the line arrowhead.
- The **IMGPATH variable** now accepts a URL location for graphics that are on web pages.
- The HTML= option enables you to animate text labels that are created with the **LABEL function**. When you specify the HTML= variable, you can include the

animation action and the parameters that control the animation. The animated label feature is available only when you use an SVG graphics device.

- The %CENTROID macro has been enhanced to return centroid locations with a higher degree of accuracy.
- SAS 9.4M3: The maximum length for the XC= and YC= annotate variables is increased to 256 characters. In prior releases, the maximum length is 32 characters.
- SAS 9.4M5: The STYLE= variable for images supports SINGLE, which centers a single instance of the image on the specified coordinates.

Procedures

GAREABAR Procedure

SAS 9.4M8: Because the ACTIVEX and ACTXIMG graphics devices are obsolete, the GAREABAR procedure is also obsolete. If you attempt to run the GAREABAR procedure, an error message is written to the SAS log. As an alternative, you can use the ODS Graphics SGPLOT procedure POLYGON statement to create simple area bar charts as described in the post Area Bar Charts using SGPLOT on the Graphically Speaking blog.

GBARLINE Procedure

SAS 9.4M5: The **GBARLINE procedure** provides a new statistic format option STATFMT= on its BAR and the PLOT statements. This option applies a specified format to a calculated statistical value such as that specified with the frequency (FREQ=) option of the TYPE= option. The STATFMT= option overrides the GBARLINE procedure's default format of the displayed statistical value. Use this option to change the default format that might contain decimal points, percentages, or commas.

GCHART Procedure

The **GCHART procedure** has the following new options on the PIE, PIE3D, and DONUT statements:

- The PPERCENT= option modifies the font, height, and color of the percentages displayed in pie slice labels. When used with the PLABEL= option (which

controls the text attributes of pie slice labels), you can easily differentiate between percentages and text labels.

- The EXPLODE=ALL option pulls all of the slices outward from the center of the pie. Use the ALL value rather than specifying a list of all the slices to explode.

SAS 9.4M5: The **GCHART procedure** has the following new options:

- The new GROUPREF option on the HBAR and VBAR statements provides the ability to draw a dividing line between bars or groups of bars. In addition, three new options enable you to manage the appearance of this reference line: option CGROUPREF manages line color, option LGROUPREF manages line type, and option WGROUPREF manages line width.
- Four new options enable you to draw a reference line between the midpoints of bars or groups of bars. Option MIDPOINTREF on the HBAR and VBAR statements provides the ability to draw a dividing line between the midpoints of bars or groups of bars. Three additional options enable you to manage the appearance of the midpoint reference line: option CMIDPOINTREF manages line color, option LMIDPOINTREF manages line type, and option WMIDPOINTREF manages line width.
- The new frequency format option STATFMT= on the BLOCK, VBAR, VBAR3d, HBAR, HBAR3d, PIE, PIE3D, DONUT, and STAR statements applies a specified format to a calculated statistical value such as that specified with the frequency (FREQ=) option of the TYPE= option. This option overrides the GCHART procedure's default format of the displayed statistical value. Use this option to change the default format that might contain decimal points, percentages, or commas.

GKPI Procedure

The **GKPI procedure** has the following changes and enhancements:

- The FORMAT= option now accepts user-defined formats.
- SAS 9.4M8: The ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices are no longer supported in the z/OS operating environment. If you attempt to use one of these devices on z/OS, an error results. Because the GKPI procedure supports only the JAVAIMG device, starting with SAS 9.4M8, the GPKI procedure no longer runs on z/OS.

GPLOT Procedure

The **GPLOT procedure** has the following changes and enhancements:

- Both the HZERO and VZERO requests on a PLOT, BUBBLE, PLOT2, or BUBBLE2 statement are ignored if the associated variable contains any time–formatted values, such as date, time, or datetime.
- The PLOT and PLOT2 statements in the GPLOT procedure support the new CBASELINE= option. This specifies the color of the horizontal baseline for

reference lines of a plot. This option compensates for a reference line obscuring the axis.

- SAS 9.4M1: The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.
- SAS 9.4M3: The CTEXT= option in the GPLOT procedure's PLOT statement is updated. It can now affect the color of POINTLABEL symbols.

GRADAR Procedure

SAS 9.4M2: The **GRADAR Procedure** has a new parameter on the SPKLABEL= option. Specifying the SPKLABEL=ALL option generates a spoke (axis) label for each graph displayed in the procedure output area, rather than for the first graph only.

GTILE Procedure

The **GTILE procedure** has the following changes and enhancements:

- SAS 9.4M2: The NOLEGEND option suppresses the legend that is automatically generated by default.
- SAS 9.4M5: If a device other than JAVA, JAVAIMG, ACTIVEX, or ACTXIMG is specified, the JAVAIMG device is used to generate the tile chart instead. In prior releases, the tile chart is not generated in that case.
- SAS 9.4M8: The ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices are no longer supported in the z/OS operating environment. If you attempt to use one of these devices on z/OS, an error results. Because the GTILE procedure supports only the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices, starting with SAS 9.4M8, the GTILE procedure no longer runs on z/OS.

SAS/GRAPH Java Applets for Web Servers

Starting with SAS 9.4M8, SAS/GRAPH Java Applets for Web Servers is not available from SAS. If you order SAS/GRAPH 9.4M8 or a later version, SAS/GRAPH Java Applets for Web Servers is not included in that order. For more information, see Unconfiguring and Uninstalling Retired Products in *SAS Guide to Software Updates and Product Changes*.

The Tile Chart Applet

The Tile Chart applet now supports adding one or more custom menu items to the tile chart pop-up menu. You can add custom menu items for all levels of the pop-up menu or for specific levels only. Typically, each custom menu item is linked to a web resource. When the user right-clicks the tile chart, and then selects a custom item in the pop-up menu, the web resource associated with that item appears in a web browser window.

You can add custom pop-up menu items by specifying parameters in the PARAMETERS= option in the ODS HTML statement. The **MENU*n*LABEL=** and **MENU*n*VALUE=** parameters add custom menu item *n* to all levels of the chart. The **MENU*n.m*LABEL=** and **MENU*n.m*VALUE=** parameters add custom menu item *n* to the level *m* pop-up menu only. In the ODS HTML statement, you can specify parameters MENU*n*LABEL and MENU*n*VALUE, or parameters MENU*n.m*LABEL and MENU*n.m*VALUE, but not both.

SAS 9.4M7: The Tile Chart applet is deprecated. Existing programs that use this applet continue to work. However, this applet is no longer supported, and it might be removed in a future release.

The Treeview Applet

The Treeview applet now supports assigning multiple drill-down links for each node in the diagram. The %DS2TREE macro arguments NRUL= and NACTION= now accept a semicolon-delimited list of URLs and menu text strings respectively as column variable values. The URLs are displayed when the user right-clicks a node and then selects **Custom Actions** from the pop-up menu.

SAS 9.4M7: The Treeview applet is deprecated. Existing programs that use this applet continue to work. However, this applet is no longer supported, and it might be removed in a future release.

SAS/GRAPH ActiveX Control

Starting with SAS 9.4M8, SAS/GRAPH ActiveX Control is not available from SAS. If you order SAS/GRAPH 9.4M8 or a later version, SAS/GRAPH ActiveX Control is not included in that order. If you have an existing installation of SAS/GRAPH ActiveX Control in your environment and plan to upgrade or migrate to SAS 9.4M8 or later, SAS recommends that you first uninstall SAS/GRAPH ActiveX Control. See “Uninstalling the SAS/GRAPH ActiveX Control” on page 200.

For more information about uninstalling retired products, see Unconfiguring and Uninstalling Retired Products in *SAS Guide to Software Updates and Product Changes*.

Global Statements

The global statements have the following changes and enhancements:

- SAS 9.4M2: The TITLE and FOOTNOTE statements now support the ALT=*"text-string"* option. This option enables you to specify descriptive text for the title or footnote. If you use ALT= in conjunction with the LINK= option, you can specify descriptive text for the URL to which the title or footnote links. The *"text-string"* can also contain occurrences of the variables named in a BY statement.
- SAS 9.4M3:
  - The new GraphTitle1Text style element controls and reduces the font size of the output of a TITLE1 statement in order to scale better with the graphs.
  - The SYMBOL statement's POINTLABEL COLOR= option now shows the sequential order of its color selection. The default color selection now aligns with the color specified for the axis label.

PART 1

Getting Started

1

Get Started Using SAS/GRAPH

What is SAS/GRAPH?

SAS/GRAPH is data visualization and presentation software. It creates and delivers accurate, high-impact visuals that enable decision makers to gain a quick understanding of critical business issues. As such, SAS/GRAPH does the following:

- organizes the presentation of your data and visually represents the relationship between data values as two- and three-dimensional graphs, including charts, plots, and maps.
- enhances the appearance of your output by enabling you to select text fonts, colors, patterns, and line styles, and control the size and position of many graphics elements.

- creates presentation graphics. SAS/GRAPH can create text slides, display several graphs at one time, combine graphs and text in one display, and create automated presentations.
- generates a variety of graphics output that you can display on your screen or in a web browser, store in catalogs, or review. You can send the graphics output to a hard copy graphics output device such as a laser printer.
- provides utility procedures and statements to manage the output.

Components of SAS/GRAPH Software

There are several components to SAS/GRAPH software:

SAS/GRAPH procedures
: enable you to create a variety of graphs, including bar charts, pie charts, scatter plots, surface plots, contour plots, and much more. The SAS/GRAPH procedures include the GAREABAR, GCHART, GPLOT, GBARLINE, GKPI, GCONTOUR, and G3D procedures, as well as others. These procedures use device drivers to generate output. SAS/GRAPH device drivers enable you to send output directly to your output device. Device drivers enable you to create output in a variety of formats such as PNG files and interactive ActiveX controls. This document, *SAS/GRAPH: Reference*, describes these procedures and how to use devices.

  SAS/GRAPH procedures enable you to create a variety of maps, including block, choropleth, prism, and surface maps. The GMAP mapping procedure is described in SAS/GRAPH and Base SAS: Mapping Reference.

  SAS/GRAPH procedures enable you to prepare map data sets as output data sets that are used as input to a procedure that creates a map. Prior to SAS 9.4M5, the SAS/GRAPH map data preparation procedures included GEOCODE, GINSIDE, GMAP, GPROJECT, GREDUCE, GREMOVE, and MAPIMPORT. Starting with SAS 9.4M5, the GEOCODE and MAPIMPORT procedures moved to Base SAS. Starting with SAS 9.4M6, all remaining mapping procedures except GMAP moved to Base SAS. Any procedure that moved to Base SAS can be run without requiring a licensed SAS/GRAPH installation. All of the mapping procedures are described in SAS/GRAPH and Base SAS: Mapping Reference.

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 device-based SAS/GRAPH procedures, including GCHART, GMAP, GPLOT, GBARLINE, GCONTOUR, and G3D, as well as others. Annotation data sets are described in Chapter 28, “Using Annotate Data Sets,” on page 721.

Network Visualization (NV) Workshop
: enables you to visualize and investigate the patterns and relationships hidden in network data (node-link data). Some common applications that use network data include supply chains, communication networks, websites, database schema, and software module dependencies. NV Workshop is designed for visualizing large networks. Using a combination of data tables, statistical graphs, and network graphs, NV Workshop enables you to extract information that would otherwise remain hidden. Help is available from the menu within the product.

Network Visualization Workshop runs in Windows operating environments only. The Network Visualization Workshop is described in *SAS/GRAPH: Network Visualization Workshop User's Guide*.

Note: Starting with SAS 9.4M5, SAS/GRAPH Network Visualization Workshop is discontinued and is no longer supported.

What You Can Create Using SAS/GRAPH

Charts and Plots

The follow table lists the charts and plots that you can create using SAS/GRAPH.

Table 1.1 *SAS/GRAPH Charts and Plots*

Category	Chart or Plot	
Charts	Area Bar Charts[1]	KPI Charts
	Bar Charts	Pie Charts
	Bar-line Charts	Radar Charts
	Block Charts	Star Charts
	Donut Charts	Tile Charts
2-D Plots	Bubble Plots	Regression Plots
	High-Low Plots	Scatter Plots
	Line Plots	Spline Plots
	Needle Plots	Step Plots
3-D Plots	Contour Plots	
	Scatter Plots	
	Surface Plots	

1 Supported in SAS 9.4M7 and earlier releases. Not supported in SAS 9.4M8 and later releases.

Multicell Graphs

SAS/GRAPH enables you to create graphs that contain multiple charts and plots. Each chart or plot occupies a single cell in the graph. You can specify a layout for the graph, which divides the graph area into individual cells, and then populate each cell with a chart or plot. SAS/GRAPH provides several predefined layouts that you can use, or you can create your own layout.

Interactive Graphs

SAS/GRAPH enables you to create graphs that display additional information when certain user actions occur. You can add data tips to your graphs, which display information when the mouse pointer is positioned on an element in your graph. The data tip can display data or additional details. You can create drill-down graphs, which link elements in your graph to resources such as other graphs, detailed descriptions, and so on. When a linked element in your graph is clicked, the linked resource opens in a new browser window. You can create multiple levels of drill-down graphs to enable your users to explore your data on their own. SAS/GRAPH also enables you to create animated graphs, which display graphs in a sequence in order to show trends in your data over time.

Maps

SAS/GRAPH enables you to create two- and three-dimensional maps that can show an area or represent values of response variables for subareas. The types of maps that you can create include block maps, choropleth maps, prism maps, and surface maps. You can use the map data sets that are provided with SAS or you can use your own data.

What You Need to Know to Get Started Using SAS/GRAPH

SAS/GRAPH Procedures

Procedures That Generate Single-Cell Plots

SAS/GRAPH provides procedures that you can use to create various single-cell charts and plots. The following table lists the procedure that you can use to generate each of the charts and plots.

Table 1.2 *Charts and Plots and the SAS/GRAPH Procedures and Statements That Generate Them*

Chart or Plot	Procedure	Statement
Area Bar Chart	GAREABAR[1]	GAREABAR procedure statement HBAR, HBAR3D, VBAR, or VBAR3D
Bar Chart	GCHART	GCHART procedure statement HBAR, HBAR3D, VBAR, or VBAR3D
Bar-line Chart	GBARLINE	GBARLINE procedure statements BAR for a vertical bar-line chart and PLOT for the line-plot overlay
Block Chart	GCHART	GCHART procedure statement BLOCK
Box Plot	GPLOT	GPLOT procedure statement PLOT Global statement SYMBOL with the INTERPOL=BOX option
Bubble Plot	GPLOT	GPLOT procedure BUBBLE statement
Contour Plot	GCONTOUR	GCONTOUR procedure statement PLOT
Donut Chart	GCHART	GCHART procedure statement DONUT

Chart or Plot	Procedure	Statement
High-Low Plot	GPLOT	GPLOT procedure statement PLOT Global statement SYMBOL with the INTERPOL=HILO option
KPI Chart	GKPI[2]	GKPI procedure statement DIAL, HBULLET, HSLIDER, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, or VSLIDER
Line Plot	GPLOT	GPLOT procedure statement PLOT Global statement SYMBOL with the INTERPOL=JOIN option
Map	GMAP	The GMAP procedure statements and the supporting data-preparation procedures GEOCODE, GINSIDE, GPROJECT, GREDUCE, GREMOVE, and MAPIMPORT are described in *SAS/GRAPH and Base SAS: Mapping Reference*.
Needle Plot	GPLOT	GPLOT procedure statement PLOT Global statement SYMBOL with the INTERPOL=NEEDLE option
Pie Chart	GCHART	GCHART procedure statement PIE or PIE3D
Radar Chart	GRADAR	GRADAR procedure CHART statement
Regression Plot	GPLOT	GPLOT procedure statement PLOT Global statement SYMBOL with the regression interpolation option
Scatter Plot (2-D)	GPLOT	GPLOT procedure statement PLOT
Scatter Plot (3-D)	G3D	G3D procedure statement SCATTER
Series Plot	GPLOT	GPLOT procedure statement PLOT Global statement SYMBOL with the INTERPOL=JOIN option
Spline Plot	GPLOT	GPLOT procedure statement PLOT Global statement SYMBOL with the INTERPOL=SPLINE option
Star Chart	GCHART	GCHART procedure statement STAR
Step Plot	GPLOT	GPLOT procedure statement PLOT Global statement SYMBOL with the INTERPOL=STEP option

Chart or Plot	Procedure	Statement
Surface Plot	G3D	G3D procedure statement PLOT
Tile Chart	GTILE[3]	GTILE procedure statement FLOW, TILE, or TOGGLE

1 Starting with SAS 9.4M8, because the ACTIVEX and ACTXIMG graphics devices are obsolete, the GAREABAR procedure is also obsolete. If you attempt to run the GAREABAR procedure, an error message is written to the SAS log.
2 Starting with SAS 9.4M8, the GKPI procedure no longer runs in the z/OS operating environment.
3 Starting with SAS 9.4M8, the GTILE procedure no longer runs in the z/OS operating environment.

An example of each of these plots is in Chapter 2, “Gallery of SAS/GRAPH Plots and Charts,” on page 15.

Procedure That Generates Multicell Graphs

SAS/GRAPH provides the GREPLAY procedure, which you can use to create various multicell graphs using a predefined layout or your own custom layout. The GREPLAY procedure also has other capabilities that you might find useful. The GREPLAY procedure is described in Chapter 44, “GREPLAY Procedure,” on page 1415.

Global Statements

A global statement is a statement that can be specified anywhere in your SAS program. It remains in effect until it is explicitly changed, overridden, canceled, or until the SAS session is terminated. TITLE, FOOTNOTE, and OPTIONS are examples of global statements. These and other global statements can be used with SAS/GRAPH. The global statements are described in “Overview of Global Statements” on page 397. SAS/GRAPH provides additional global statements that affect only the graphics output that is generated by the SAS/GRAPH procedures and by the SAS/GRAPH Annotate facility.

ODS Destinations and Styles

The SAS Output Delivery System (ODS) manages output from SAS, including SAS/GRAPH. ODS enables you to generate output in various document formats such as HTML, PDF, RTF, and so on. To generate output in ODS, you must open the ODS destination for the document format that you want. By default, SAS opens an ODS destination for HTML output. You can open other ODS destinations as desired. For each destination, the visual attributes of the output such as colors, fonts, plot markers, and so on, are controlled by an ODS style. Several styles are provided with SAS. Each destination has a default style, but you can specify any

one of the available ODS styles for your output. ODS is described in *SAS Output Delivery System: User's Guide*.

Graphics Options

SAS/GRAPH provides several graphics options that control various aspects of SAS/GRAPH output such as size, format, fonts, and so on. These options are listed in "Graphics Options and Device Parameters Dictionary" on page 577. The GOPTIONS statement sets one or more graphics options. The GOPTIONS statement is described in GOPTIONS Statement on page 430. The GOPTIONS procedure displays the current value of the graphics options. The GOPTIONS procedure is described in GOPTIONS Procedure on page 1249.

Output Devices

Graphics output in SAS/GRAPH is generated by a graphics device. A graphics device generates output in a specific format such as PNG, SVG, PDF, and so on. The device is specified with the DEVICE= graphics option in a GOPTIONS statement or with the DEVICE= system option in an OPTIONS statement. The SAS/GRAPH devices are described in "Using Graphics Devices" on page 89. A list of the commonly used graphics devices is in "Commonly Used Devices" on page 91.

By default, SAS/GRAPH output is determined by the ODS destination. The default output for the ODS destinations is described in "Default Devices for ODS Destinations" on page 93. There are some exceptions, which are indicated in this document, where applicable.

Learning By Example: Create Your First Chart Using SAS/GRAPH

About the Scenario in This Chapter

This chapter presents an example that demonstrates the basic steps that are required to generate a simple bar chart using SAS/GRAPH. When you have completed this example, you will know the basic steps that are needed to create a chart or plot using SAS/GRAPH. In this example, you will perform the following tasks:

- Use the GOPTIONS statement to set graphics options for your graph.
- Use ODS statements to specify an output document format and style for your output.

- Use SAS/GRAPH AXIS global statements to specify various attributes of your graph axes.
- Use global statements TITLE and FOOTNOTE to add a title and a footnote to your graph.
- Use the SAS/GRAPH GCHART procedure to generate a vertical bar chart.

Creating a Bar Chart Using SAS/GRAPH

Here is the SAS program for this example. The numbers inside the code comments identify the major steps in the program.

```
goptions border xpixels=460 ypixels=500;            /* 1 */
axis1 label=(angle=90 "Models Produced");           /* 2 */
axis2 label=("Vehicle Type");
title "Models Produced By Vehicle Type and Origin"; /* 3 */
footnote "Data: SASHELP.CARS";
proc gchart data=sashelp.cars;                      /* 4 */
   vbar3d type /                                    /* 5 */
      subgroup=origin
      type=freq freq
      shape=cylinder space=1
      axis=axis1 maxis=axis2 autoref clipref;
run;
quit;                                               /* 6 */
title;                                              /* 7 */
footnote;
```

Here is a description of each step in the program.

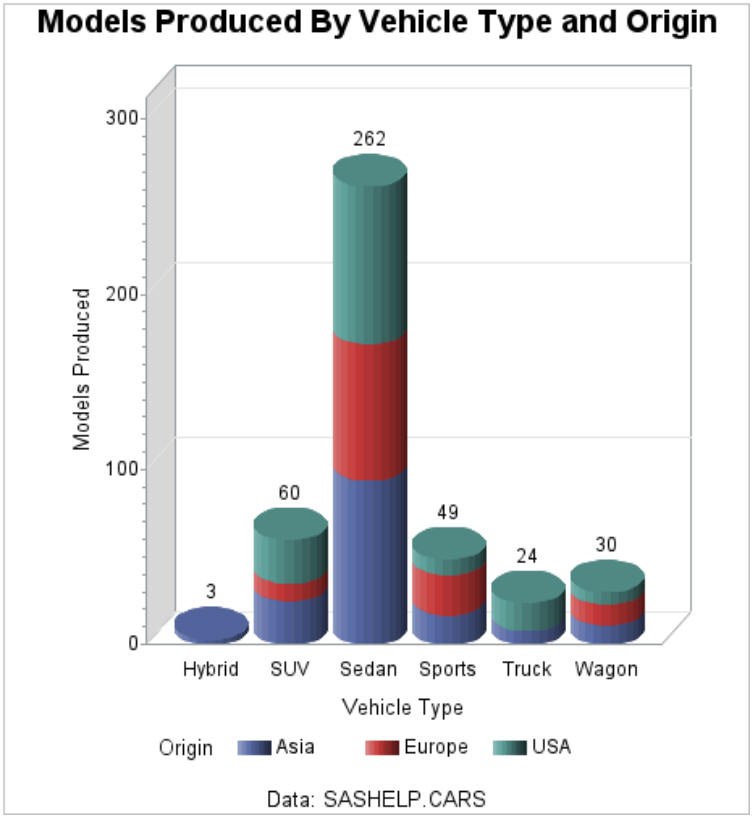
1. The GOPTIONS statement on page 430 specifies the graphics options. Option RESET=ALL sets all of the options to their default values. The XPIXELS=460 option specifies a width of 460 pixels, and the YPIXELS=500 specifies a height of 500 pixels for the graph.
2. The AXIS statements on page 399 specify the attributes of the chart axes.The AXIS1 statement specifies a label for the response axis. The ANGLE=90 option rotates the label 90 degrees to display it vertically along the response axis. By default, the label is displayed horizontally at the top of the axis. The AXIS2 statement specifies a label for the midpoint axis.
3. The TITLE global statement specifies the title for the graph, and the FOOTNOTE global statement specifies the footnote. See “TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement” on page 507.
4. The PROC GCHART statement on page 1001 starts the GCHART procedure step. The DATA= option specifies the input data set, Sashelp.Cars.
5. The VBAR3D statement generates a 3-D vertical bar chart. Variable Type is specified as the category variable, and variable Origin is specified as the subgroup variable. Option TYPE=FREQ specifies frequency as the summary statistic. Option FREQ displays the frequency at the top of each bar. The SHAPE=CYLINDER and SPACE=1 specify the bar shape and spacing. The AXIS=AXIS1 option assigns the AXIS1 statement to the response axis, and the MAXIS=AXIS2 assigns the AXIS2 statement to the midpoint axis. The

AUTOREF option draws reference lines at the major tick marks while the CLIPREF option clips the reference lines at the bars.

6 The QUIT statement terminates the GCHART procedure step.

7 The TITLE and FOOTNOTE global statements clear the title and footnote.

Here is the program output.



Using what you have learned from this example, you can begin exploring how to create other charts and plots using SAS/GRAPH.

Where to Go from Here

Now that you have a basic understanding of how to use SAS/GRAPH, review the remaining topics in this book to expand on your knowledge of the SAS/GRAPH product. The following table provides suggestions for your next steps.

Table 1.3 Next Steps

Task	Reference
If necessary, review the basic concepts of programming in the SAS language.	"Prerequisites" (p. xiii)

Task	Reference
Learn more about the basic elements of a SAS/GRAPH program.	Chapter 5, “Elements of a SAS/GRAPH Program,” (p. 45)
See other charts and plots that you can create using SAS/GRAPH.	Chapter 2, “Gallery of SAS/GRAPH Plots and Charts,” (p. 15)
Learn how to perform common tasks that are associated with developing SAS/GRAPH programs.	Chapter 3, “Common Tasks Associated with Developing SAS/GRAPH Programs,” (p. 35)
Identify resources that you can use to help you develop your SAS/GRAPH programs.	Chapter 4, “Additional Resources to Help You Develop Your SAS/GRAPH Programs,” (p. 39)

2

Gallery of SAS/GRAPH Plots and Charts

A Quick Look at the Gallery

SAS/GRAPH provides a wide variety of plots. This section lists the basic plots and charts that are available with SAS/GRAPH and provides a sample of each. The statement that generates each plot or chart is also listed.

This gallery contains a sample of each of the following chart and plot types:

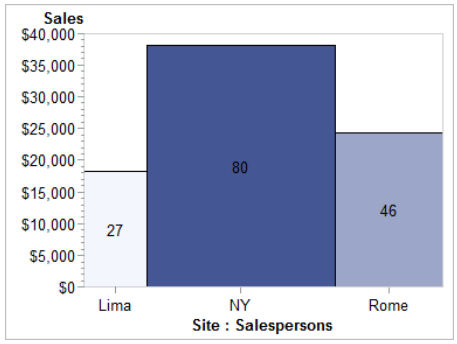
- Area Bar Charts
- Area Plot
- Bar Charts
- Bar-Line Charts
- Block Charts
- Box Plots
- Bubble Plots
- Contour Plots
- Donut Charts
- Geographical Maps
- High-Low Plots
- KPI Charts
- Line Plots
- Needle Plots
- Pie Charts
- Radar Charts
- Regression Plots
- Scatter Plots
- Scatter Plots, 3-D
- Series Plots
- Spline Plots
- Star Charts
- Step Plots
- Surface Plots
- Tile Charts

Gallery of Basic Charts and Plots

Area Bar Charts

An area bar chart is a bar chart in which the length or height of the bar represents the value of a chart statistic for each category of data. A second statistic is represented by the width of each bar. Here is a sample.

Note: Starting with SAS 9.4M8, area bar charts are no longer supported.

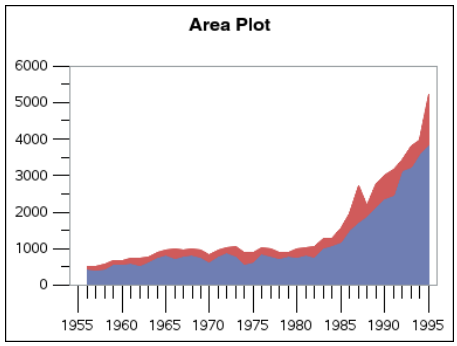


In SAS 9.4M7 and earlier releases, use the GAREABAR procedure VBAR statement to generate an area bar chart. The GAREABAR procedure syntax is described in Chapter 34, “GAREABAR Procedure,” on page 905. The code that generates this area bar chart is in “Example 1: Generating an Area Bar Chart” on page 913. For more examples, see:

- “Example 2: Generating an Area Bar Chart with a Numeric Chart Variable” on page 916
- “Example 3: Generating an Area Bar Chart with Subgroups” on page 918
- “Example 4: Area Bar Chart with Subgroups; Using the RSTAT= option and the WSTAT= option to Calculate Statistics as Percentages” on page 921

Area Plot

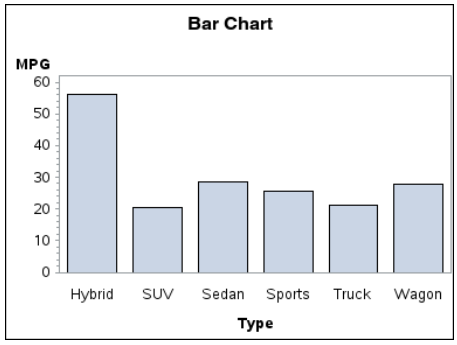
An area plot is a line, step, or spline plot where the area between the line and the category axis is filled with a color, pattern, or both. They are typically used to visualize quantitative data over time. Here is a sample.



Use the SAS/GRAPH GPLOT procedure PLOT statement with the AREAS= option and a SYMBOL statement that specifies JOIN, SPLINE, or STEP interpolation. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257. The code that generates this area plot is in “Example 7: Filling Areas in an Overlay Plot” on page 1355.

Bar Charts

Bar charts use vertical or horizontal bars to represent statistics based on the values of a category variable. You can also provide a response variable. Here is a sample.

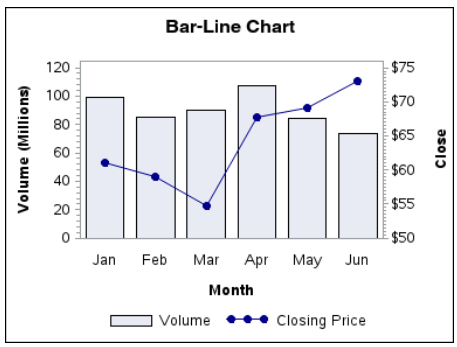


Use the GCHART procedure VBAR statement for a vertical bar chart or HBAR statement for a horizontal bar chart. The GCHART procedure syntax is described in Chapter 36, “GCHART Procedure,” on page 983. For examples, see:

- “Example 3: Specifying the Sum Statistic in Bar Charts” on page 1098
- “Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 1100
- “Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 1103
- “Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 1107

Bar-Line Charts

A bar-line chart is a vertical bar chart with one or more plot overlays. This chart graphically represents the value of a statistic calculated for one or more variables in an input SAS data set. Here is a sample.

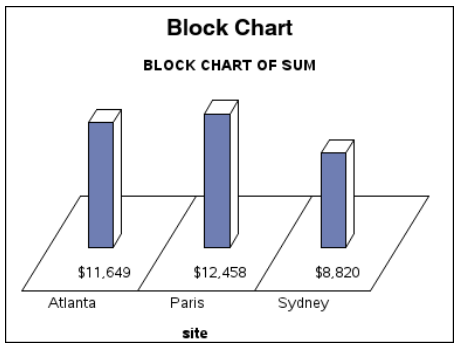


Use the GBARLINE procedure BAR and PLOT statements to generate a bar-line chart. The GBARLINE procedure syntax is described in Chapter 35, “GBARLINE Procedure,” on page 925. For examples, see:

- “Example 1: Producing a Basic Bar-Line Chart” on page 974
- “Example 2: Calculating Weighted Statistics” on page 976
- “Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs” on page 978

Block Charts

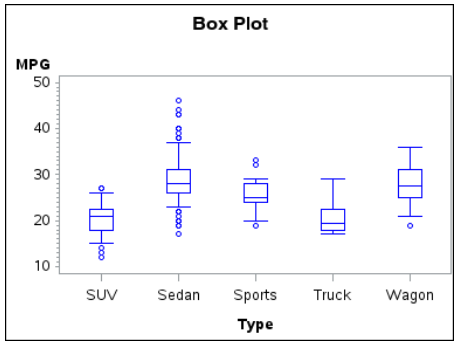
Block charts display the relative magnitude of data with blocks of varying height, each set in a square that represents a category of data (midpoint). Because block charts do not use axes, they are most useful when the relative magnitude of the blocks is more significant than the exact magnitude of any particular block. Here is a sample.



Use the GCHART procedure BLOCK statement to generate a block chart. The GCHART procedure syntax is described in Chapter 36, “GCHART Procedure,” on page 983. The code that generates this block chart is in “Example 1: Specifying the Sum Statistic in a Block Chart” on page 1093. For another example, see “Example 2: Grouping and Subgrouping a Block Chart” on page 1095.

Box Plots

A box plot summarizes the data and indicates the median, upper and lower quartiles, and minimum and maximum values. The plot provides a quick visual summary that easily shows center, spread, range, and any outliers. Here is a sample.

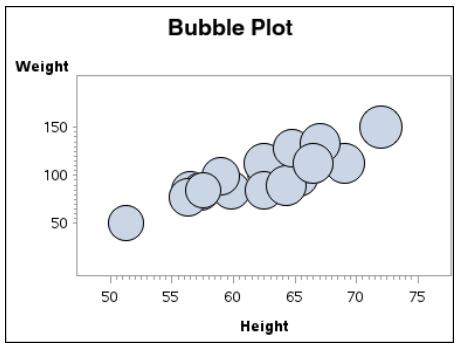


Use the GCHART procedure PLOT statement and the SYMBOL statement with the INTERPOL=BOX option to generate a box plot. A HILO interpolation can be used instead of the BOX interpolation. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257. The SYMBOL statement syntax is

described in “SYMBOL Statement” on page 470. For an example, see “Example 4: Creating and Modifying Box Plots” on page 538.

Bubble Plots

A bubble plot consists of X and Y values, which specify the location of the center of each bubble. The value of a third variable determines the size of the bubble. Here is a sample.

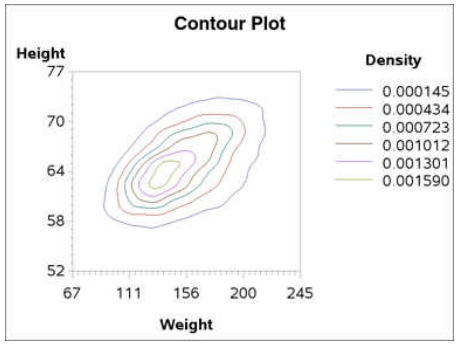


Use the GPLOT procedure BUBBLE statement to generate a bubble plot. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257. For examples, see:

- “Example 1: Generating a Simple Bubble Plot” on page 1337
- “Example 2: Labeling and Sizing Plot Bubbles” on page 1339
- “Example 3: Adding a Right Vertical Axis” on page 1342

Contour Plots

A contour plot represents a three-dimensional surface by plotting Z values on a two-dimensional format. The data provides the X, Y, and Z values. Here is a sample.

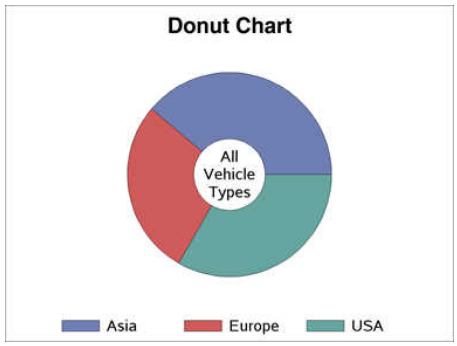


Use the GCONTOUR procedure PLOT statement to generate a countour plot. The GCONTOUR procedure syntax is described in Chapter 37, “GCONTOUR Procedure,” on page 1125. For examples, see:

- “Example 1: Simple Contour Plot” on page 1150
- “Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1152
- “Example 3: Specifying Contour Levels” on page 1155
- “Example 4: Using Patterns and Joins” on page 1157

Donut Charts

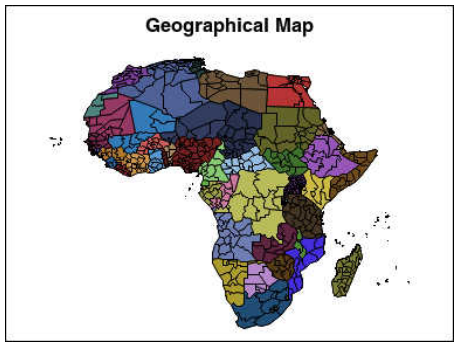
A donut chart is a variation of a pie chart, which represents the relative contribution of parts to the whole as wedge-shaped slices of a circle. Unlike a pie chart, a donut chart has a hole in the center in which you can add text. Here is a sample.



Use the GCHART procedure DONUT statement to generate a donut chart. The GCHART procedure syntax is described in Chapter 36, “GCHART Procedure,” on page 983. For an example, see “Example 8: Subgrouping a Donut or Pie Chart” on page 1112.

Geographical Maps

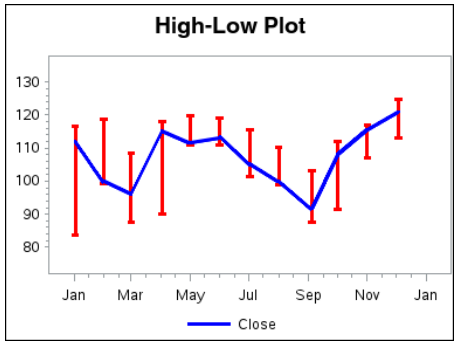
A geographical map displays values or attributes related to areas on the map. Map types include block, choropleth, prism, and surface. Maps can be generated by SAS/GRAPH. Here is a sample.



Use the GMAP procedure BLOCK, CHORO, PRISM, or SURFACE statement to generate a geographical map. See *SAS/GRAPH and Base SAS: Mapping Reference*.

High-Low Plots

Creates a display of floating vertical or horizontal lines or bars that represent high and low values for each value of a variable. You can show a tick mark at each mean Y value or you can join the mean values with a line or a bar. You can also show a top and bottom for each line. Here is a sample.



Use the GPLOT procedure PLOT statement and the SYMBOL statement with a HILO interpolation option to generate a high-low plot. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 470.

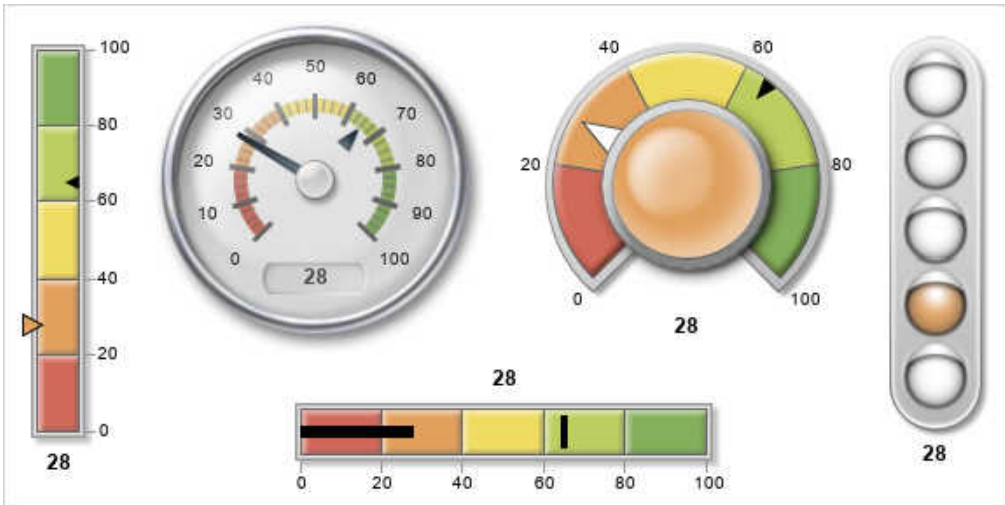
KPI Charts

Creates graphical key performance indicators (KPIs), which show metrics that help a business monitor its performance and measure its progress toward specific goals.

Note: Starting with SAS 9.4M8, KPI charts are no longer supported in the z/OS operating environment.

You can create five KPI chart types, as displayed left to right in the graphic following this list:

- vertical slider
- speedometer
- radial dial
- vertical traffic light
- horizontal bullet graph (bottom)

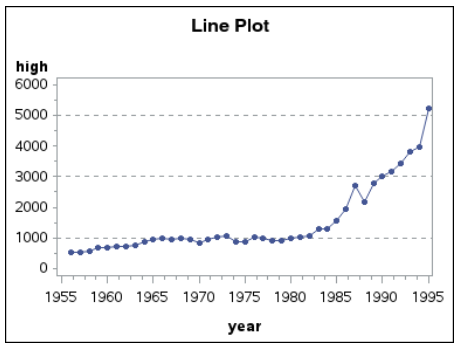


Use the GKPI procedure to create a KPI chart.The GKPI procedure syntax is described in Chapter 40, “GKPI Procedure,” on page 1221. For examples, see:

- “Example 1: Using the Default Colors as the Active Colors” on page 1240
- “Example 2: Creating a Gray Scale Bullet Graph” on page 1242
- “Example 3: Creating a Dial KPI Chart” on page 1243
- “Example 4: Defining a Speedometer” on page 1244
- “Example 5: Defining a Speedometer with Reversed Colors” on page 1246
- “Example 6: Creating a Traffic Light” on page 1247

Line Plots

A line plot shows the relationship of one variable to another, often as movements or trends in the data over a period of time. Typically, each variable value on the horizontal axis has only one corresponding value on the vertical axis. Here is a sample.



Use the GPLOT procedure PLOT statement and the SYMBOL statement with the INTERPOL=JOIN option to create a simple line chart.The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257 The SYMBOL statement syntax is described in “SYMBOL Statement” on page 470. The code that

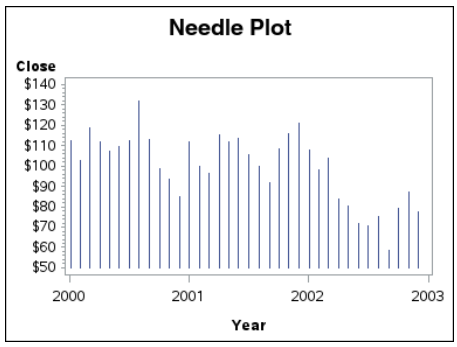
generates this line plot is in “Example 5: Connecting Plot Data Points” on page 1348. For more examples, see:

- “Example 6: Generating an Overlay Plot” on page 1351
- “Example 6: Generating an Overlay Plot” on page 1351
- “Example 8: Plotting Three Variables” on page 1358

TIP The line connecting data points can be smoothed using a variety of interpolation methods, including the Lagrange and the cubic spline interpolation methods. See “Spline Plots” on page 28.

Needle Plots

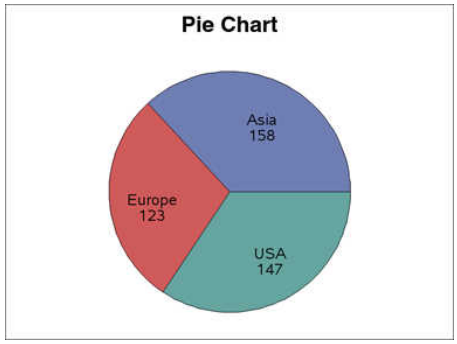
A needle plot uses vertical line segments, or needles, to connect each data point to a baseline. Here is a sample.



Use the GPLOT procedure PLOT statement and the SYMBOL statement with the NEEDLE interpolation option to generate a needle plot. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 470. For an example, see “Example 9: Plotting with Different Scales of Values” on page 1363.

Pie Charts

A pie chart represents input data as slices on the pie. Here is a sample.

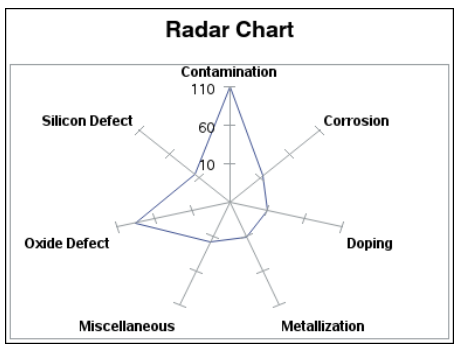


Use the GCHART procedure PIE statement to generate a pie chart. Use the PIE3D statement to generate a 3-D pie chart. The GCHART procedure syntax is described in Chapter 36, “GCHART Procedure,” on page 983. For examples, see:

- “Example 7: Specifying the Sum Statistic for a Pie Chart” on page 1109
- “Example 9: Ordering and Labeling Slices in a Pie Chart” on page 1115
- “Example 10: Grouping and Arranging Pie Charts” on page 1116
- “Example 13: Creating a Detail Pie Chart” on page 1123

Radar Charts

A radar chart shows the relative frequency of data measures in quality control or market research problems. The chart statistics are displayed along spokes that radiate from the center of the chart. Radar charts are often stacked on top of one another with reference circles, which resembles a radar screen. Radar charts are also referred to as star charts or spider charts, depending on the appearance of the chart and the variable types that the chart depicts. Here is a sample of a radar chart.



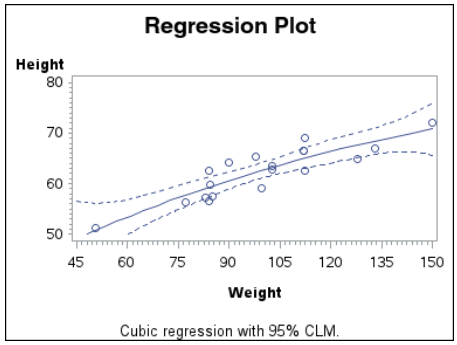
Use the GRADAR procedure CHART statement to generate a radar chart. The GRADAR procedure syntax is described in Chapter 43, “GRADAR Procedure,” on page 1375. The code that generates this radar chart is in “Example 1: Producing a Basic Radar Chart” on page 1400. For more examples, see:

- “Example 2: Overlaying Radar Charts” on page 1401
- “Example 3: Tiling Radar Charts” on page 1402
- “Example 4: Using Multiple Classification Variables in Radar Charts” on page 1403
- “Example 5: Modifying the Appearance of Radar Charts” on page 1405

- “Example 7: Modifying Chart Axes” on page 1408
- “Example 8: Creating a Wind Rose Chart” on page 1411
- “Example 9: Creating a Calendar Chart” on page 1412

Regression Plots

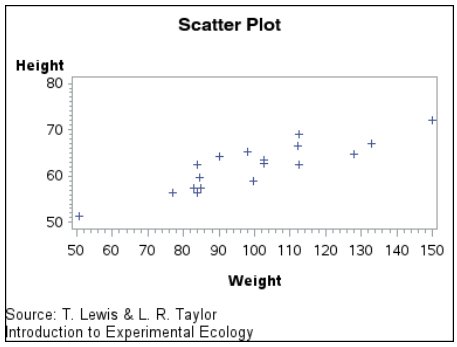
A regression plot includes a scatter plot of two numeric variables along with an overlaid linear or nonlinear fit line that enables you to perform a regression analysis. You can specify with the DEGREE option range of 1–10 a certain type of regression equation. Examples are linear (DEGREE=1), quadratic (DEGREE=2), or cubic (DEGREE=3). You can display confidence limits for mean predicted values or individual predicted values. Here is a sample.



Use the GPLOT procedure PLOT statement and the SYMBOL statement with a regression interpolation option to generate a regression plot. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 470. The code that generates this regression plot is in “Example 4: Plotting Two Variables” on page 1345.

Scatter Plots

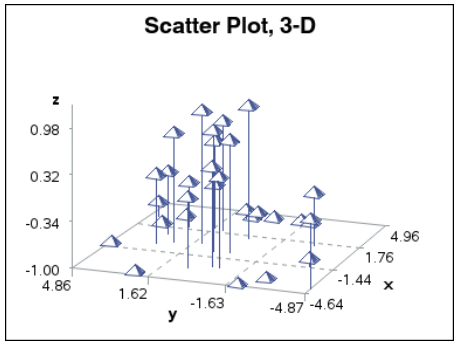
A scatter plot generates a marker for each observation that has nonmissing X and Y values. Markers can be symbols or character strings. Symbol markers can be labeled. Here is a sample.



Use the GPLOT procedure PLOT statement to generate a scatter plot. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257. The code that generates this scatter plot is in “Example 4: Plotting Two Variables” on page 1345.

Scatter Plots, 3-D

A 3-D scatter plot generates a marker for each observation that has nonmissing values for three variables. Markers can be symbols or character strings. Symbol markers can be labeled. Here is a sample.

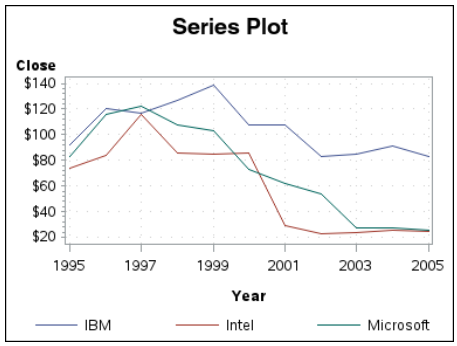


Use the G3D procedure SCATTER statement to generate a 3-D scatter plot.The G3D procedure syntax is described in Chapter 47, “G3D Procedure,” on page 1517. For examples, see:

- “Example 4: Generating a Scatter Plot” on page 1546
- “Example 5: Generating a Scatter Plot with Modified Shapes” on page 1547
- “Example 6: Generating a Scatter Plot with Modified Shapes and a Grid” on page 1549
- “Example 7: Generating a Rotated Scatter Plot with Modified Axes” on page 1551

Series Plots

A series plot, also referred to as a line plot, displays a series of line segments that connect observations of input data. Here is a sample.

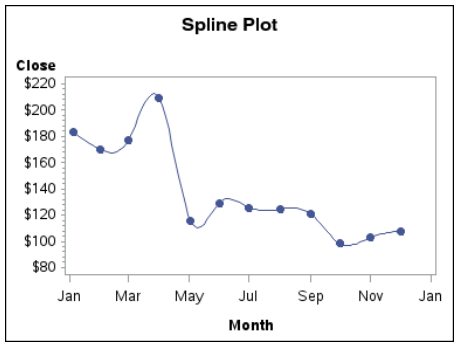


Use the GPLOT procedure PLOT statement and the SYMBOL statement with the JOIN interpolation option to generate a series plot. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 470. For examples, see:

- “Example 5: Connecting Plot Data Points” on page 1348
- “Example 6: Generating an Overlay Plot” on page 1351
- “Example 6: Generating an Overlay Plot” on page 1351
- “Example 8: Plotting Three Variables” on page 1358

Spline Plots

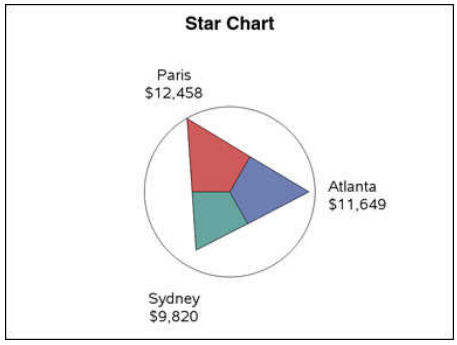
A spline plot is a line plot that uses a cubic spline method with continuous second derivatives to produce smooth curves. Here is a sample.



Use the GPLOT procedure PLOT statement and the SYMBOL statement with the INTERPOL=SPLINE option to create a spline plot. The GPLOT procedure syntax is described in Chapter 42, “GPLOT Procedure,” on page 1257 The SYMBOL statement syntax is described in “SYMBOL Statement” on page 470. For an example, see “Example 8: Plotting Three Variables” on page 1358.

Star Charts

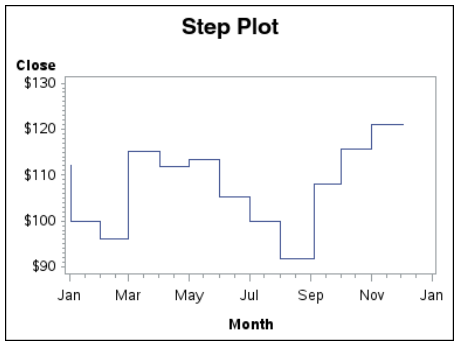
Star charts display data as lines (“spines”) radiating from the center of a circle toward the perimeter. Each spine represents a category of data (midpoint). The length of a spine represents the magnitude of the chart statistic for that midpoint starting at the center of the circle, which by default represents 0. The radius of the circle is the length of the longest spine (greatest statistic value) in the chart. Here is a sample.



Use the GCHART procedure STAR statement to generate a start chart.The GCHART procedure syntax is described in Chapter 36, “GCHART Procedure,” on page 983. The code that generates this start chart is in “Example 11: Specifying the Sum Statistic in a Star Chart” on page 1118. For another example, see “Example 12: Charting a Discrete Numeric Variable in a Star Chart” on page 1120.

Step Plots

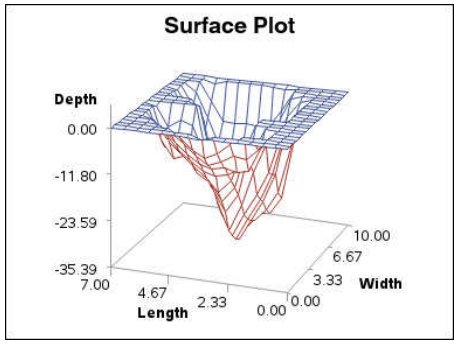
A step plot displays a series of horizontal and vertical line segments that connect observations of input data. The plots use a step function to connect the data points. The vertical line can change at each step. Here is a sample.



Use the GPLOT procedure PLOT statement and the SYMBOL statement with the STEP interpolation option to generate a step plot. The GPLOT procedure syntax is described in GPLOT Procedure on page 1257. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 470.

Surface Plots

A 3-D surface plot forms an evenly spaced grid of horizontal values (X and Y) and one or more vertical values (Z) for each combination. You can use the G3GRID procedure to interpolate the necessary values. The input data must be sorted by Y and X in order to obtain the correct lighting. Here is a sample.



Use the G3D procedure PLOT statement to generate a surface plot. The G3D procedure syntax is described in Chapter 47, “G3D Procedure,” on page 1517. The code that generates this surface plot is in “Example 1: Generating a Surface Plot” on page 1541. For more examples, see:

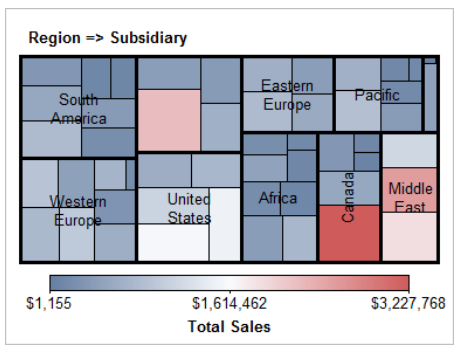
- “Example 2: Generating a Rotated Surface Plot” on page 1543
- “Example 3: Generating a Tilted Surface Plot” on page 1544

Tile Charts

A tile chart consist of rectangles or squares that are divided into tile-shaped segments.

Note: Starting with SAS 9.4M8, tile charts are no longer supported in the z/OS operating environment.

The chart is sometimes referred to as a rectangular tree map. The chart represents the relative sizes of the tiles to one another and to the whole. Here is a sample.



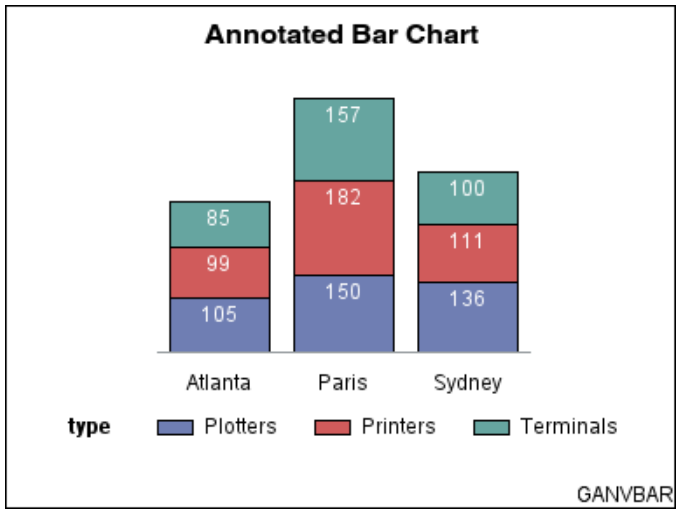
Use the GTILE procedure TILE statement to generate a tile chart.The GTILE procedure syntax is described in Chapter 46, “GTILE Procedure,” on page 1487. The code that generates this tile chart is in “Example 1: Simple GTILE with the COLORVAR= Option” on page 1509. For more examples, see:

- “Example 2: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range” on page 1511
- “Example 3: Specifying Discrete Tile Colors” on page 1513

Gallery of Additional Graph Features

Annotations

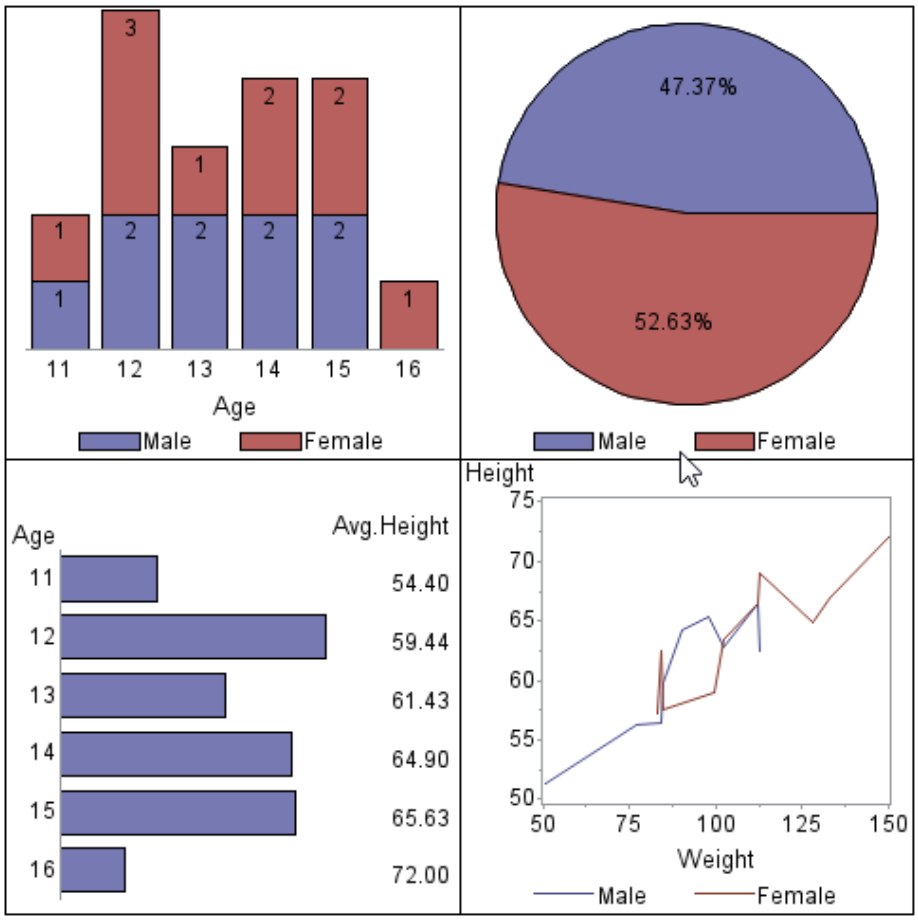
You can use the SAS/GRAPH Annotate facility to enhance your graphs with annotations. Here is a sample.



In this sample, labels are drawn inside the bar segments using text attributes that differ from the rest of the graph. The SAS/GRAPH Annotate facility is described in Chapter 28, “Using Annotate Data Sets,” on page 721. The code that generates this annotated bar chart is in “Example 1: Labeling Subgroups in a Vertical Bar Chart” on page 751. For another example, see “Example 2: Drawing a Circle of Stars” on page 754.

Multicell Graphs

You can use the SAS/GRAPH GREPLAY procedure to create multicell graphs. Here is a sample.

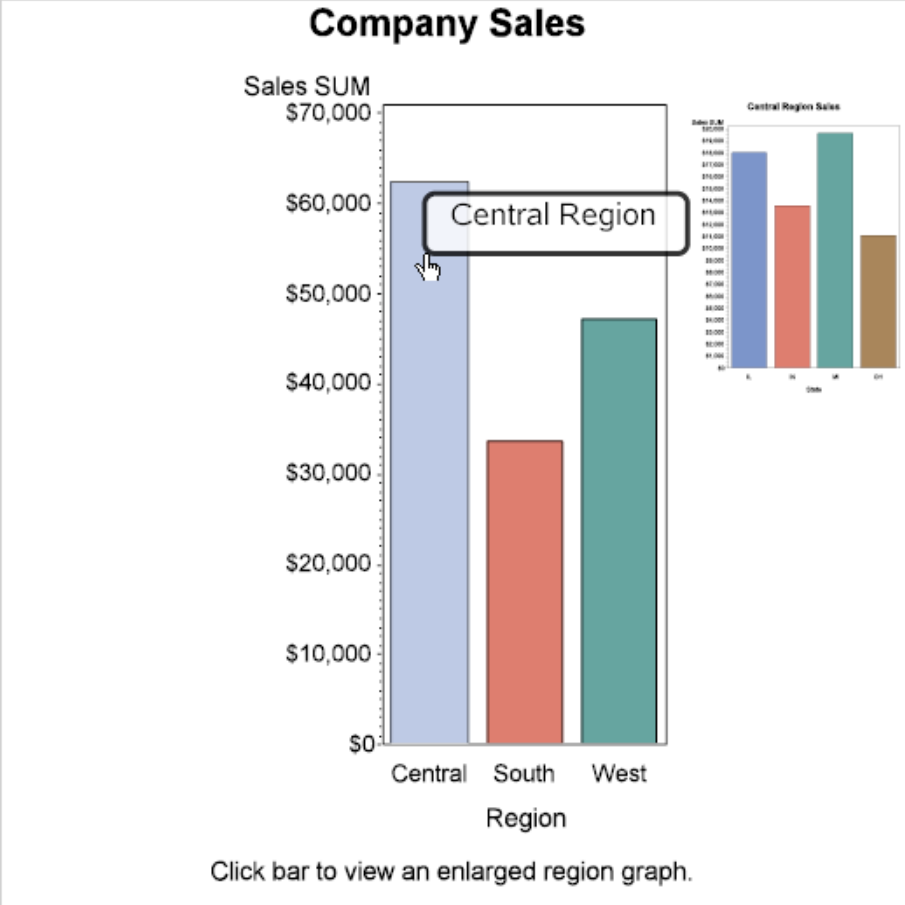


A layout template provides the layout for the multicell graph. Several predefined templates are provided. You can also create your own template. The GREPLAY procedure syntax and usage are described in Chapter 44, “GREPLAY Procedure,” on page 1415 and “Example 3: Replaying Graphs into a Template” on page 1464. The code that generates this multicell graph is in “Example 3: Replaying Graphs into a Template” on page 1464. For another example, see “Example 3: Replaying Graphs into a Template” on page 1464.

Interactive Graphs

Drill-Down Graphs

You can use SAS/GRAPH to create drill-down graphs that enable your users to explore your data. Elements in the graph such as bars, markers, and so on, can link to additional resources that show more detail. You can create multiple drill-down levels. When a user clicks a linked element in the graph, a new resource with more detail opens in the browser.



Drill-down graphs are described in Chapter 17, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” on page 249. The code that generates this drill-down graph is in “Example 3: Enhancing an SVG Drill-Down Presentation Using HTML Attributes” on page 273. For another example, see “Example 3: Enhancing an SVG Drill-Down Presentation Using HTML Attributes” on page 273.

Animated Graphs

You can use SAS/GRAPH to create animated graphs in the GIF or SVG format. An animated graph shows a sequence of graphs automatically. It is useful for showing how data changes over time. You can control the amount of time that each graph is displayed and how the graphs transition. Animated graphs are described in Chapter 16, “Generating Animations,” on page 227. For examples, see “Examples: Generating Animated Images” on page 238.

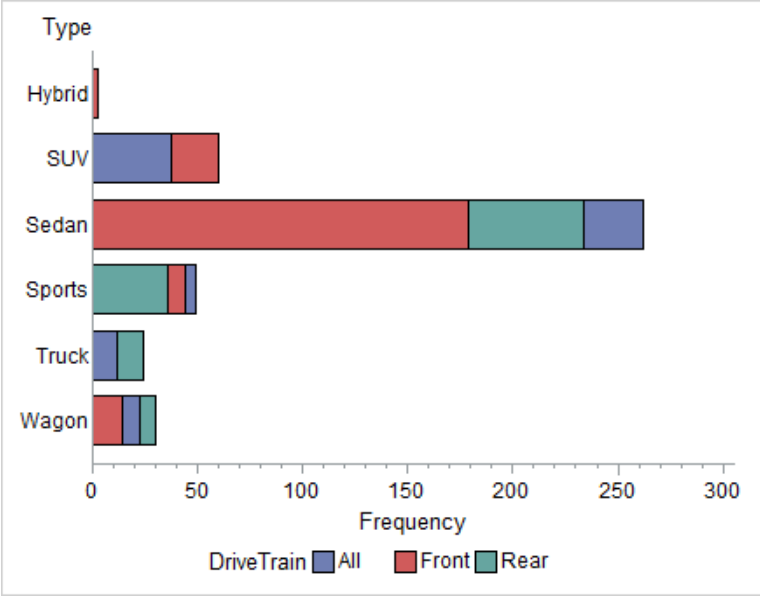
ActiveX Graphs

In SAS 9.4M7 and earlier releases, SAS/GRAPH provides an ActiveX control that enables you to create interactive graphs where ActiveX controls are supported. The ACTIVEX graphics device generates the SAS/GRAPH ActiveX Control, which

displays your graph, and embeds it in an HTML document or in an RTF document. You can embed your ACTIVEX output in Microsoft documents such as Microsoft Word and Microsoft PowerPoint. ActiveX graphs provide pop-up menus that enable your users to change various attributes of the graph dynamically.

Note: Starting with SAS 9.4M8, ActiveX graphs are no longer supported.

Figure 2.1 *SAS/GRAPH ActiveX Control Horizontal Bar Chart*



ActiveX graphs are described in Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

TIP You can use the ACTXIMG graphics device to generate PNG images of ActiveX graphs for use where the ActiveX control is not supported or is not installed.

3

Common Tasks Associated with Developing SAS/GRAPH Programs

About the SAS/GRAPH Tasks

This chapter lists common tasks that are associated with developing SAS/GRAPH programs. For each task, one or more links are provided to information about how to complete the task. Because there are many tasks associated with SAS/GRAPH, the tasks are categorized as follows:

- Output environment
- Graph creation
- Graph enhancement

The task lists in this chapter can help you quickly locate information in this reference that you need to create your graphs.

Output Environment Tasks

The following table lists tasks that are related to establishing the output environment for SAS/GRAPH.

Task	Reference
Learn about the SAS/GRAPH output environment.	Chapter 8, “The Graphics Output Environment,” on page 79 Chapter 11, “SAS/GRAPH Output,” on page 119
Learn about the SAS/GRAPH graphics devices.	“Using Graphics Devices” (p. 89)
Create a custom SAS/GRAPH graphics device.	“Creating Universal Printers and Shortcut Devices” on page 103 “Creating a Custom Device” on page 105
Specify the graph size.	“Setting the Size of Your Graph” on page 129
Specify the image resolution.	“Setting the Resolution of Your Graph” on page 130
Use ODS to specify the output document format.	Chapter 10, “Managing Your Graphics with ODS,” (p. 111)

Graph Creation Tasks

The following table lists tasks that are related to generating graphs using SAS/GRAPH.

Task	Reference
Determine the SAS/GRAPH language elements that you need to use to create your graph.	Chapter 2, “Gallery of SAS/GRAPH Plots and Charts,” (p. 15)
Create your SAS/GRAPH program.	Chapter 5, “Elements of a SAS/GRAPH Program,” (p. 45)

Task	Reference
Add titles, footnotes, and notes to your graph.	"TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement" (p. 507)
Specify fonts in your SAS/GRAPH program.	Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," (p. 349)
Specify colors in your SAS/GRAPH program.	Chapter 23, "Using Colors in SAS/GRAPH Programs," (p. 363)
Create multiple graphs in a single procedure step.	Chapter 7, "Using Run-Group Processing," on page 75 "BY Statement" on page 424
Generate a graph in the SVG format.	Chapter 13, "Using SVG Graphics," (p. 159)
Generate a graph in a bitmap format.	Chapter 14, "Generating SVG, PNG, GIF, and TIFF Graphics," (p. 175)

Graph Enhancement Tasks

The following table lists tasks that are related to adding enhancements to your graphs.

Task	Reference
Add data tips and a description to your graph.	Chapter 17, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," (p. 249)
Add one or more images to your graph.	Chapter 24, "Adding Images to SAS/GRAPH Output," (p. 383)
Annotate your graph.	Chapter 23, "Using Colors in SAS/GRAPH Programs," (p. 363)
Animate your graph.	Chapter 16, "Generating Animations," (p. 227)
Add drill-down links to your graph.	Chapter 17, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," (p. 249)

4

Additional Resources to Help You Develop Your SAS/GRAPH Programs

SAS/GRAPH Sample Programs

The documentation for each procedure, for global statements, and for features such as the Annotate facility provide examples that demonstrate these features of SAS/GRAPH. You can copy the example code from the Help or from the SAS website and paste it into the Program Editor in your SAS session.

In the SAS windowing environment, many of these samples are also included in the SAS Sample Library. How you access the code in the sample library depends on how it is installed at your site.

- In most operating environments, you can access the sample code through the SAS Help and Documentation. Select **Help** ⇨ **SAS Help and Documentation**. On the **Contents** tab, select **Learning to Use SAS** ⇨ **Sample SAS Programs** ⇨ **SAS/GRAPH Samples**.
- In other operating environments, the SAS Sample Library might be installed in your file system. Ask your on-site SAS support personnel where it is located on your system.

The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This document includes the name of each sample file that it references (for example, GBLWTSTA or GCHBRMID). The naming convention for SAS/GRAPH samples is G*pcxxxxx*, where *pc* is the product code and *xxxxx* is an abbreviation of the example title. The product code can be a code for a procedure, a statement, or in the case of Java and ActiveX examples, WB for "web graphs." To illustrate, the code for an example in the GCHART procedure chapter, "Example 7: Specifying the Sum Statistic for a Pie Chart" on page 1109, is stored in sample member GCHPISUM.

- In the SAS Sample Library, the sample programs are organized by product. Within each product category, most of the samples are sorted by procedure. Thus, to access the code for an example in the GCHART procedure chapter, select **Learning to Use SAS** ⇨ **SAS/GRAPH Samples**, scroll to **GCHART Procedure**, and select **GCHPISUM-Sum Statistic for a Pie Chart**.
- In your file system, the files that contain the sample code have file names that match the sample member names. For example, in a directory-based system, the code for sample member GCHPISUM is located in a file named GCHPISUM.SAS.

Note: For Java and ActiveX (web graph) samples, the naming convention is GWB*xxxxx*.

Table 4.1 *Product Codes for SAS/GRAPH Procedures*

Procedure	Product Code
ganno	AN
gareabar[1]	AB
gbarline	BL
gchart	CH
gcontour	CT
gfont	FO
gkpi[2]	KP
goptions	OP
gplot	PL
gradar	RR
greplay	RE
gslide	SL
gtile[3]	TL

Procedure	Product Code
g3d	TD
g3grid	TG

1 Starting with SAS 9.4M8, because the ACTIVEX and ACTXIMG graphics devices are obsolete, the GAREABAR procedure is also obsolete. If you attempt to run the GAREABAR procedure, an error message is written to the SAS log.
2 Starting with SAS 9.4M8, the GKPI procedure no longer runs in the z/OS operating environment.
3 Starting with SAS 9.4M8, the GTILE procedure no longer runs in the z/OS operating environment.

Table 4.2 *Product Codes for SAS/GRAPH Statements*

Statement	Product Code
axis	AX
by	BY
footnote	FO
goptions	ON
legend	LG
note	NO
pattern	PN
symbol	SY
title	TI

Examples and Resources on the Web

The SAS website contains a large number of examples that can help you visualize and code your graphs. The examples cover a range of SAS technologies including the SAS/GRAPH procedures.

- The SAS Training Post is a blog that provides tutorials, tips, and practical information about SAS. Dr. Robert Allison contributed a great deal of SAS/GRAPH and ODS Graphics data visualization examples to this blog during his tenure at SAS. His blog is no longer active, but it provides an extensive set of very useful examples.

  https://blogs.sas.com/content/sastraining/author/robertallison/

- The SAS Knowledge Base contains an abundance of searchable samples and SAS Notes. You can browse by topic, search for a particular note or a particular technology such as the name of a procedure, and conduct other searches.

  https://support.sas.com/en/knowledge-base.html

  Note: The SAS Knowledge Base content is currently available only in English.

- The Graphics Samples Output Gallery in the SAS Knowledge Base is a collection of graphs organized by SAS procedure. The graphs link to the source code. The gallery is maintained by SAS Technical Support.

  https://support.sas.com/en/knowledge-base/graph-samples-gallery.html

- The Focus Area Graphics site provides a simple interface to business and analytical graphs. The site is maintained by the SAS Data Visualization team.

  https://support.sas.com/rnd/datavisualization/index.htm

- You can share your questions, suggestions, and experiences related to graphics on the Graphics Programming community site.

  https://communities.sas.com/t5/Graphics-Programming/bd-p/sas_graph.

In addition, SAS offers instructor-led training and self-paced e-learning courses to help you get started with platform graphics software. The SAS Training site provides information about the courses that are available.

sas.com/training.

Books

For a list of books that might help you develop your SAS/GRAPH programs, see “Recommended Reading” on page xx.

PART 2

SAS/GRAPH Concepts

5

Elements of a SAS/GRAPH Program

Overview

The elements used by SAS/GRAPH programs can include SAS/GRAPH language elements, ODS statements, and Base SAS language elements. The purpose of this chapter is to familiarize you with the overall structure of a typical SAS/GRAPH program. This chapter defines a program's basic parts and shows how these parts relate to one another.

A Typical SAS/GRAPH Program

Most SAS/GRAPH programs have Base SAS statements, ODS statements, and SAS/GRAPH statements. Annotate DATA steps are also used in many SAS/GRAPH programs. The sample program below identifies the basic parts of a typical SAS/GRAPH program. Each element is described in more detail in the following sections.

```
/* The following is an ODS destination statement
that defines the output file name and location, and changes the style
from the default of HTMLBlue to Analysis
*/
ods html file="c:\regression.htm"      1
    style=analysis;                      11
/* The following four statements are Global statements */
goptions reset=all device=png;         2
title "Study of Height vs. Weight";      3
footnote j=r h=2 "Data: SASHELP.CLASS";  3
symbol interpol=rcclm95 value=circle;    4
/* The next five lines represent the PROC step */
proc gplot data=sashelp.class;           5
    plot height*weight  /                6
         haxis=45 to 155 by 10;          10
run;                                     7
quit;                                    8
/* The following is an ODS destination statement.
   It resets the default style. */
ods html close;                          9
```

1. For information about ODS destinations, see “Destination Statements” on page 50.
2. For information about the GOPTIONS statement, see “Global Statements” on page 48.
3. For information about the TITLE and FOOTNOTE statements, see “Global Statements” on page 48.
4. For information about the SYMBOL statement, see “Global Statements” on page 48.
5. For information about the procedure statement, see “Procedure Statement” on page 47.
6. For information about the plot (action) statement, see “ Subordinate Statement” on page 47.
7. For information about the RUN statement, see “Shared Language Statements” on page 51.
8. For information about the QUIT statement, see “Shared Language Statements” on page 51.
9. For information about specifying the style for an ODS destination, see “Destination Statements” on page 50.

10 For information about the HAXIS option, see "Other Statements and Options" on page 47.

11 For information about ODS styles, see "ODS Statement Options" on page 51.

SAS/GRAPH PROC Step

A group of SAS procedure statements is called a PROC step. The PROC step consists of all the statements, variables, and options that are contained within the (beginning) PROC 5 and (ending) RUN 7 statements of a procedure. These statements can identify and analyze the data in SAS data sets. They can generate the graphics output and control the appearance of the output. Statements can define variables and perform other operations on your data. You can also specify global statements and options within the PROC step to customize the appearance of your graph. However, it is often more efficient to specify global statements before the PROC step.

Procedure Statement

The procedure statement 5 identifies which procedure you are invoking (for example, GPLOT, GCHART, or GCONTOUR) and identifies which input data set is to be used.

Subordinate Statement

Subordinate statements 6 are statements used within the procedure that perform the work of the procedure. Subordinate statements that generate graphs are called *action statements*. At least one action statement is required for a procedure to produce a graph. Examples of action statements are the HBAR statement in the GCHART procedure and the BUBBLE statement in the GPLOT procedure.

Non-action statements are those that do not generate graphs. The GRID statement in PROC G3GRID and the DELETE statement in PROC GDEVICE are examples of non-action statements.

Other Statements and Options

There are many options 10 that you can specify within the PROC step to control your graphics output. PROC step options always follow the forward slash (/) following the action statement of the procedure. These options might control such things as axis characteristics, midpoint values, statistics, catalog entry descriptions, or appearance elements of your graph. An example is the SUBGROUP= option in

the BLOCK statement of the GCHART procedure. The option tells the procedure to divide the graph's bars into segments according to the values of the SUBGROUP= variable. The HAXIS option in the PLOT statement of the GPLOT procedure, as shown in "A Typical SAS/GRAPH Program" on page 46, specifies where to draw the major tick mark values for the horizontal axis.

Global Statements

A *global statement* is a statement that you can specify anywhere in a SAS program. A global statement sets values and attributes for all the output created after that global statement is specified in the program. The specifications in a global statement are not confined to the output generated by any one procedure. However, they do apply to all the output generated thereafter in the program, unless they are overridden by a procedure option or another global statement. The RESET= option in the GOPTIONS statement also overrides global statements by resetting them.

Below is a list of all the SAS/GRAPH statements along with their brief descriptions. See "SAS/GRAPH Statements" on page 397 for a more detailed description of each of these statements.

AXIS
: modifies the appearance, position, and range of values of axes in charts and plots.

BY
: processes data and orders output according to the values of a classification (BY) variable. The BY statement in SAS/GRAPH is essentially the same as the BY statement in Base SAS, but the effect on the output is different when it is used with SAS/GRAPH procedures. When used with SAS/GRAPH procedures, the BY statement subsets the data and creates a graph for each unique value of the BY-variable.

Note: The BY statement is an exception here because it is not a global statement. It must be specified within a DATA or PROC step.

GOPTIONS 2
: specifies graphics options that control the appearance of graphics elements by specifying characteristics such as default colors, fill patterns, fonts, or text height. Graphics options can also temporarily change device settings.

LEGEND
: modifies the appearance and position of legends generated by procedures that produce charts, plots, and maps.

PATTERN
: defines the characteristics of patterns used in graphs created by the GAREABAR, GBARLINE, GCHART, GCONTOUR, GMAP, and GPLOT procedures.

Note: See *SAS/GRAPH and Base SAS: Mapping Reference* for GMAP procedure examples that use the PATTERN global statement.

SYMBOL 4
: defines the characteristics of symbols that display the data plotted by a PLOT statement used by PROC GBARLINE, PROC GCONTOUR, and PROC GPLOT as well the interpolation method for plot data. The SYMBOL statement also controls the appearance of lines in contour plots.

TITLE, NOTE, and FOOTNOTE 3
: add text to maps, plots, charts, and text slides. They control the content, appearance, and placement of text on your graph. The TITLE statement is used to specify up to ten title lines to be printed on the title area of the output. The NOTE statement is used to add text to the procedure output area of your graph. The FOOTNOTE statement is used to display lines of text at the bottom of the page.

Note: The NOTE statement is a local statement. It can be specified only within a PROC step, and it affects the output of that PROC step only.

Annotate DATA Set

An Annotate DATA set is a data set containing graphics commands that can be applied to SAS/GRAPH output. See Chapter 28, “Using Annotate Data Sets,” on page 721 for information about building and using Annotate data sets. The Annotate facility can be used to create a completely new graph or to annotate existing PROC output. See “Annotate Function Dictionary” on page 759 and “Annotate Variable Dictionary” on page 803 for a complete description of all Annotate functions and variables. Below is an example of how the Annotate facility can be used to annotate numbers on bars in a graph that was created using the GCHART procedure.

Figure 5.1 *Using Annotate with GCHART Procedure Output*

ODS Statements

Destination Statements

Like Base SAS, SAS/GRAPH uses ODS destination statements (1, 9) to control where the output goes and how it looks. Although ODS statements are not required in every SAS/GRAPH program, they are necessary if you want to generate graphs for destinations other than the default HTML destination. Some other destinations include LISTING, RTF, and PDF. For more information about ODS destinations, see "Understanding ODS Destinations" in *SAS Output Delivery System: User's Guide* in *SAS Output Delivery System: User's Guide*

As shown in "A Typical SAS/GRAPH Program" on page 46, the ODS destination statement is used at the beginning and end of the program to modify the default destination. The beginning statement modifies the default path and filename of the output file as well as the default style. The end statement sets the HTML style back to its default of HTMLBlue. The ODS destination statement can also be used to open a different destination. If you do choose to use a destination other than the default and need to use the ODS destination statement, you should always open the destination before calling the procedure. Opening a non-default destination results in output being sent both to HTML by default as well as to the additional specified destination. Conserve system resources by using the ODS destination statement at the end of the SAS program to close a destination that was opened in that program.

ODS Statement Options

You can use the STYLE= option 11 on the ODS destination statement to change the style that is applied to your output. Starting with SAS 9.4, the default style is HTMLBlue. For more information about the STYLE= option, see Chapter 21, "Using ODS Styles, Device Parameters, and Options," on page 321.

Base SAS Language Elements

Shared Language Statements

The following Base SAS language statements are also part of SAS/GRAPH:

FORMAT statement
: assigns a format to a variable. SAS/GRAPH procedures use formatted values to determine such aspects of the graph as midpoints, axis labels, tick-mark values, and legend entries. For a complete description of this statement, see *SAS DATA Step Statements: Reference*.

FILENAME
: associates a SAS fileref with an external file or output device. See "FILENAME Statement" on page 52 for a more detailed description of this statement. For a complete description of this statement, see *SAS Global Statements: Reference*.

RUN statement 7
: executes the statements in the PROC step. For a complete description of this statement, see *SAS Global Statements: Reference*.

LABEL statement
: assigns a descriptive text string (a "label") to a variable. The label appears in place of the variable name on the axis and legend. For a complete description of this statement, see *SAS DATA Step Statements: Reference*.

LIBNAME
: associates a libref with a SAS library. See "LIBNAME Statement" on page 52 for a more detailed description of this statement. For a complete description of this statement, see *SAS Global Statements: Reference*.

ODS statements
: control the output of SAS/GRAPH procedures, where the output is sent (destination), the appearance of the output (STYLE=), and the output file type (DEVICE=). Starting with SAS 9.4, the default style is HTMLBlue.See Chapter 6, "Overview of Devices, ODS Destinations, and ODS Styles," on page 55 for information about using ODS with SAS/GRAPH procedures.

OPTIONS statement
: changes the value of one or more SAS system options. For a complete description of this statement, see *SAS Global Statements: Reference*.

QUIT statement 8
: executes any statements that have not executed and ends the procedure. It also ends a procedure that is using RUN-GROUP processing.

WHERE statement
: specifies observations from SAS data sets that meet a particular condition. You can use a WHERE statement to easily subset your data. For a complete description of this statement, see *SAS DATA Step Statements: Reference*.

FILENAME Statement

The FILENAME statement associates a SAS fileref with an external file or output device. With SAS/GRAPH software, you can use a FILENAME statement to do the following tasks:

- point to a text file that you want to use for data input or output.
- assign the destination of a graphics stream file (GSF). This destination can be either a single, specific file or an aggregate file storage location, such as directory or PDS. See "Exporting Your Output" on page 152 for information about creating graphics stream files.

You can also use the FILENAME statement to route input to and from other devices. For details, see the SAS documentation for your operating environment.

A FILENAME statement that points to an external file has this general form:

FILENAME *fileref* '*external-file*';

fileref
: is any SAS name.

external-file
: is the physical name of the external file or aggregate file storage location that you want to reference. For details about specifying the physical names of external files, see the SAS documentation for your operating environment.

LIBNAME Statement

The LIBNAME statement associates a libref with a SAS library. A SAS library can be either temporary or permanent. Typically, SAS libraries used with SAS/GRAPH contain the following items:

- SAS files for data input and output.
- SAS catalogs that contain SAS/GIS maps, fonts, GRSEG, CMAP, TEMPLATE, or device entries.
- SAS catalogs that contain graphics output. These catalogs are often stored in permanent libraries. See "Controlling Where Your Output Is Stored" on page 133 for information about storing graphics output in a permanent catalog.

The LIBNAME statement has this general form:

LIBNAME *libref* '*SAS-library*';

libref
is any SAS name.

SAS-library
is the physical name for the SAS library on your host system. For details about specifying *SAS-library*, see the SAS documentation for your operating environment.

The libref WORK is reserved; it always points to an area where temporary data sets and catalogs are kept. The contents of WORK are deleted when you exit a SAS session.

Other Resources

- For more information about using and managing SAS/GRAPH programs to create graphics output, see Chapter 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 55.
- For more information about generating SAS/GRAPH output for the web, see Chapter 14, “Generating SVG, PNG, GIF, and TIFF Graphics,” on page 175 and Chapter 16, “Generating Animations,” on page 227.
- For information about using and managing SAS/GRAPH output, see Chapter 11, “SAS/GRAPH Output,” on page 119.

6

Overview of Devices, ODS Destinations, and ODS Styles

Introduction to Devices, ODS Destinations, and ODS Styles

The output from SAS/GRAPH procedures is controlled by the following:

SAS/GRAPH device
: SAS/GRAPH uses device drivers to generate graphics output. SAS/GRAPH device drivers control the format of your graphics. For example, devices determine whether SAS/GRAPH produces a PNG file, an SVG file, or an ActiveX control. You can specify a device in the OPTIONS statement, the SAS/GRAPH GOPTIONS statement, or the ODS destination statement.

Note: For information about the ActiveX control, see Chapter 15, "Generating Interactive ActiveX Graphics," on page 191.

ODS destination
: Like other SAS procedures, the output from SAS/GRAPH procedures is controlled by ODS (Output Delivery System). An ODS destination controls where your output is sent, which could be to an HTML file, a PDF file, to the GRAPH window, directly to a printer, and so on. The ODS destination is specified by the ODS destination statement.

ODS style
: By default, ODS also applies a style to your output. The ODS style controls the overall appearance of your output, including colors and fonts. You can specify a style with the STYLE= option in the ODS destination statement.

Note: This document deals only with device-based graphics. For information about template-based graphics (also called ODS Graphics), see *SAS Graph Template Language: User's Guide*, *SAS Graph Template Language: Reference*, and *SAS ODS Graphics: Procedures Guide*.

Each ODS destination is associated with a default ODS style and a default SAS/GRAPH device to optimize your output for that destination. However, you can change the default values in order to customize your graphics output. Combining ODS statements with SAS/GRAPH statements and options enables you to control all of the aspects of your output. This includes where your output is sent, its appearance, and the format of your graphics.

Note: The LISTING destination is unique. For the LISTING destination, the SAS/GRAPH device controls not only the graphics format but where your output is sent.

The following sections discuss these concepts of SAS output and describe how you can use SAS/GRAPH and ODS statements and options to create the graphic output that you want.

For complete information about ODS, see also *SAS Output Delivery System: User's Guide*.

Working with ODS Destinations and Styles

ODS destinations determine where your SAS/GRAPH output is sent. For example, the HTML destination sends output to an HTML file (by default), and the LISTING destination sends output to the GRAPH window. By default, ODS styles determine the overall appearance of your output

Opening and Closing Destinations

A *destination* is a designation that ODS uses to determine where to send your output. Valid destinations include HTML (by default) , LISTING (the GRAPH window), RTF, and PDF, but other destinations are also available. Depending on the options available for the destination, you can specify options such as the filename or the path to an output directory.

To generate output, a valid ODS destination must be open. By default, the HTML destination is open, but you can open other destinations as needed by specifying an ODS destination statement. Your output is sent to all destinations that are open, so you can create, for example, HTML and PDF output by submitting your code only once. With the exceptions of the HTML and LISTING destinations, you must also close the destination before output is generated.

Here is the general form of the statements that open and close an ODS destination.

```
ods destination <options>;    /* opens the destination */
    /* procedure statements and other program elements here */
ods destination close;    /* closes the destination */
```

For example, to send output to the LISTING destination, you would specify the following:

```
ods listing;
    /* procedure statements and other program elements here */
ods listing close;
```

For more information about ODS destinations, see “Managing ODS Destinations” on page 111 and “ODS Destination Statement Options” on page 113.

The LISTING Destination

The LISTING destination is somewhat different from the other ODS destinations. For the LISTING destination, if you do not specify a SAS/GRAPH device, then your output is sent to the GRAPH window. However, if you specify a device, then the device determines where your output is sent in addition to determining the graphics format. For example, the PNG device sends output to a PNG file instead of the GRAPH window. Your company might have device drivers specific to your site that

send output directly to a certain printer. Where your output is sent is controlled by the device entry in the **SASHELP.DEVICES** catalog. For more information about devices, see “Controlling the Graphics Output Format with the DEVICE= Option” on page 69 and “Using Graphics Devices” on page 89.

The LISTING destination is not open by default. You must open the LISTING destination at the beginning of your SAS program if you want to send output to it. You should close the LISTING destination at the end of your program to conserve resources. See “Closing Destinations to Save System Resources” on page 73 for more information.

The LISTING destination does not have to be closed before output can be generated.

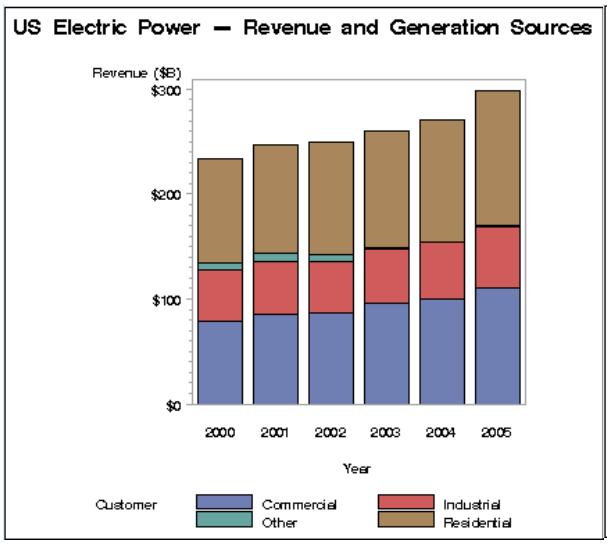
Introduction to Styles

By default, ODS applies a *style* to all output. A style is a template, or set of instructions, that determines the colors, font face, font sizes, and other presentation aspects of your output. SAS ships many predefined styles in the STYLES item store, such as Analysis, Statistical, and Journal. Examples of some of these predefined styles are shown in Table 6.1 on page 58. Many additional styles (see “Viewing the List of Styles Provided by SAS” on page 333) are available in the STYLES item store in SASHELP.TMPLMST.

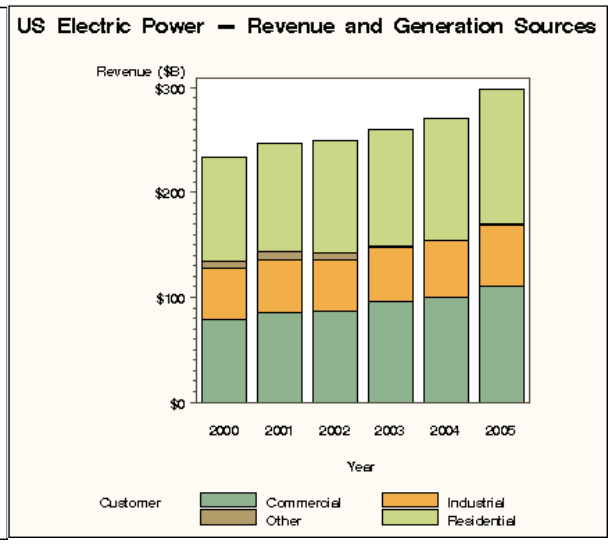
Each destination has a default style that is associated with it. For example, the default style for the PDF destination is Pearl, and the default style for the HTML destination is HTMLBlue. See “ODS Destinations and Default Styles” on page 324 and “Recommended Styles” on page 324 for more information.

Table 6.1 *Examples of Styles Available in SASHELP.TMPLMST*

`Style=Statistical`

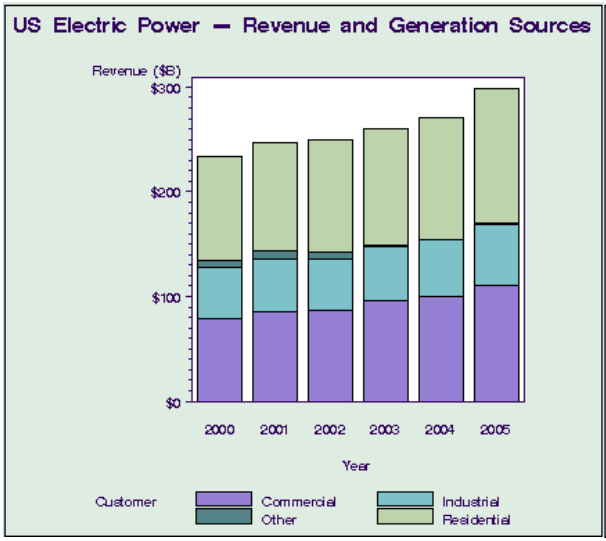


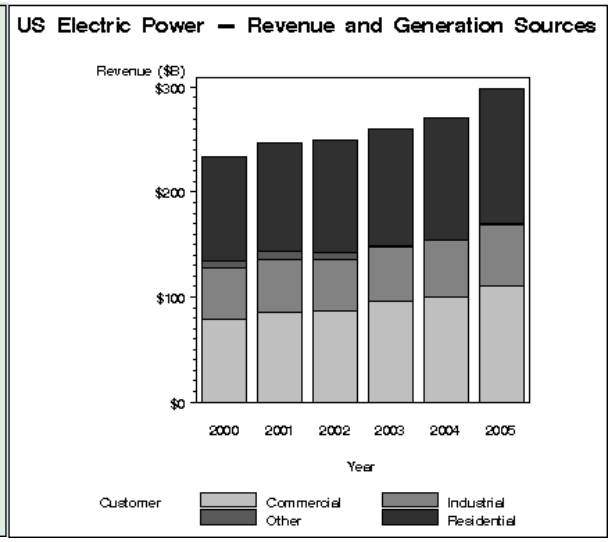
`Style=Analysis`



`Style=Ocean`

`Style=Journal`





Specifying a Style

To change the style that is applied to your output, specify the STYLE= option on your ODS destination statement. For example, suppose you want to change the overall look of your graph for the HTML destination to the Analysis style. Do this by specifying `style=analysis` in the ODS HTML destination statement as follows:

```
ods html style=analysis;
```

See “About Style Templates” on page 323 and “Specifying a Style” on page 328 for more information.

Note: You can turn off the use of styles by default by specifying the NOGSTYLE option. See “Changing the Appearance of Output to Match That of Earlier SAS Releases” on page 347 and the “GSTYLE System Option” on page 717 for more information.

Generating Output with SAS/GRAPH Procedures

Overview of Generating Output

ODS provides many destinations to which you can send output. Some of the most often used destinations are HTML (a web page), LISTING, RTF (Rich Text Format), and PDF (Portable Document Format). The topics that follow show the default output for each of these destinations.

Each ODS destination is associated with a default style and a default SAS/GRAPH device to optimize your output for that destination. For more information, see “Introduction to Devices, ODS Destinations, and ODS Styles” on page 55.

The rules change when you are running SAS without a terminal. In this case you must either specify the DEVICE= option in the GOPTIONS statement or specify the DEVICE= or PRINTER= option in the ODS destination statement. A terminal is not associated with your SAS session if you are running in batch mode on z/OS. This is also true if you are running SAS on UNIX without an X Server or if you have specified the NOTERMINAL system option. See the global option “DEVICE Graphics Option” on page 608, and “Using Graphics Devices” on page 89 for more information.

Output in the SAS Windowing Environment

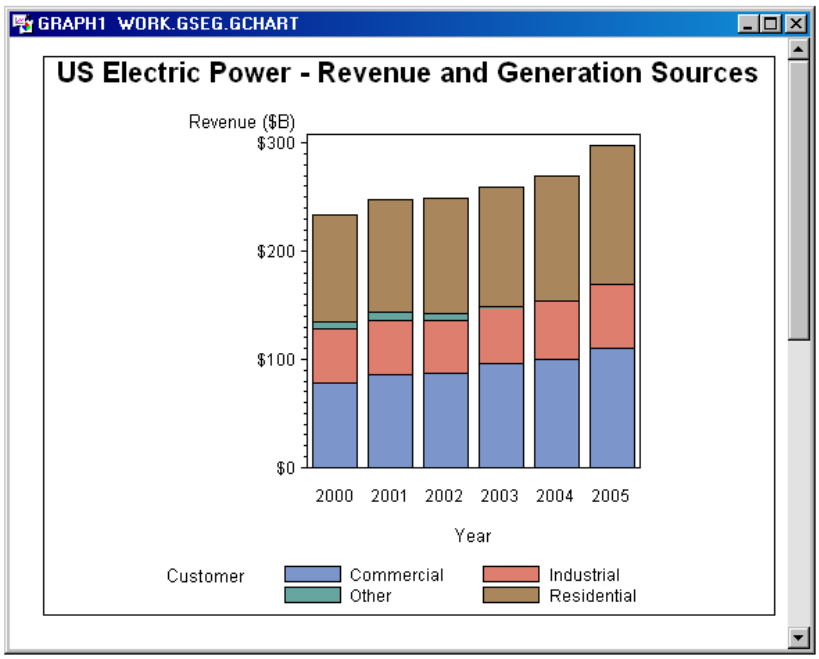
Sending Output to the GRAPH Window (LISTING Destination)

When working in an interactive environment such as Windows, the default LISTING destination is the GRAPH window. Use the ODS destination statement at the beginning and end of the SAS program to open and close the LISTING destination, respectively. The following example is a simple GCHART program that produces the output shown in Figure 6.1 on page 61.

```
ods html close;
ods listing;
goptions reset=all border hsize=6 vsize=4.5;
title "US Electric Power - Revenue and Generation Sources";
proc gchart data=sashelp.electric (where=(year >= 2000)) ;
  vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
```

```
quit;
ods listing close;
```

Figure 6.1 *LISTING Destination Output Using the LISTING Style (Shown in the GRAPH Window)*



The default style applied to output sent to the LISTING destination is LISTING in a Display Manager Session and in batch mode. When you send output to the LISTING destination, SAS/GRAPH uses a default device driver that generates output for the GRAPH window. This device driver does not write an image file to disk.[1] For the LISTING destination, the default device driver varies by operating environment. In a Display Manager Session, the default device driver on Windows systems is WIN. On UNIX systems, the default device driver is XCOLOR, and on z/OS systems, the default device driver is IBMPCGX.

Sending Output to a File

To send output to disk file, send your output to the ODS LISTING destination, but specify a graphics output device using the DEVICE= graphics option. You can use a FILENAME statement and the GSFNAME= graphics option to specify a name and location for the graphics output file.

To create a GIF file with the graph shown in Figure 6.1 on page 61, in the procedure code, add a FILENAME statement to create a file reference to the desired output file. Then, add the DEVICE=GIF and GSFNAME=*FileRef* graphics options to the GOPTIONS statement, where *FileRef* is the file reference that you created in the FILENAME statement.

```
ods html close;
```

1. SAS/GRAPH procedures create GRSEG catalog entries when you send output to the LISTING destination, but the GRSEG file format is an internal file format specific to SAS/GRAPH. It cannot be used as if it were an image file such as a PNG or GIF file.

```
ods listing;
goptions reset=all border hsize=6 vsize=4.5;
filename gout "./revgensrcs.gif";
goptions device=gif gsfname=gout;
title  "US Electric Power - Revenue and Generation Sources";

proc gchart data=sashelp.electric (where=(year >= 2000)) ;
   vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;

ods listing close;
```

The LISTING style is applied to the graph as shown in Figure 6.1 on page 61. In the FILENAME statement, the current directory is the default SAS output directory.

For more information about sending graphics output to a file, see “Controlling Where Your Output Is Stored” on page 133 .

Sending Output to a Web Page

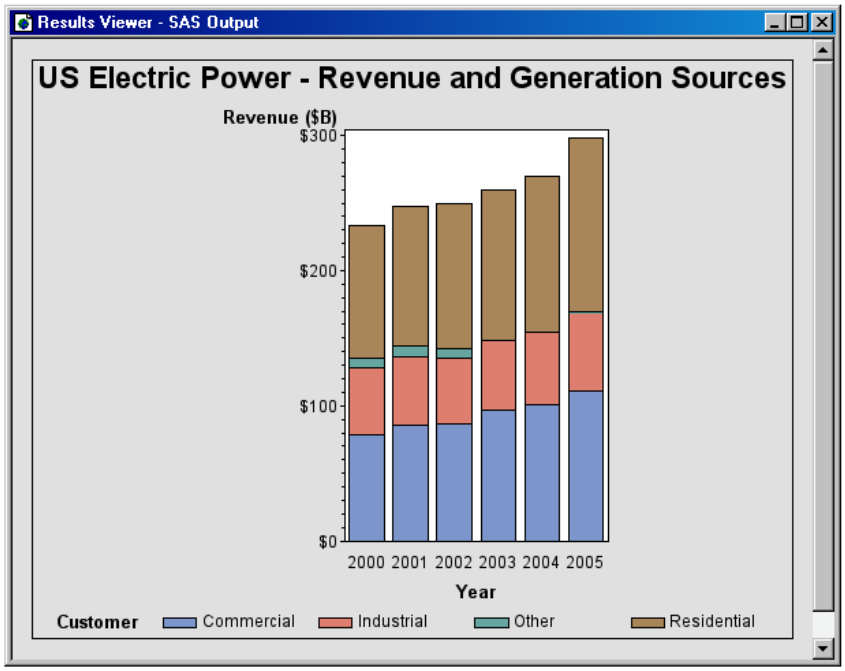
To send output to a web page, you simply submit your SAS program that sends your output to the HTML destination, which by default is open. No ODS destination statements need to be specified.

To specify web page output with a style other than the default style of HTMLBlue, you need to specify the ODS HTML destination statement with the STYLE= option. To restore to the default style at the end of the program, specify an ODS HTML destination statement with the STYLE= option to indicate HTMLBlue.

To create a web page using the style of Default with the graph shown in Figure 6.1 on page 61, add the ODS HTML statements around the procedure code. The first style= option overrides the default style of HTMLBlue. The second ODS statement resets that style. Submit the following SAS program:

```
ods html style=default;
goptions reset=all border;
title  "US Electric Power - Revenue and Generation Sources";

proc gchart data=sashelp.electric (where=(year >= 2000)) ;
   vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;

ods html close;
```

Figure 6.2 *HTML Destination Output Using the Default Style (Styles.Default)*



By default, SAS/GRAPH creates a PNG file that contains the graph and an HTML page that references the PNG file. You can use the BODY= and PATH= options in the ODS HTML statement to specify a specific filename and location for the HTML and PNG files. SAS/GRAPH displays the HTML page in the Results Viewer. You can also view the graph outside of your SAS session by displaying the HTML page in your browser. The default device driver is PNG, and the default style is HTMLBlue.

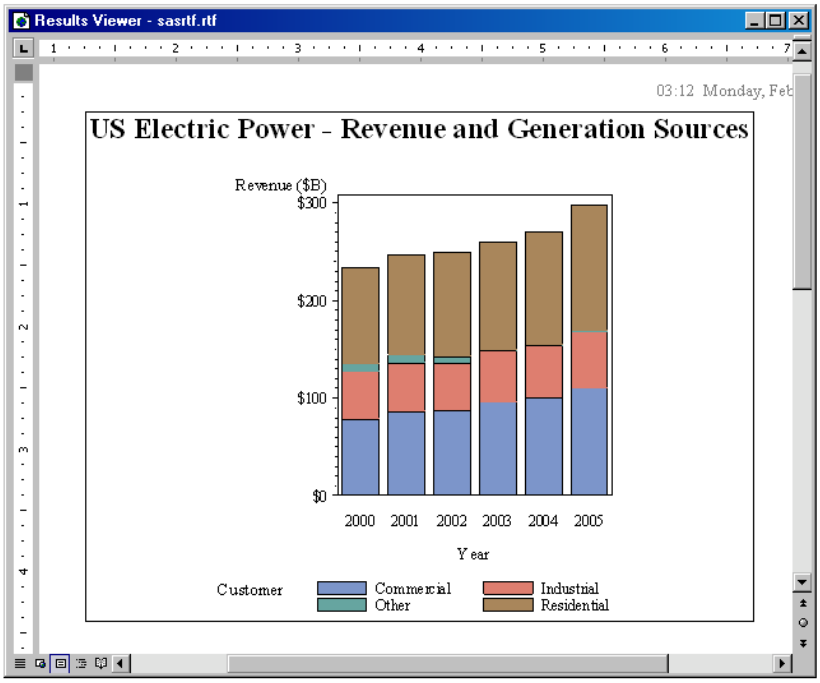
Sending Output to an RTF File

To send output to an RTF file, send your output to the RTF destination by specifying the ODS RTF statement. This statement opens the RTF destination so that it can receive output. You must also close the RTF destination before output can be generated.

To create an RTF document that contains the graph shown in Figure 6.1 on page 61, add the ODS RTF statements around the procedure code and submit the following SAS program. To conserve system resources, close the default HTML destination with another ODS destination statement.

```
ods html close;
ods rtf;
goptions reset=all border;
title  "US Electric Power - Revenue and Generation Sources";

proc gchart data=sashelp.electric (where=(year >= 2000)) ;
   vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;
ods rtf close;
```

Figure 6.3 RTF Output Using the RTF Style



By default, SAS/GRAPH creates an RTF file with the graph embedded in it and, in some cases, displays the RTF file in a viewer. When you send output to the RTF destination, SAS/GRAPH does not write a separate image file to disk. The default device driver is the SASEMF driver, and the default style is RTF.

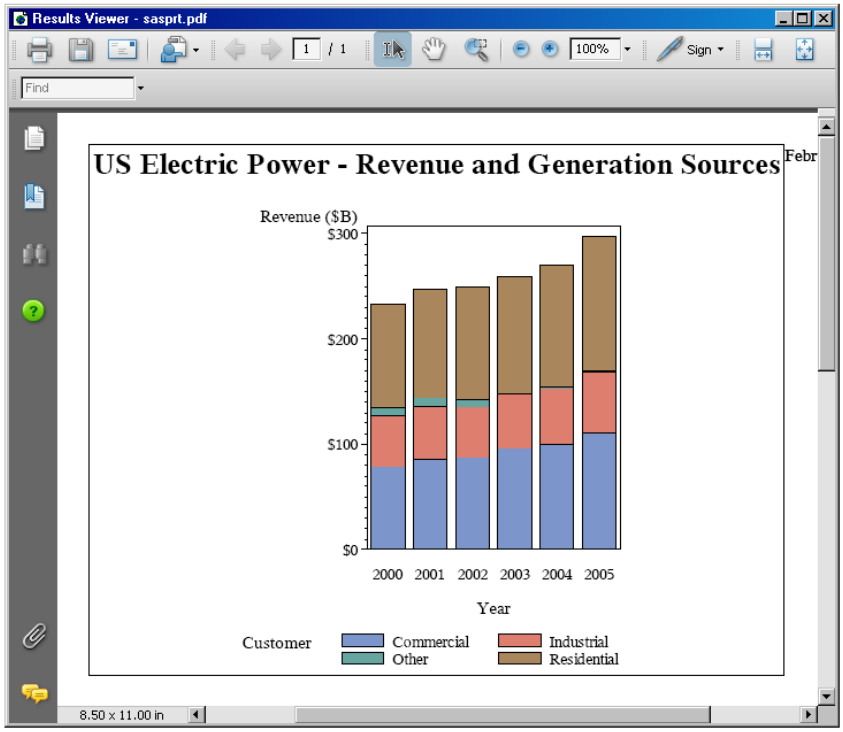
Sending Output to a PDF File

To send output to a PDF file, send your output to the PDF destination by specifying the ODS PDF statement. This statement opens the PDF destination so that it can receive output. You must also close the PDF destination before output can be generated.

To create a PDF document that contains the graph shown in Figure 6.1 on page 61, add the ODS PDF statements around the procedure code and submit the following SAS program:

```
ods html close;
ods pdf;
goptions reset=all border;
title  "US Electric Power - Revenue and Generation Sources";

proc gchart data=sashelp.electric (where=(year >= 2000)) ;
   vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;
ods pdf close;
```

Figure 6.4 PDF Output Using the Printer Style



By default, SAS/GRAPH creates a PDF file and, in some cases, displays the PDF file in a PDF viewer. When you send output to the PDF destination, SAS/GRAPH does not write a separate image file to disk. The default device driver is the SASPRTC device driver, and the default style applied to output sent to the PDF destination is Pearl.

Output in SAS Studio

About SAS Studio

SAS Studio is a web interface to the SAS system. Using SAS Studio, you can access your SAS resources such as data, libraries, and existing programs, and you can write new programs. You can use SAS Studio to access SAS installed on your local machine, on a remote server in your local environment, or on a hosted server in a cloud environment.

Note: SAS/GRAPH is not included in SAS University Edition.

To see which edition of SAS Studio you are using, select **About SAS Studio** in the SAS Studio Help menu. see *SAS Studio: User's Guide* for your version of SAS Studio. You can find the SAS Studio documentation on the SAS® Studio Learn & Support page on support.sas.com.

The SAS Studio Output Environment

SAS Studio provides a default output environment for your SAS programs. Using the SAS Studio default output environment, you can do the following:

- Generate HTML5, PDF, and RTF output by default, and view the HTML5 output in the **RESULTS** tab.
- Download your generated output using the download buttons for the three default output types.
- Change the default output style for each destination using the Preferences window.

For information about results in SAS Studio, see *SAS Studio: User's Guide* for your version of SAS Studio. You can find the documentation for all versions of SAS Studio on the SAS Studio documentation page on support.sas.com.

Viewing the Default HTML5 Results

In SAS Studio, by default, output is generated in the HTML5, PDF, and RTF formats. The HTML5 results are displayed in the **RESULTS** tab. The PDF and RTF results are generated but are not displayed. If you do not want the PDF and RTF results, use the Preferences window in SAS Studio to deselect the results them. You can also use the Preferences window to change the style for your results to any of the ODS styles that are delivered with SAS.

Downloading Default HTML5, PDF, and RTF Results from SAS Studio

If you want to save results from SAS Studio, you can download your results in the default output formats from the SAS Studio **RESULTS** tab. The **RESULTS** tab toolbar provides a separate download button for the HTML5, PDF, and RTF results. To download your results, click the icon for the desired format, and then specify a location for the output file.

Customizing the SAS Studio Output Environment

The default output environment in SAS Studio supports most SAS programs when HTML5, PDF, or RTF results are desired. However, you must customize the SAS Studio output environment when you need to do any of the following:

- generate results for another output destination, such as HTML, PowerPoint, LISTING, and so on

- send your results directly to another location
- use a custom style such as a corporate style for your results
- use an image format other than the default for your graphics
- create a drill-down graph
- create an animated graph

To customize the SAS Studio output environment, you should first disable the default output environment in order to conserve system resources. Next, establish your own output environment, and then execute the SAS statements that are required to generate your output. Use ODS statements, ODS Procedures, or ODS options in your SAS program to define the environment that you need. As a best practice, if your SAS program requires a customized output environment in SAS Studio, your program should always perform these steps:

1. Create a file reference for your ODS output. You can use the &_SASWS_ macro variable that is defined in SAS Studio to reference your home directory as shown in the following statement:

   ```
   filename odsout "&_SASWS_/charts";
   ```

   If you want to store your image files in a separate directory, create a second file reference for your image files as shown in the following statement:

   ```
   filename odsiout "&_SASWS_/charts/images";
   ```

   Note: The directories that you specify must already exist, and you must have Write access to the directories.

2. To conserve system resources, disable the default output environment using the following statement:

   ```
   ods _all_ close;
   ```

3. Open the desired ODS destination. Use the PATH= option to specify the file reference that you created for your ODS output. If you created a separate file reference for your image files, use the GPATH= option to specify the image output file reference. Here is an example.

   ```
   ods html path=odsout gpath=odsiout file="saleschart.html";
   ```

4. Execute the SAS statements that are required to generate your output.

5. Close your ODS destination.

When you disable the default SAS Studio output environment, results are no longer displayed in the **RESULTS** tab for the duration of your program. The results are generated only by the ODS destination that you open.

Generate Output for Other Output Destinations

If you need to generate results other than the default HTML5, PDF, or RTF results in SAS Studio, you must open your own ODS destination. Examples of output destinations include HTML, PowerPoint, and LISTING. After you disable the default

output environment, use an ODS statement to open your own output destination. Here is an example.

```
filename odsout "&_SASWS_/charts";
ods _all_ close;
ods powerpoint path=odsout file="filename";
```

For information about ODS destination statements, see *SAS Output Delivery System: User's Guide*.

Send Your Results to Another Location

When you execute a program in the SAS Studio default output environment, you can download the output from the **RESULTS** tab to your local machine. If you want to send your output directly to another location, you must open your own ODS destination. By default, the results that are generated by the ODS destinations that you open are written to your current working directory in SAS Studio. Your home directory appears in the SAS Studio navigation pane under **Files and Folders** or **Server Files and Folders**.

If you want to send the results to a specific location, use a FILENAME statement to define a file reference to the desired location. You can use the &_SASWS_ macro variable to reference your home directory. After you create the file reference, use the PATH=*file-reference* option in your ODS statement. Here is an example.

```
filename odsout "&_SASWS_/charts";
ods _all_ close;
ods html path=odsout file="sales.htm";
```

In this case, file sales.htm and any image files that are generated are written to subdirectory **charts** in your home directory.

Use a Custom Style for Your Output

When you need to use a custom ODS style such as a corporate style for your results in SAS Studio, you must open your own ODS destination. You cannot specify a custom style for the default results. Use the STYLE= option in your ODS statement to specify your custom style. Here is an example.

```
filename odsout "&_SASWS_/charts";
ods _all_ close;
ods html path=odsout file="filename.htm" style=style-name;
```

For information about creating custom styles, see *SAS Output Delivery System: User's Guide*.

Use an Image Format Other Than the Default

When you need to use an image format other than the default, you must specify the desired output format, and then open your own ODS destination. In the default SAS

Studio output environment, the default image format for the HTML5 results is PNG. For example, if you want to use SVG, use the DEVICE=SVG option in a GOPTIONS statement, and then open your ODS destination. Here is an example.

```
filename odsout "&_SASWS_/charts";
ods _all_ close;
goptions reset=all device=svg;
ods html5 path=odsout file="filename.htm";
```

For information about using the DEVICE= graphics option, see "Controlling the Graphics Output Format with the DEVICE= Option" on page 69.

Create a Drill-Down Graph

When you need to create a drill-down graph in SAS Studio, you must open your own ODS destination. Drill-down graphs provide a convenient means for users to explore complex data. In a drill-down graph, certain elements of the graph contain active links. When a user clicks a linked element, the linked resource appears in a new browser window by default. For information about creating drill-down graphs, see Chapter 17, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 249.

Create an Animated GIF or SVG Graph

When you need to create an animated graph in SAS Studio, you must open your own ODS destination. An animated graph displays a series of charts automatically when the graph is viewed in a web browser or other viewer that supports animation. The animation plays as a sequence of graphs in a slide-show fashion with a delay between each graph. The sequence can play one time, loop a fixed number of times, or loop indefinitely.

For information about creating animated graphs, see Chapter 16, "Generating Animations," on page 227.

Controlling the Graphics Output Format with the DEVICE= Option

Overview of Devices and Destinations

SAS/GRAPH procedures use device drivers to generate graphics output. Device drivers determine the format of your graphics output. For example, the GIF device driver generates GIF image files. The SASPRTC device generates graphics output

for the current printer as determined by the PRINTERPATH= system option (or the SYSPRINT= system option on Windows).

Every ODS destination has a default device driver that is associated with it. For example, the default device driver for the HTML destination is PNG. By default, when you send output to the HTML destination, your graphics output is rendered as a PNG file. (An HTML file is also generated. This HTML file contains any non-graphical output generated by your application plus an `<IMG>` tag that links to the PNG output that was generated.)

Each destination supports several devices. For example, the HTML destination supports the SVG, PNG, and GIF devices, in addition to several others. "Viewing the List of All Available Devices" on page 95 describes how to display the entire list of devices that are available. Table 6.2 on page 71 lists the default and supported devices for the LISTING, EPUB, HTML, HTML5, RTF, PDF, PowerPoint, and PRINTER destinations.

Specifying the DEVICE= Graphics Option

You can change the device, and therefore the format of your graphics output, by changing the device driver that SAS uses. You can specify a device in the OPTIONS statement, the SAS/GRAPH GOPTIONS statement, or the ODS destination statement.

For example, to use the GOPTIONS statement to change the device, submit this code:

```
goptions  device=device-entry;
```

The DEVICE= option in the ODS destination statement overrides the DEVICE= option in a GOPTIONS or OPTIONS statement.

> **TIP** In SAS Studio, the default ODS HTML5 destination statement specifies DEVICE=PNG, which overrides the GOPTIONS and OPTIONS statement DEVICE= option. If you want to use a different device for your HTML output in SAS Studio, open a separate ODS HTML destination.

Devices that you might specify include SVG, PNG, GIF, ACTXIMG, and many others. For all open destinations, SAS/GRAPH attempts to use the device that you specify. If the device that you specify is not valid for an open destination, SAS/GRAPH switches to the default device for that destination.

For details, see "GOPTIONS Statement" on page 430. "Summary of Default Destinations, Styles, and Devices" on page 71 describes the supported devices for the HTML, LISTING, RTF, and PDF destinations. See also "Using Graphics Devices" on page 89.

Summary of Default Destinations, Styles, and Devices

Each destination has a default device and default style that are used if you do not specify otherwise. Also, each destination has a set of recommended devices. Table 6.2 on page 71 summarizes this information for the HTML, LISTING, RTF, and PDF destinations.

You can use any style with any destination. If you specify a device with the GOPTIONS DEVICE= option, you should specify a device that is compatible with all of the destinations that you have open.

Table 6.2 *Default Devices and Styles for Commonly Used ODS Destinations*

ODS Destination	Default Device	Default Style	Default Output	Recommended Devices
EPUB	PNG	Daisy	EPUB file	GIF PNG SVG
Excel	PNG	Excel	EMF	EMF JPEG JPG PNG
HTML	PNG	HTMLBlue	HTML and PNG files	ACTIVEX[4, 6, 7] ACTXIMG[4, 3] GIF JAVAIMG PNG SVG
HTML5	SVG	HMTLEncore	HTML with inline SVG	ACTXIMG[4] GIF JAVAIMG PNG SVG[3]
LISTING	WIN (Windows) XCOLOR (UNIX) IBMPCGX (z/OS)	Listing	Graphics output is displayed in the GRAPH window.[1]	All devices except ACTIVEX

ODS Destination	Default Device	Default Style	Default Output	Recommended Devices
PDF	SASPRTC	Pearl	PDF file	SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)
PowerPoint	PNG	PowerPointLight	PPTX file	ACTXIMG[4, 3] BMP EMF GIF JAVAIMG PNG TIFF
PRINTER	SASPRTC	Pearl	Controlled by the PRINTERPATH= system option (and by the SYSPRINT= system option on Windows)[2]	SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)
RTF	EMF	Rtf	RTF file (with embedded metafile)	ACTIVEX[4, 5, 7] ACTXIMG[4, 3] EMF PNG PNG300 SASEMF
WORD	PNG	Word	Microsoft Office XML format for Office 2010 and later	EMF GIF JPEG PNG SVG TIFF

1 The default devices for the LISTING destination do not write image files to disk. However, you can produce files on disk by specifying a device that produces an image file (such as GIF, PNG, and so on), with the DEVICE= option in the GOPTION statement.

2 In Windows, if the PRINTERPATH= option is not specified, then SAS uses the setting of the SYSPRINT= system option. If neither the SYSPRINT= nor the PRINTERPATH= option has been set, then SAS uses the default Windows printer.

3 The ACTXIMG device is available only on Windows platforms.

4 Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

5 See Problem Note 65142: Creating an RTF file by using the ACTIVEX device driver results in an RTF file that Microsoft Word cannot open.

6 See Usage Note 57554: Displaying an ActiveX graph in Microsoft Internet Explorer might result in a BEX error after the application of Microsoft patch KB3000850.

7 For more information about the ACTIVEX device, see Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

Note: SASHELP.DEVICES also has high-resolution versions of the PNG and JPEG devices, PNG300 and JPEG300. These devices are not appropriate choices for the HTML destination. Web browsers cannot display images in high resolution, so high resolution images appear very large.

Sending Output to Multiple Open Destinations

Tips for Sending Output to Multiple Destinations

When you are sending output to more than one destination at the same time, you should remember two points:

- You should close any open destinations whose output you are not interested in. Doing so saves system resources.
- If you specify a device that is not supported for an open destination, SAS/GRAPH switches to the default device for that destination and prints a warning to the SAS log.

Closing Destinations to Save System Resources

SAS/GRAPH creates output for every open destination. The HTML destination is open by default, and you can open as many additional destinations as needed. For example, you can open the LISTING and PDF destinations, and generate output for all three destinations by submitting your SAS code only once. However, SAS/GRAPH goes through the process of generating GRSEG catalog entries and graphics output files for each open destination. This process uses system resources. Each open destination increases system resources required by your application. If you are not interested in the output of a destination, it is recommended that you close that destination.

Specifying Devices and Styles with Multiple Open Destinations

Unless you specify a different device or different style, SAS/GRAPH uses the default device and default style for each open destination.

For example, suppose your SAS program specifies the following:

```
ods rtf;
      /* procedure statements and other program elements here */
ods rtf close;
```

SAS/GRAPH uses the PNG device and the HTMLBlue style to generate output for the HTML destination (which is open by default). It also uses the SASEMF device and the RTF style to generate output for the RTF destination.

If you specify a different device with the DEVICE= option in the GOPTIONS statement, SAS/GRAPH attempts to use that device to generate output for every open destination. If you want to use a different style for all output, you need to specify that style on each ODS destination statement. For example, suppose you want to use the JAVAIMG device and the ANALYSIS style for all output sent to both the HTML and RTF destinations. In this case you would specify the GOPTIONS statement and the STYLE= option as follows:

```
goptions device=javaimg;
ods html style=analysis;
ods rtf style=analysis;
      /* procedure statements and other program elements here */
ods rtf close;
```

If you specify a device that is not supported for an open destination, SAS/GRAPH switches to the default device for that destination and prints a warning to the SAS log.

Related Topics

Additional information is available on all of the SAS/GRAPH output concepts that are described in this chapter. For more information about generating output with the SAS/GRAPH procedures, see the following topics:

- Chapter 11, “SAS/GRAPH Output,” on page 119
- “Developing Web Presentations with the PNG, SVG, and GIF Devices” on page 179 describes the devices that can generate non-interactive graphs that contain interactive capabilities such as drill-down links or data tips. These graphs are often used in web presentations.“Developing Web Presentations with the JAVAIMG and ACTXIMG Devices” on page 180 describes the options available for creating a web presentation with interactive graphs. For details about displaying web presentations using the ActiveX control, see Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

For more information about ODS, ODS destinations, and ODS styles, see the following topics:

- Chapter 10, “Managing Your Graphics with ODS,” on page 111
- Chapter 21, “Using ODS Styles, Device Parameters, and Options,” on page 321

For more information about using the graphics devices, see “Using Graphics Devices” on page 89.

7

Using Run-Group Processing

Using Run-Group Processing

You can use RUN-group processing with the GAREABAR, GBARLINE, GCHART, GKPI, GPLOT, GRADAR, GREPLAY, GSLIDE, and GTILE procedures to produce multiple graphs without restarting the procedure every time.

To use RUN-group processing, you start the procedure and then submit multiple RUN-groups. A *RUN-group* is a group of statements that contains at least one action statement and ends with a RUN statement. The procedure can contain other SAS statements such as AXIS, BY, GOPTIONS, LEGEND, TITLE, or WHERE. As long as you do not terminate the procedure, it remains active and you do not need to resubmit the PROC statement.

To end RUN-group processing and terminate the procedure, submit a QUIT statement or start a new procedure. If you do not submit a QUIT statement, SAS/GRAPH does not terminate RUN-group processing until it reaches another step boundary.

Note: When using SAS/GRAPH with the ODS statement, it is best to use a QUIT statement after each procedure that uses RUN-group processing. Do this rather than relying on a new procedure to end the processing. Running too many procedures without an intervening QUIT statement can use up too much memory. Also, note that failing to submit a QUIT statement before submitting an ODS CLOSE statement results in the process memory not being freed at all.

RUN-group Processing with Global and Local Statements

Global statements and NOTE statements that are submitted in a RUN-group affect all subsequent RUN-groups until you cancel the statements or exit the procedure. For example, each of these two RUN-groups produces a plot and both plots display the title defined in the first RUN-group:

```
/* first run group*/
proc gplot data=sashelp.retail;
   title1 "Sales Summary";
   by year;
   plot sales*year;
run;
   /* second run group */
   plot sales*date;
run;
quit;
```

RUN-group Processing with BY Statements

BY statements persist in exactly the same way as global and local statements. Suppose you submit a BY statement within a RUN-group. Then the BY-group processing produces a separate graph for each value of the BY variable for the RUN-group in which you submit it. This continues for all subsequent RUN-groups until you cancel the BY statement or exit the procedure. Thus, as you submit subsequent action statements, you continue to get multiple graphs (one for each value of the BY variable). For more information, see the “BY Statement” on page 424.

RUN-group Processing with the WHERE Statement

The WHERE statement enables you to graph only a subset of the data in the input data set. If you submit a WHERE statement with a RUN-group, the WHERE

definition remains in effect for all subsequent RUN-groups until you exit the procedure or reset the WHERE definition.

Using a WHERE statement with RUN-group processing follows most of the same rules as using the WHERE statement outside of RUN-group processing with one exception. With a procedure that is using an Annotate data set, the following requirements must be met:

- The ANNOTATE= option must be included in the action statement.
- The WHERE variable must occur in both the input data set and the Annotate data set.

8

The Graphics Output Environment

Overview

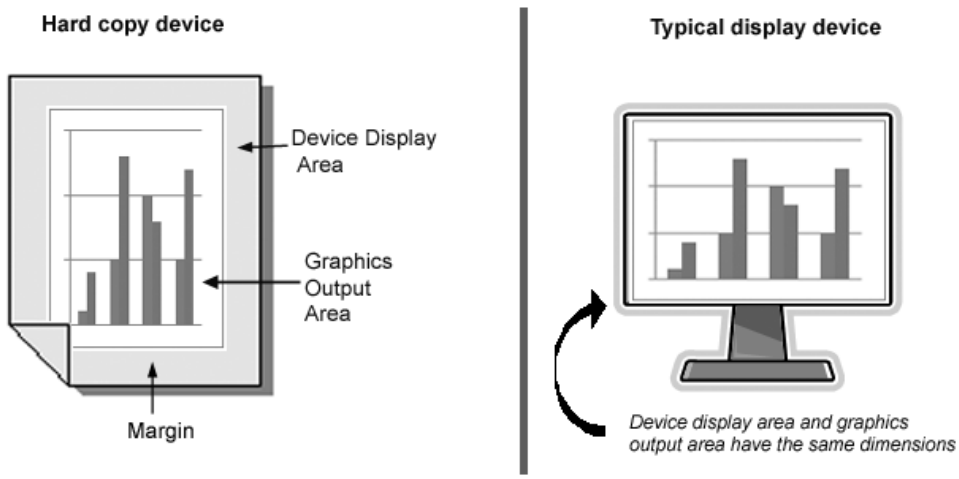
The result of most SAS/GRAPH procedures is the graphic display of data in the form of graphics output. *Graphics output* consists of commands that tell a graphics device how to draw graphics elements. A *graphics element* is a visual element of graphics output. Examples include a plot line, a bar, a footnote, the outline of a map area, or a border.

To generate graphics output, your SAS program uses a device driver that directs the graphics output to a display device (a monitor or terminal), a hard-copy device, or a file. Even though all graphics devices do not understand the same commands, SAS/GRAPH can produce graphics output on many types of graphics devices.

Your program controls this process as well as the environment in which the graphics appear. This section describes this graphics environment and how you can modify it to make your programs work for different output devices.

The Graphics Output and Device Display Areas

When SAS/GRAPH produces graphics output, it draws the graphics elements inside an area called the *graphics output area*. The graphics output area is contained within the *device display area*. Characteristics of both the graphics output area and the device display area are determined by the values of specific device parameters. In many cases the dimensions of the graphics output area equals those of the device display area. This is particularly true for display devices such as monitors and terminals. Hard-copy devices, such as a printed output, create a margin since the dimensions of the graphics output area are smaller than those of the device's display area.



You can modify some of the characteristics of the graphics output area and the device display area by using graphics options to change the values of the device parameter.

This section describes how you can change the dimensions of the output and display areas. It discusses how these changes in dimension affect the output. It will also describe the types of units that you can specify for your output. For a description of the graphics options and device parameters referred to in this section, see “Graphics Options and Device Parameters Dictionary” on page 577.

Controlling Dimensions

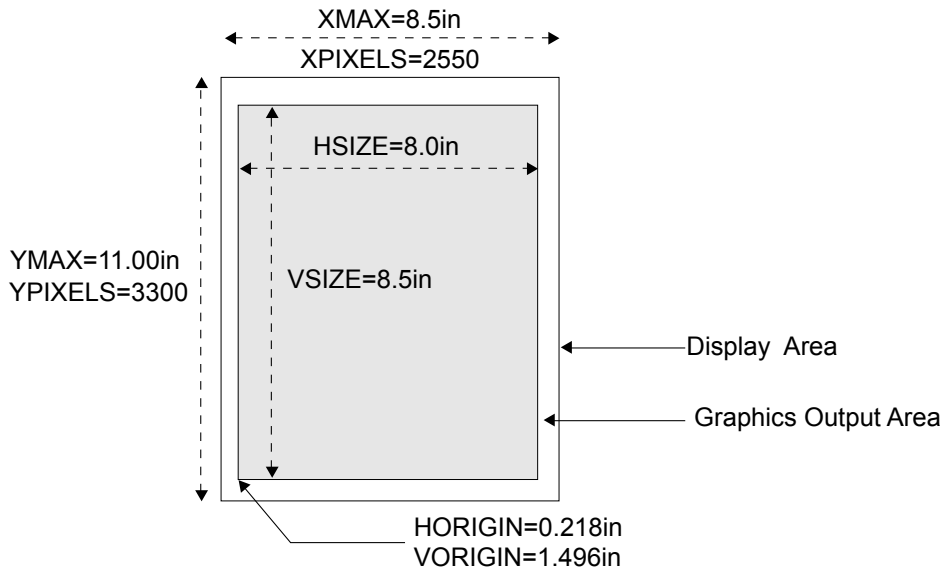
The outer dimensions of the device's display area are controlled by the values of the XMAX and YMAX device parameters. XMAX sets the maximum horizontal dimension; YMAX sets the maximum vertical dimension.

The outer dimensions of the graphics output area are controlled by the values of the HSIZE and VSIZE device parameters.

Since the dimensions of the device display area are typically the same as the dimensions of the graphics output area, the default value of HSIZE and VSIZE is 0. However, for hard-copy devices, because the XMAX, YMAX values represent the outer boundaries of the output medium (such as a sheet of paper), these devices might need a margin. Therefore, HSIZE, VSIZE, HORIGIN, and VORIGIN are assigned default values and the default graphics output area is somewhat smaller than the device's display area. Figure 8.1 on page 81 illustrates such a device.

Note: The default unit of measurement for the XMAX and YMAX options is inches.

Figure 8.1 *Default Dimensions of the PSCOLOR Device*



For additional discussion of how the default values for HSIZE and HORIGIN are determined using the value of the LEFTMARGIN option, see "HSIZE Device Parameter and Graphics Option" on page 649 and "HORIGIN Device Parameter and Graphics Option" on page 647.

Note that HORIGIN and VORIGIN define the left margin and bottom margin, respectively. The right margin and top margin are calculated by the device driver as follows:

right-margin = XMAX − (HSIZE + HORIGIN)

top-margin = YMAX − (VSIZE + VORIGIN)

You cannot specify values for *right-margin* and *top-margin*.

You can change the dimensions of the graphics output area for a SAS session or for a single graph with the HSIZE= and VSIZE= graphics options. Changing the size of the graphics output area does not change the dimensions of the device's display area or affect the resolution. The values of HSIZE= and VSIZE= cannot exceed the maximum dimensions for the device as specified by XMAX and YMAX. Furthermore, you cannot specify values for graphics options HSIZE= and VSIZE= that exceed the maximum HSIZE and VSIZE values for that device.

Controlling Display Area Size and Image Resolution

The resolution of an image is the number of pixels per inch. Resolution is determined by the values of the device parameters XMAX, YMAX, XPIXELS, and YPIXELS, and is calculated by dividing the number of pixels by the corresponding outer dimension. For example:

x-resolution = XPIXELS / XMAX

Therefore, the X resolution of the PSCOLOR device illustrated in Figure 8.1 on page 81 is 300dpi (dots per inch).

Ordinarily, you do not want to change the image resolution because changing it might distort your image. However, you might want to change the size of the display area. To do so without changing the resolution, use the GOPTIONS statement to change either the values of XPIXELS= and YPIXELS=, or the values of XMAX= and YMAX=. SAS/GRAPH automatically calculates the correct value for the unspecified parameters so that the device retains the default resolution.

For information about controlling the resolution of your image see “Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” on page 131.

Units

Cells

Within the graphics output area, SAS/GRAPH defines an invisible grid of rows and columns. This grid consists of *character cells* as shown in Figure 8.2 on page 83.

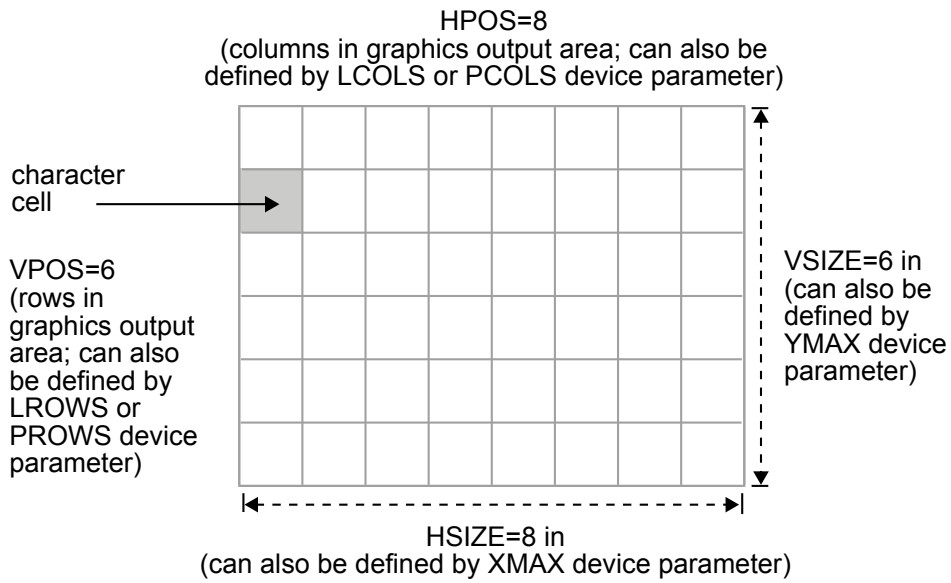
The size and shape of these cells affect the size and appearance of your graph since each graphics element is drawn using units of cells. The size and shape of the cells are determined by the size of the graphics output area. They are also

determined by the number of rows and columns that SAS/GRAPH has defined in the grid. You can control the number of rows by specifying the LROWS device parameter (for a landscape orientation) or the PROWS device parameter (for a portrait orientation). Similarly, the number of columns is controlled by the LCOLS (landscape) or PCOLS (portrait) device parameter.

It is not recommended that you change the number of rows and columns in the grid from the default for your device. If you must do so, you can specify the HPOS= and VPOS= graphics options. HPOS= overrides the value of LCOLS or PCOLS and sets the number of columns in the graphics output area. VPOS= overrides the value of LROWS or PROWS and sets the number of rows in the graphics output area.

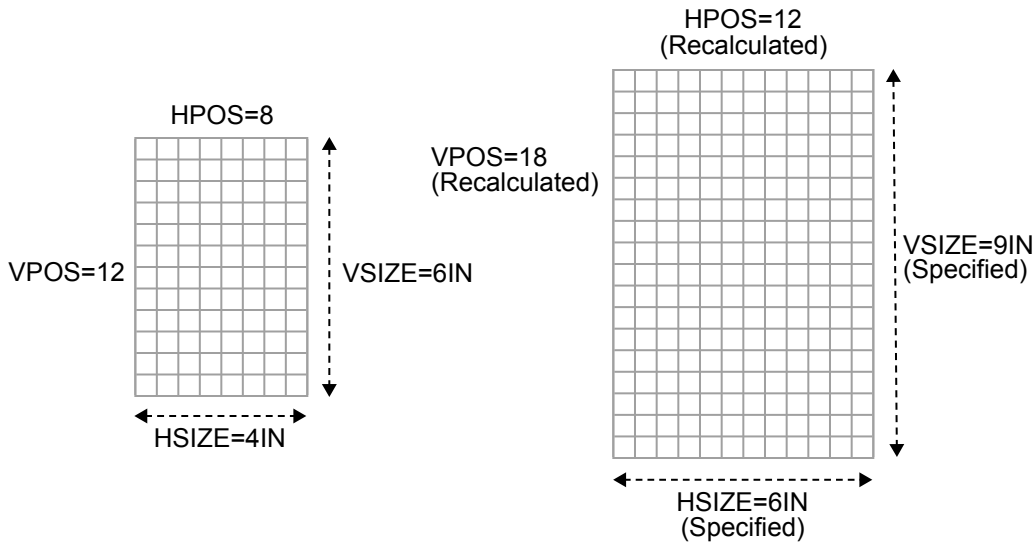
Figure 8.2 on page 83 illustrates how device parameter settings for the size of the output area relate to the parameter settings for the number of character cells in the output area.

Figure 8.2 *Rows, Columns, and Cells in the Graphics Output Area*



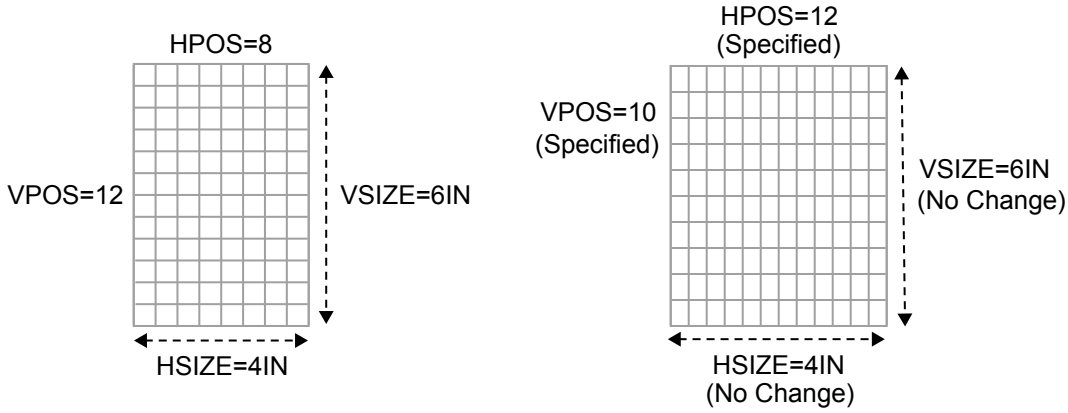
Changing only the outer dimensions of the graphics output area (HSIZE= and VSIZE=) retains the cell size. However, this causes SAS/GRAPH to automatically recalculate the number of rows and columns, as illustrated in Figure 8.3 on page 84.

Figure 8.3 Changing HSIZE= and VSIZE= Changes Dimensions and Recalculates the Number of Rows and Columns



Changing only the number of rows and columns (HPOS and VPOS) changes the size of the cells without altering the overall size of the output. Figure 8.4 on page 84 shows how increasing the number of rows and columns reduces the size of the individual cells.

Figure 8.4 Changing HPOS= and VPOS= Changes Cell Size



Suppose you use units of CELLS to control the size of the text in your graph while also changing the number of rows and columns. In this case the size of the text changes. If the cells are large (that is, HPOS= and VPOS= have small values), the text might not fit. If the cells are too small, the text might be too small to read. In this case, you can adjust the size of the text with the HEIGHT= statement option or the HTEXT= graphics option.

To change all the attributes of the graphics output area, specify values for all four options, as shown in Figure 8.5 on page 85.

Figure 8.5 Changing HSIZE=, VSIZE=, HPOS=, and VPOS= Changes Dimensions and the Number and Size of Cells

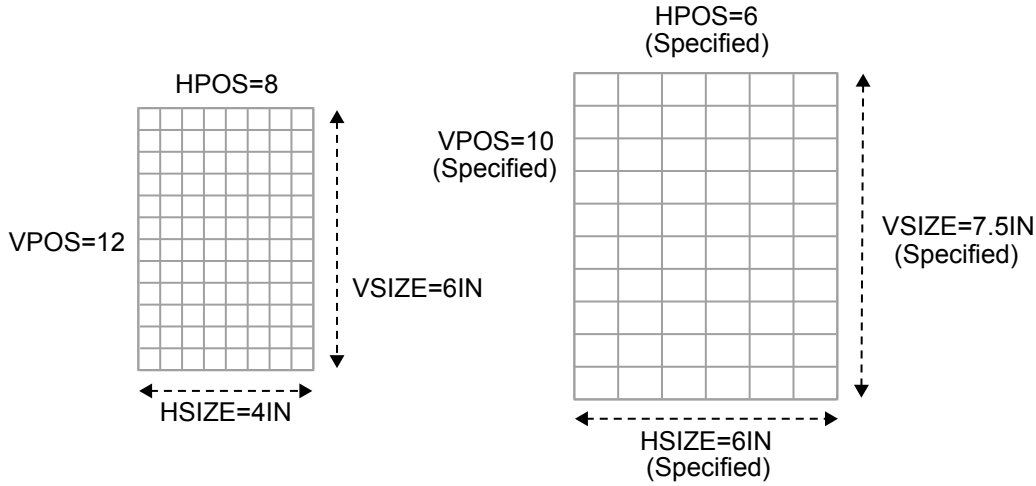


Table 8.1 on page 85 summarizes the interaction of the HSIZE=, VSIZE=, HPOS=, and VPOS= graphics options.

Table 8.1 Interaction of Graphics Options Affecting Cells

Options Specified	Options Not Specified	Result
HSIZE= and VSIZE=	HPOS= and VPOS= (or specify HPOS=0 and VPOS=0)	Changes the external dimensions of the graphics output area and recalculates the number of rows and columns in order to retain cell size and proportions.
HPOS= and VPOS=	HSIZE= and VSIZE=	Keeps the external dimensions but changes the cell size according to the number of rows and columns.
HSIZE=, HPOS=, VSIZE=, and VPOS=		Changes the dimensions of the graphics output area, the number of rows and columns, and recalculates the cell size.

Other Units

By default, most graphics elements are drawn using units of CELLS to determine their size. For example, the default character height for the TITLE1 definition is two cells; for all other text the default height is one cell.

Changing the cell size to control the size of one element, such as text, can distort other parts of your graph. Instead, you might want to change the type of units that SAS/GRAPH uses to control the size of the graphics elements. In addition to CELLS, you can use the following units:

- inches (IN)
- centimeters (CM)
- points (PT)
- percent (PCT)

The percent unit specification is often a good choice because it changes the height of the graphics elements in proportion to the size of the graphics output area.

You can specify the unit for individual graphics elements, or you can use the GUNIT= graphics option to set the units for most graphics element heights.

Maintaining the Quality of Your Image across Devices

When you want to write a program that produces the same graphics output on two different devices, you can use features in SAS/GRAPH to simplify the process.

Maintaining Proportions

You can use percent of the graphics output area (PCT) as the unit of measure when specifying text size to make sure that text is proportional across devices. For example, a one-inch-high title might be appropriate on a standard piece of paper, but a title of this size uses almost all of the display area of a slide. To make units of percentage the default for size specifications, use the GUNIT= graphics option:

```
goptions gunit=pct;
```

You can also specify PCT anywhere you specify a size:

```
axis1 label=(height=3 pct 'Year');
```

See a complete description of the graphics option “GUNIT Graphics Option” on page 642.

Getting the Colors That You Want

ODS styles are designed to provide optimal results for a variety of devices. Therefore, you use the STYLE= option in the ODS statement to specify a style most appropriate for your device. For example, you might want to specify the ODS style **Journal** since it works well with black and white devices. You can also set a different style for each ODS output destination. For information about ODS styles and destinations, see “Specifying Devices and Styles with Multiple Open Destinations” on page 73. You can compare colors and patterns for different devices and choose the device that has the fewest colors. A slide camera, for example, offers more than 16 million colors from which to chose, but some graphics monitors display significantly fewer colors.

Previewing Your Output

You can preview the appearance of the output on a different device with the TARGETDEVICE= graphics option. For example, to see how the output looks on a color PostScript printer, specify as follows:

```
goptions targetdevice=pscolor;
```

How Graphics Elements Are Placed in the Graphics Output Area

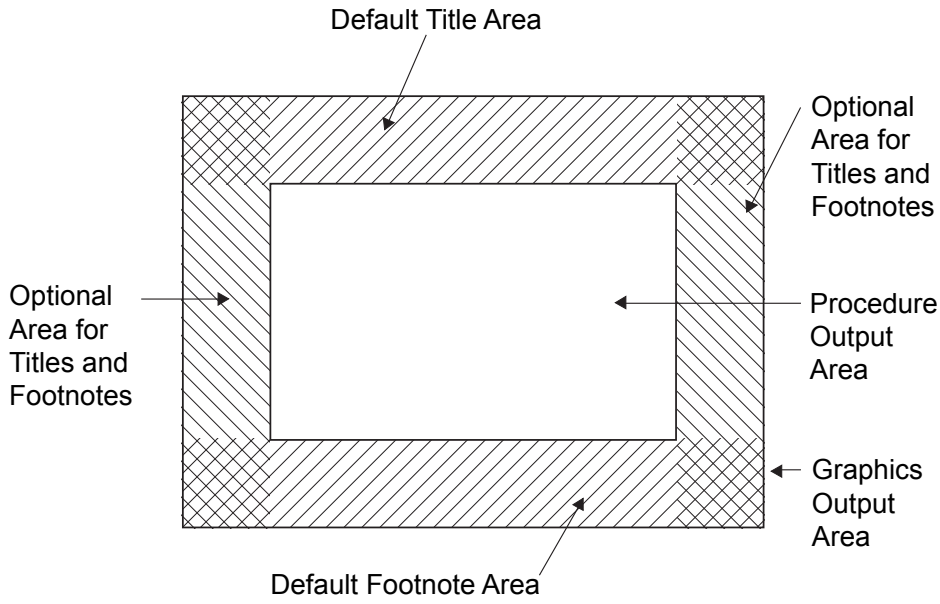
By default, SAS/GRAPH software positions certain graphics elements in predefined locations in the graphics output area.

Figure 8.6 on page 88 shows the graphics output area and the areas within it that are used by the following graphics elements:

- Titles are placed in the title area at the top of the graphics output area.
- Footnotes are placed in the footnote area at the bottom of the graphics output area.
- The graph itself uses the *procedure output area*, which is the area left after the titles and footnotes have been drawn.
- Legends use the procedure output area and can affect the amount of space available for the graph. By default, space is reserved for the legend below the axis area of a graph and above the footnote area. However, you can position the legend in the part of the procedure output area that is reserved for the graph. For details, see the "LEGEND Statement" on page 432.

Note: Titles and footnotes can be positioned elsewhere on the graph as well, with different effects on space allocation. See "TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement" on page 507 for details. For destinations other than the LISTING destination, some graphics elements, such as the title and footnote, can appear in the graphics output instead of the procedure output area.

Figure 8.6 Default Locations for Graphics Elements in the Graphics Output Area



Note: If the titles, footnotes, and legend are very large, they can make the procedure output area too small for the graph. You can control the size of title and footnote text and of most legend elements with statement options. For details, see Chapter 46, "GTILE Procedure," on page 1487 for a description of the appropriate statement. In addition, the "SAS/GRAPH Statements" on page 397 lists the graphics options that control the size of various graphics elements.

How Errors in Sizing Are Handled

Sometimes SAS/GRAPH cannot fit one or more graphics elements on the graph. This can happen if an element is too big for the available space. An example is when the title is too long. Or perhaps you have too many elements to fit in a given space. An example of this is a bar chart with too many bars. In these cases, SAS/GRAPH does one of the following:

- resizes the graphics element and issues a warning explaining what it did
- issues an error message and does not attempt to produce the graph

For example, it adjusts the size of titles to make them fit but it does not drop bars in order to produce a readable bar chart. If you get unexpected results or no graph, check the SAS log for notes, warnings, and errors.

9

Using Graphics Devices

Overview: Using Graphics Devices

SAS/GRAPH procedures that produce graphics output require a device to create the output. The following topics discuss the role of devices in generating SAS/GRAPH output, provide directions for selecting and specifying them, and explain how you can change the settings of device parameters.

What Is a SAS/GRAPH Device?

A SAS/GRAPH device generates graphical output in a specified format. It might send output to a file on disk, such as a PNG file or a GIF file. Or it might send output directly to a hardware device, such as a PostScript printer or a display. A SAS/GRAPH device consists of two parts: a device entry and device driver.

Device entry
: A device entry is a SAS catalog entry of type DEV. Every device that is shipped with SAS/GRAPH has a device entry in the SASHELP.DEVICES catalog. Device entries contain parameters that control the following:

- the appearance of the output when styles are not in effect, such as dimensions and orientation, cell size, colors, and default SAS/GRAPH or device-resident fonts
- where output is sent (when you send output to the LISTING destination and use a SAS/GRAPH device)
- communications between the operating environment and the device
- host commands that are issued before and after its driver produces output
- the device driver that is used to generate graphics output

See also “Viewing and Modifying Device (DEV) Entries” on page 104.

Device driver
: A device driver is the executable module that produces device-specific commands that a device can understand. Every device entry specifies the name of the executable module (device driver) that is to be used to generate output. The device driver uses the parameters specified in the device entry or the current style to tell it exactly how to do so.

When you specify a device, you are specifying the name of a device entry. SAS/GRAPH uses that device entry to determine which device driver to use in order to generate final output. However, most users do not ever need to deal directly with device drivers, so for simplicity, this document simply refers to “devices”.

Commonly Used Devices

The following table lists some of the more commonly used SAS/GRAPH devices and describes the output that they produce.

Table 9.1 *SAS/GRAPH Devices and the Output They Generate*

Device	External files
ACTIVEX[1, 2, 3, 4, 5]	In SAS 9.4M7 and prior releases, an HTML file or an RTF file that contains XML code that is consumed by the SAS/GRAPH ActiveX control. This device must be used with the ODS HTML or ODS RTF destination. When the HTML or RTF file is viewed, the SAS/GRAPH output is displayed as an interactive ActiveX control. The user must install the ActiveX control to view the output of the ACTIVEX device. This device and the ACTXIMG device together are often referred to as ActiveX devices in this document.
ACTXIMG[1, 2]	In SAS 9.4M7 and prior releases, a PNG file that contains an image of the graph that is generated with the ACTIVEX device. The user does not need to install the SAS/GRAPH ActiveX control to view the output from the ACTXIMG device. This device and the ACTIVEX device together are often referred to as ActiveX devices in this document.
BMP	A BMP file that contains the graph.
CGM	A CGM file that contains the graph.
CGMOF97L	A CGM file suitable for inserting into Microsoft Word or PowerPoint presentations.
EMF	An EMF file that contains the graph. The file contains EMF Plus language extensions.
EMFDUAL	An EMF file that contains the graph. The file contains both traditional EMF and EMF Plus language extensions.
GIF	A GIF file that contains the graph.
IBMPCGX	Display device. This device is available on z/OS hosts only.
JAVAIMG[2]	A PNG file that contains a graph that is generated using Java technology. This device together with other devices is sometimes referred to as a Java device in this document.
JPEG	A JPG file that contains the graph. You can use the JPEGQUALITY= system option to control the image quality and

Device	External files
	compression of JPEG images. See *SAS System Options: Reference* for more information.
PCL5	A PCL file that contains one or more graphs and tables.
PDF	A PDF file that contains one or more graphs and tables.
PNG	A PNG file that contains the graph. This is often the default driver when using SAS both in Batch mode and in the window environment.
PSCOLOR	A PostScript file that contains one or more graphs.
PSL	A PostScript file that contains the graph in gray scale.
SASEMF	An EMF file that contains the graph. The file contains traditional EMF language.
SVG	An SVG file that contains the graph.
TIFF	A TIFF file that contains the graph in color. This device uses RGBA colors.
TIFFK	A TIFF file that contains the graph in color. This device uses CMYK colors.
WIN	Display device. This device is available on Windows hosts only.
XCOLOR	Display device. This device is available on UNIX hosts only.

1 Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

2 In the z/OS operating environment, starting with SAS 9.4M8, this device is no longer supported. If you attempt to use this device on z/OS, an error results.

3 For the RTF destination, see Problem Note 65142: Creating an RTF file by using the ACTIVEX device driver results in an RTF file that Microsoft Word cannot open.

4 For the HTML destination, see Usage Note 57554: Displaying an ActiveX graph in Microsoft Internet Explorer might result in a BEX error after the application of Microsoft patch KB3000850.

5 For detailed information about the ACTIVEX device, see Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

Note: Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191 describes any requirements or limitations associated with using the ACTIVEX device.

Note: Chapter 13, “Using SVG Graphics,” on page 159 describes any requirements or limitations associated with using the SVG device.

Default Devices for ODS Destinations

Each ODS destination has a default device. Table 9.10 summarizes this information for the most commonly used destinations. These default devices are used to generate output for each open destination unless you override the default device as described in "Overriding the Default Device" on page 97.

Table 9.2 *Default Devices and Styles for Commonly Used ODS Destinations*

ODS Destination	Default Device	Default Style	Default Output	Recommended Devices
EPUB	PNG	Daisy	EPUB file	GIF PNG SVG
Excel	PNG	Excel	EMF	EMF JPEG JPG PNG
HTML	PNG	HTMLBlue	HTML and PNG files	ACTIVEX[4, 6, 7] ACTXIMG[4, 3] GIF JAVAIMG PNG SVG
HTML5	SVG	HMTLEncore	HTML with inline SVG	ACTXIMG[4] GIF JAVAIMG PNG SVG[3]
LISTING	WIN (Windows) XCOLOR (UNIX) IBMPCGX (z/OS)	Listing	Graphics output is displayed in the GRAPH window.[1]	All devices except ACTIVEX
PDF	SASPRTC	Pearl	PDF file	SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)

ODS Destination	Default Device	Default Style	Default Output	Recommended Devices
PowerPoint	PNG	PowerPointLight	PPTX file	ACTXIMG[4, 3] BMP EMF GIF JAVAIMG PNG TIFF
PRINTER	SASPRTC	Pearl	Controlled by the PRINTERPATH= system option (and by the SYSPRINT= system option on Windows)[2]	SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)
RTF	EMF	Rtf	RTF file (with embedded metafile)	ACTIVEX[4, 5, 7] ACTXIMG[4, 3] EMF PNG PNG300 SASEMF
WORD	PNG	Word	Microsoft Office XML format for Office 2010 and later	EMF GIF JPEG PNG SVG TIFF

1 The default devices for the LISTING destination do not write image files to disk. However, you can produce files on disk by specifying a device that produces an image file (such as GIF, PNG, and so on), with the DEVICE= option in the GOPTION statement.

2 In Windows, if the PRINTERPATH= option is not specified, then SAS uses the setting of the SYSPRINT= system option. If neither the SYSPRINT= nor the PRINTERPATH= option has been set, then SAS uses the default Windows printer.

3 The ACTXIMG device is available only on Windows platforms.

4 Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

5 See Problem Note 65142: Creating an RTF file by using the ACTIVEX device driver results in an RTF file that Microsoft Word cannot open.

6 See Usage Note 57554: Displaying an ActiveX graph in Microsoft Internet Explorer might result in a BEX error after the application of Microsoft patch KB3000850.

7 For more information about the ACTIVEX device, see Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

Viewing the List of All Available Devices

You can view the complete list of devices that are available in any of the following ways:

- Use the SAS Explorer window to display the contents of the default device catalog, SASHELP.DEVICES, or any other device catalog.
- Use the GDEVICE procedure to open the GDEVICE DIRECTORY window, which lists all of the devices in the current catalog. By default, the current catalog is SASHELP.DEVICES. To specify a catalog, include the CATALOG= option, as shown in the following statement:

  ```
  proc gdevice catalog=sashelp.devices;
  run;
  ```

  If you do not specify a catalog, and you have defined a libref named GDEVICE0, then the GDEVICE procedure looks in the GDEVICE0 catalog first. See "Using the GDEVICE Windows" on page 1174 for details.
- Use GDEVICE procedure statements to write the list of device drivers to the Output window. For example:

  ```
  proc gdevice catalog=sashelp.devices nofs browse;
      list;
  run;
  quit;
  ```

  The "NOFS" on page 1166 option in the PROC GDEVICE statement causes the procedure not to use the GDEVICE window.

  If you want to write the list of devices to an external file, you can do either of the following actions:

  - save the contents of the Output window.
  - use the PRINTTO procedure to redirect the GDEVICE procedure output to an external file. See *Base SAS Procedures Guide* for a description of the PRINTTO procedure.
- Use the QDEVICE procedure to write the list of all device drivers to a SAS data set. Specify the keyword _ALL_ in the DEVICE statement. See "Displaying Information with the QDEVICE Procedure" on page 95 for more information.

Displaying Information with the QDEVICE Procedure

The QDEVICE procedure produces reports about graphics devices and Universal Printers. You can use the information in these reports to determine the best device

or printer to use for a specific application. See "QDEVICE Procedure" in *Base SAS Procedures Guide* for more information.

The QDEVICE procedure generates six different reports. These reports summarize information such as color support, default output sizes, margin sizes, resolution, supported fonts, hardware symbols, hardware fill types, hardware line styles, device options, and much more. The default report type is the GENERAL report. You can send the output of this procedure to the SAS log or to an output SAS data set.

For example, the following code produces a GENERAL report for the PNG and GIF devices and the PDF and SVG Universal Printers. It sends this report to WORK.MYREPORT.

```
proc qdevice out=myreport;
  device png imgpng gif imggif;
  printer svg pdf;
run;
```

The following code prints specific information from WORK.MYREPORT.

```
proc print data=myreport;
   var name desc type clrspace
      height width units;
run;
```

Figure 9.1 *PRINT Procedure Output*

Obs	NAME	DESC	TYPE	CLRSPACE	HEIGHT	WIDTH	UNITS
1	PNG	Portable Network Graphics	Shortcut Device	RGBA	6.25	8.33	IN
2	IMGPNG	Portable Network Graphics Format -- Perf	Graph Device	RGB	6.25	8.33	IN
3	GIF	Graphics Interchange Format RGB Color/Alpha Blending	Shortcut Device	RGBA	6.25	8.33	IN
4	IMGGIF	Graphics Interchange Format -- Performan	Graph Device	RGB	6.25	8.33	IN
5	SVG	Scalable Vector Graphics 1.1	Universal Printer	RGBA	6.25	8.33	IN
6	PDF	Portable Document Format Version 1.4	Universal Printer	RGBA	10.50	8.00	IN

You can display information about all of the available device drivers by specifying the keyword _ALL_ in the DEVICE statement. For example, the following code produces a GENERAL report for all devices and sends this report to WORK. ALLDEV.

```
proc qdevice out=alldev;
  device _ALL_;
run;
```

Deciding Which Device to Use

The default device for each ODS destination generates optimal results for that destination. It is recommended that you use the default device whenever possible. If you do not specify a device, then SAS/GRAPH automatically uses the default device listed in Table 9.9 on page 91 for each open destination.

Note: If you are working with multiple open destinations, see "Specifying Devices and Styles with Multiple Open Destinations" on page 73.

If you need to specify a different device, you should specify one of the recommended devices in the table in Table 9.9 on page 91. If you specify a device that cannot be used with an open destination, SAS/GRAPH switches to a device that produces similar results as the device that you specified.

The SAS/GRAPH device that you specify should be appropriate for your specific output device. For example, if you use a color PostScript printer and you select a device for a black and white PostScript printer, your graph is not printed in color.

When you are sending output to the HTML destination, there are several devices that you can specify. See "Developing Web Presentations with the PNG, SVG, and GIF Devices" on page 179 for information and recommendations on which device to use.

Note: High resolution devices such as PNG300 and JPEG300 are not appropriate choices for the HTML destination. Web browsers cannot display images in high resolution, so high resolution images appear very large. These drivers are appropriate for high resolution output that can be inserted into other software applications.

When you are running SAS without a terminal, there are choices in options that you must specify. Use the DEVICE= option in the GOPTIONS statement or specify the DEVICE= or PRINTER= option in the ODS destination statement. No terminal is associated with your SAS session if you are running in batch mode on z/OS, if you are running SAS on UNIX without an X Server. Neither is there a terminal associated with your SAS session if you have specified the NOTERMINAL system option. See *SAS Output Delivery System: User's Guide* for information about specifying the DEVICE= and PRINTER= options on the ODS destination statements.

Starting with SAS 9.4M5, for SAS jobs that generate a large number of graphs, using the IMGPNG or IMGGIF device might improve job performance. When used to render a large number of graphs, the performance of these devices is better than their corresponding Universal Printer shortcut devices, PNG and GIF.

Overriding the Default Device

You can override the default device in a SAS session in the following ways:

- Specify the name of a device entry with the DEVICE= option in a GOPTIONS statement. For example:

  ```
  goptions device=gif;
  ```

  For details, see "GOPTIONS Statement" on page 430.
- Specify the name of a device entry with the DEVICE= option in an OPTIONS statement. For details, see "DEVICE= System Option" on page 716.
- Specify the name of the device entry with the DEVICE= option in an ODS destination statement. You can specify this option on the ODS HTML, ODS

LISTING, ODS MARKUP, and ODS RTF statements. When you specify a device in an ODS destination statement, do not specify the ACTIVEX, ACTXIMG, or JAVAIMG devices. For additional information about these ODS destination statements, see *SAS Output Delivery System: User's Guide*.

- Enter **OPTIONS** on the SAS command line, or select **Tools** ⇨ **Options** ⇨ **System** to open the **SAS System Options** window. Expand **Graphics**, and select **Driver settings**. Right-click **Device**, select **Modify value**, and specify the name of the graphics device that you want to use.
- Enter the device name in the DEVICE prompt window. This window appears automatically if you submit a SAS/GRAPH program that produces graphics output, no device has been specified, and you are running outside of the SAS windowing system environment.

If you specify a device in more than one way, the last specification that SAS/GRAPH encounters is the one that it uses. The device specification stays in effect until you specify another device, submit the graphics option RESET=GOPTIONS or RESET=ALL, or end your SAS session.

If you use the same device for most or all of your SAS/GRAPH programs, you can put the GOPTIONS DEVICE= statement in an AUTOEXEC file. See the SAS companion for your operating environment for details about setting up an AUTOEXEC file.

You can also specify a device for previewing or printing your output with the TARGETDEVICE= graphics option. For details, see "Printing Your Graph" on page 151.

If you submit a SAS procedure without specifying a device and your display device does not support the GRAPH window, then SAS/GRAPH prompts you for a device. SAS/GRAPH also prompts you when you are running outside the SAS windowing system.

Device Categories and Modifying Default Output Attributes

Note: Chapter 21, "Using ODS Styles, Device Parameters, and Options," on page 321 describes the recommended methods for controlling the attributes of your SAS/GRAPH output. Modifying device parameters should be attempted only in unusual circumstances when modifying parameters and options in the GOPTIONS statement is not sufficient. If you need to modify a device entry, consider contacting SAS Technical Support for assistance first.

There are four general categories of devices that are distributed with SAS/GRAPH. The type of device determines how you control certain aspects of your output.

Traditional SAS/GRAPH devices
: produce output in the native language of the device. They do not set or use the SYSPRINT= or PRINTERPATH= system options as do some other device types. For example, TIFFP, IMGPNG, and IMGGIF are traditional SAS/GRAPH devices.

Note: The IMGPNG and IMGGIF devices are valid starting with SAS 9.4M5.

In output from the QDEVICE procedure, a traditional SAS/GRAPH device is identified as a “Graph Device”. See Figure 9.13 on page 96 for more information. With traditional SAS/GRAPH devices, you can specify default attributes for your output by customizing the device entry (the DEV catalog entry). For example, by editing the DEV catalog entry for the device, you can change several attributes. You can change the default size and resolution of your output and the default colors and fonts that are used when styles are turned off.

Java and ActiveX devices
: produce output using different technologies than the traditional SAS/GRAPH devices. These devices are the JAVAIMG, ACTIVEX, and ACTXIMG devices. These devices do not use the information specified in the device (DEV) catalog entry.

Note: For more information about the ACTIVEX device, see Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

Universal Printer shortcut devices
: use the Universal Printing system to generate output. Universal Printing is a printing system that provides printing capabilities to SAS applications and procedures on all the operating environments that are supported by SAS. It is part of Base SAS. For information about universal printing, see *SAS 9.4 Universal Printing*.

Universal Printer shortcut devices can generate output in the following formats: PDF, PostScript, PCL, PNG, GIF, EMF, TIFF, and SVG. For example, TIFFK, EMFDUAL, PNG, and SVG are Universal Printer shortcut devices. The description of a Universal Printer shortcut device frequently says “Universal Printer” when you view the contents of the SASHELP.DEVICES catalog. In output from the QDEVICE procedure, a Universal Printer shortcut device is identified as a “Shortcut Device”. See Figure 9.13 on page 96 for an example.See “Viewing the List of All Available Devices” on page 95. The list of all Universal Printer shortcut devices is shown in Table 9.12 on page 102.

Universal Printer shortcut devices are designed to emulate a traditional SAS/GRAPH device, which means that these devices behave as much as possible like traditional SAS/GRAPH devices. For example, these devices set the value of PRINTERPATH= so that you need only specify the device name with the GOPTIONS statement. However, for these devices there are some attributes of your output, such as default resolution, that cannot be changed by modifying the DEV catalog entry. See “Using Universal Printer Shortcut Devices” on page 101 for more information.

Interface devices
: are devices that, in some operating environments, use the facilities of the operating environment, and, in other operating environments, use Universal Printing to generate output. There are three subcategories of interface devices: printer, display, and metafile.

The printer interface devices are the SASPRTC, SASPRTG, and SASPRTM devices (and the WINPRT* devices on Windows systems). In Windows operating environments, if the PRINTERPATH= system option has not been set, these devices use the setting of the SYSPRINT= system option. This setting helps determine the default output device and the Windows Print Manager to control the generation of output. In Windows operating environments, the Universal

Printing System is used if the PRINTERPATH= system option is specified or if the UPRINT system option has been specified at invocation. Otherwise, they use the setting of the PRINTERPATH= system option to determine the default output device and the Universal Printing system to control the generation of output.

Table 9.11 on page 100 summarizes the device categories and describes how to control output attributes based on the device category. For more information about the ACTIVEX device, see Chapter 15, "Generating Interactive ActiveX Graphics," on page 191. See also "Viewing and Modifying Device (DEV) Entries" on page 104.

Table 9.3 *Device Categories, GOPTIONS, and DEV Entries*

Device category		Examples	Honor GOPTIONS specifications?	Honor specifications in the device (DEV) catalog entry?
Shortcut devices		EMF GIF PNG SASEMF SVG	Yes, except for resolution [1, 4]	Yes, except for size, resolution, and fonts
Interface devices	Printer[5]	SASPRTC SASPRTG SASPRTM	Yes, except for resolution[6]	Yes, except for size, resolution, and fonts
	Display	WIN XCOLOR	Yes, except for resolution[8]	Yes, except for size[7] , resolution[8] , and fonts
	Metafile	BMP[2]	Yes	Yes, except for resolution[8] and fonts
Traditional SAS/GRAPH devices		BMP[2] CGM IBMPCGX IMGGIF[9] IMGPNG[9] TIFFP	Yes[3]	Yes
Java and ActiveX devices		ACTIVEX[10] ACTXIMG[10] JAVAIMG	Yes, except as noted in the documentation for specific graphics options. Also, resolution is controlled by the	No

Device category	Examples	Honor GOPTIONS specifications?	Honor specifications in the device (DEV) catalog entry?
		operating environment.	

1 With SVG devices, the XMAX= and YMAX= graphics options set the size of the page, and the HSIZE= and VSIZE= graphics options set the size of the SVG output. With other devices, all four options set the size of the graphics output, and if all four are specified, the smaller specifications are used.

2 On Windows, BMP is an interface metafile device. In all other operating environments, BMP is a traditional SAS/GRAPH device.

3 Some traditional devices have a set resolution, and others have a fixed set of supported resolutions that you can specify.

4 Shortcut devices use Universal Printers. Universal Printers have a fixed set of supported resolutions that can be selected through the Print Setup dialog box or through the PRINTDEF procedure.

5 The WINPRT* devices are identical to the SASPRT* devices. They differ in name only.

6 The interface printer devices use a mix of host printing facilities and Universal Printing, depending on the operating environment. On Windows systems, use the Windows Print Manager to change the default resolution and size. On other systems, resolution and size are set through the Print Setup dialog box or through the PRINTDEF procedure.

7 The device is queried. The size is constrained by the window.

8 Display resolution is set in the display properties for the operating environment. The device is queried, and the resolution is set according to the value returned.

9 Valid starting with SAS 9.4M5.

10 Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

Using Universal Printer Shortcut Devices

Universal Printer shortcut devices enable you to generate SAS/GRAPH output using the Universal Printing system without specifying ODS statements or an OPTIONS PRINTERPATH= statement. The shortcut devices were created primarily for use with the LISTING and HTML destinations. They perform two functions:

- set the PRINTERPATH= system option. These options determine which Universal Printer is used to generate your final output. (See “PRINTERPATH= System Option” in *SAS System Options: Reference* for more information.)
- convert SAS/GRAPH GRSEG output into instructions understood by Universal Printers.

Using a Universal Printer shortcut device requires that there is a Universal Printer with the same name in the SAS registry. Universal printers have already been defined for all of the shortcut devices that are shipped with SAS. You can create your own device by copying one of the Universal Printer shortcut device entries,. However, you must make sure that you define a Universal Printer with the same name as your new device entry. For information about creating a new SAS/GRAPH device, see “Creating a Custom Device” on page 105.For information about defining a new Universal Printer, see “Setting Up and Managing Universal Printing” in *SAS 9.4 Universal Printing*.

An example of the differences in specifying a shortcut device and in specifying a Universal Printer directly (without going through the shortcut device) is shown in Table 9.12.

Table 9.4 Differences in Using Shortcut Devices and Universal Printers

Using a shortcut device	Using a Universal Printer directly

`goptions device=PNG;`
`/* procedure step */`
	The following two sets of code are equivalent.
	`ods printer printer=PNG;`
	`/* procedure step */`
	`ods printer close;`
	`options printerpath=PNG;`
	`ods printer;`
	`/* procedure step */`
	`ods printer close;`

The device is set to PNG by the GOPTIONS statement. The default output file name is controlled by the procedure (for example, sasgraph.png).	The device is set to SASPRTC because SASPRTC is the default device for the PRINTER destination. The Universal Printer is set to PNG by the PRINTER= or PRINTERPATH= option. The default output file name is controlled by ODS (for example, sasprt.png).

Table 9.13 lists all of the Universal Printer shortcut devices that are provided by SAS.

Table 9.5 Universal Printer Shortcut Devices

Name	Description
GIF	GIF Universal Printer
EMF	Enhanced Metafile Format Plus Extensions (EMF Plus)[1]
EMFDUAL	EMF and EMF Plus Format Support[1]
PCL5	PCL5 Universal Printer
PCL5C	PCL5c Universal Printer
PCL5E	PCL5e Universal Printer
PDF	PDF Version 1.3 — color
PDFA	Archive PDF - ISO-19005-1/b
PDFC	PDF Version 1.3 — color
PNG	PNG Universal Printer

Name	Description
PNG300	PNG Universal Printer-300 dpi
PNGT	PNG Universal Printer with Transparency
PSCOLOR	PostScript Level 1 (Color)
PSL	PostScript Level 1 (Gray Scale)
PSLEPSF	PostScript EPS (Gray Scale)
PSLEPSFC	PostScript EPS (Color)
SASEMF	Enhanced Metafile Format[1]
SVG	SVG Universal Printer
SVGT	SVG Transparency Universal Printer
SVGVIEW	SVG Printer w/ Control Buttons
SVGZ	SVG Compressed Universal Printer
TIFF	TIFF RGBA Universal Printer
TIFFK	TIFF CMYK Universal Printer

1 SAS can produce EMF files in three different formats: EMF, EMF Plus, and EMF Dual. EMF Dual files contain both EMF and EMF Plus records. The EMF viewer determines which set of records to display. For more information, see "Creating EMF Graphics Using Universal Printing" in *SAS 9.4 Universal Printing*.

Creating Universal Printers and Shortcut Devices

You can use the %SHORTCUT macro to create a Universal Printer shortcut device for an existing Universal Printer. You can also use this macro to define a new Universal Printer and create a shortcut device for the new printer at the same time.

For information about this macro, see "%SHORTCUT Autocall Macro" on page 107.

Viewing and Modifying Device (DEV) Entries

What Is a SAS/GRAPH Device Entry?

As described in "What Is a SAS/GRAPH Device?" on page 90, device entries contain parameters that control much of the default behavior and default output attributes of a device. However, even though a device entry exists for every device, the information in it is not always used. This is especially true for interface and shortcut devices, which derive many attributes from the operating environment and the Universal Printer definition in the SAS registry. Device Categories And Modifying Default Output Attributes See "Device Categories and Modifying Default Output Attributes" on page 98 for more information.

Viewing the Contents of a Device Entry

SAS/GRAPH provides device entries for your operating environment in the SASHELP.DEVICES catalog. If your site has created custom device entries, they might also be stored in SASHELP.DEVICES, although custom devices are typically stored in the catalog GDEVICE0.DEVICES. For more information about custom device entries, see "Device Catalogs" on page 1164 or ask your on-site SAS support personnel.

Use any of the following methods to view the contents of a device entry:

- In the SAS windowing environment, use the SAS Explorer window to display the contents of the DEVICES catalog in the SASHELP library. Double-click a device entry to display the contents of the device entry in the Output window.
- Run the GDEVICE procedure in program mode. For example, the following statements list the contents of the PSCOLOR device entry in the Output window:

  ```
  proc gdevice catalog=sashelp.devices nofs browse;
     list pscolor;
  run;
  quit;
  ```

- In the SAS windowing environment, run the GDEVICE procedure in windowing mode. The following statements open the GDEVICE Directory window, which lists the available devices:

  ```
  proc gdevice catalog=sashelp.devices;
  run;
  ```

  From the GDEVICE Directory window, select the device name to open the GDEVICE Detail window. From there you can move to the other GDEVICE windows for the entry, either by selecting windows from the **Tools** menu or

entering commands on the command line. For details, see "Using the GDEVICE Windows" on page 1174.

Modifying Device Entry Parameters

Use the GDEVICE procedure to modify the properties of an existing device entry. See Chapter 38, "GDEVICE Procedure," on page 1163.

The modifications made to a device entry are in effect for all SAS sessions.

The new values that you specify for device parameters must be within the device's capabilities. For example, devices are limited in the size of the output that they can display. Some output devices cannot display color. If you try to increase the size of the display past the device's capability, results are unpredictable. Also, if you specify colors for a device that cannot display them, results are unpredictable. You cannot force a device to act as a device with different capabilities by choosing a different device driver.

Note: The device driver that is associated with a device entry is shown in the **Module** field in the device entry. It is recommended that you do not change the device driver associated with a device entry. Please contact SAS Technical Support before changing the device driver associated with a device entry.

Note: If you run SAS/GRAPH software in a multi-user environment, you should not change the device entries in the SASHELP.DEVICES catalog. Exceptions to this rule apply if you are the system administrator or other on-site SAS support personnel.

If you need to change a device entry in SASHELP.DEVICES, copy it into a personal catalog named DEVICES, and then modify the copy. To use the new device, assign the libref GDEVICE0 to the library that contains the modified copy. See "Creating or Modifying Device Entries" on page 1181 for details.

Creating a Custom Device

You can use the GDEVICE procedure to create a custom device. For each new device, you need to create a new device entry. Device entries that you create or modify are typically stored in the catalog GDEVICE*n*.DEVICES.

If you want to create a custom device, it is recommended that you copy an existing device and modify it as needed. If you cannot find a device that is suitable for your purposes, contact SAS Technical Support.

See "Modifying Device Entry Parameters" on page 105 and Chapter 38, "GDEVICE Procedure," on page 1163 for more information.

Compatibility Device Drivers (Z Drivers)

The ZPNG and ZGIF devices are intended primarily to provide compatibility with previous releases of SAS/GRAPH.

Note: The ZPNG device is disabled starting with SAS 9.4M2.

These devices ignore the FONTRENDERING= system option and force host font rendering. See "Changing the Appearance of Output to Match That of Earlier SAS Releases" on page 347 for more information.

Additional Considerations for Devices

- Some ODS styles, such BlockPrint, include background images. Not all devices support these images. You can suppress the display of these images by specifying the NOIMAGEPRINT graphics option. See "Disabling and Enabling Image Output" on page 393.

  Note: Starting with SAS 9.4M5, the BlockPrint ODS style is removed from suggested use. This style still functions when used in SAS programs, but it no longer appears in the list of available styles.

- When you are replaying a graph with the GREPLAY procedure, it is recommended that you replay the graph with the same device that it was originally created with. Replaying the graph with other devices can produce undesirable results.See "Replaying Your Output Using the GREPLAY Procedure" on page 147 for additional information.

Related Topics

Other tasks related to devices are discussed in the following topics:

- Chapter 11, "SAS/GRAPH Output," on page 119 provides general information about graphics output formats and the SAS/GRAPH output process. Tasks discussed include, setting the size and resolution of your graphics output and previewing on one device how output looks on another device. Tasks also include sending output directly to a printer or other hard-copy device, and replaying output.

- "Developing Web Presentations with the PNG, SVG, and GIF Devices" on page 179 describes the devices that can generate non-interactive graphs that contain interactive capabilities such as drill-down links or data tips. These graphs are often used in web presentations."Developing Web Presentations with the JAVAIMG and ACTXIMG Devices" on page 180 describes the options available for creating a web presentation with interactive graphs. For details about displaying web presentations using the ActiveX control, seeChapter 15, "Generating Interactive ActiveX Graphics," on page 191.
- Chapter 18, "Using SAS/GRAPH Output with Microsoft Office Products," on page 281 describes how to choose a device for output that you want to use in Microsoft Office products.
- Chapter 38, "GDEVICE Procedure," on page 1163 describes how to create and modify devices.

Dictionary

%SHORTCUT Autocall Macro

Creates a shortcut device that is linked to the existing Universal Printer of the same name, or defines a new Universal Printer and shortcut device and links the device to the Universal Printer.

Syntax

%SHORTCUT (NAME=*new-name* <PROTOTYPE=*existing-name*>
<DESCRIPTION=*description*> <LIBREF=*libref*>
<REGISTRY=SASUSER | SASHELP>)

Required Argument

NAME=*new-name*
specifies the name of the shortcut device and Universal Printer (if a Universal Printer is also being created). The name must be a valid SAS name less than or equal to 8 characters in length with no embedded spaces.

Restriction A device with the same name cannot already exist in the location specified by the LIBREF= option. If you specify the name of a device that already exists, the macro writes an error to the SAS log and does not create the device.

Interactions If you do not also specify the PROTOTYPE= option, then the macro assumes that you want to create a shortcut device to an existing Universal Printer. The *new-name* must match the name of an existing Universal Printer. The list of valid printer names is

available in the SAS registry at CORE\PRINTING\PRINTERS. To view the SAS registry, issue the REGEDIT command.

If you also specify the PROTOTYPE= option, then the macro assumes that you want to create a new Universal Printer. A Universal Printer with the same name must not already exist. If you specify the name of a Universal Printer that already exists, the macro writes an error to the log and does not create the Universal Printer.

Optional Arguments

PROTOTYPE=*prototype-name*
specifies the name of the Universal Printer prototype to use when defining a new Universal Printer. The list of valid prototype names is available in the SAS registry at CORE\PRINTING\PROTOTYPES. To view the SAS registry, issue the REGEDIT command, or select **Solutions** ⇨ **Accessories** ⇨ **Registry Editor**.

DESCRIPTION=*description*
specifies the description to use for the new shortcut device and the new Universal Printer (if created).

LIBREF=*libref*
specifies the library where the new device driver is to be created. You can specify any of the GDEVICE*n* librefs, from GEDEVICE0 to GEDEVICE9. You can also specify the SASHELP libref, but you must have Update access to SASHELP.

If you do not specify the LIBREF= option, SAS/GRAPH assigns the libref GDEVICE0 to the current directory and uses that libref. If you specify a GDEVICE*n* libref, but that libref has not already been assigned to a directory, SAS/GRAPH assigns that libref the current directory.

See "Device Catalogs" on page 1164

REGISTRY=SASUSER | SASHELP
specifies the portion of the SAS registry where the new printer is to reside. If you specify SASHELP, you must have Update access to SASHELP.

Default SASUSER

See "Introduction to the SAS Registry" in *SAS Programmer's Guide: Essentials*

Details

To create both a new Universal Printer and a shortcut device for the new printer, specify both the NAME= and PROTOTYPE= options. The %SHORTCUT macro defines a Universal Printer and a shortcut device with the name specified in the NAME= option. The Universal Printer is based on the specified prototype. The macro links the new Universal Printer to the new shortcut device.

To create a shortcut device for an existing Universal Printer, specify the name of an existing Universal Printer with the NAME= option. Do not specify the PROTOTYPE= option. The %SHORTCUT macro creates a new shortcut device using the NAME=

value as the name. It links the new shortcut device to the Universal Printer of the same name.

In both cases, when the shortcut device is specified in a SAS application, the Universal Printer is used to generate output.

Examples

Example 1: Creating a Shortcut Device to an Existing Universal Printer

This example creates a shortcut device named SVGnotip and links the device to the Universal Printer of the same name.

```
%shortcut(name=SVGnotip,description=SVG with no tooltips,
               libref=gdevice0)
```

Example 2: Defining a New Printer and Creating a New Shortcut Device

This example defines a new Universal Printer and shortcut device named NEWEMF. The Universal Printer is based on the EMF Plus prototype.

```
%shortcut(name=newEMF,prototype=EMF Plus,
               description=Custom EMF Plus,
               libref=gdevice0,registry=sasuser)
```

10

Managing Your Graphics with ODS

Managing Your Graphics with ODS

The Output Delivery System (ODS) manages all output created by procedures and enables you to display the output in a variety of forms, such as HTML, PDF, and RTF. The ODS destination statements provide options for control of many relevant features.

Managing ODS Destinations

ODS supports multiple destinations for procedure output. The most frequently used destinations are LISTING, HTML, RTF, and PDF, although many more destinations are available.

ODS destinations can be open or closed. When a destination is open, ODS can send output to it, and when a destination is closed, ODS cannot send output to it. You can have several destinations open at the same time, and SAS sends output to each destination. The HTML destination is open by default.

An open destination always uses system resources. It is best to close any destinations if you do not need the output from that destination.

Note: For more information about ODS destinations, see *SAS Output Delivery System: User's Guide*.

The following table lists the ODS destinations and the default type of output that results from each destination.

Table 10.1 *Relevant Destination Table*

Destinations	Results	Default Style	Default ImgFmt	Default DPI
DOCUMENT	ODS document	Not applicable	Not applicable	Not applicable
LISTING	SAS output listing	Listing	PNG	96
OUTPUT	SAS data set	Not applicable	Not applicable	Not applicable
HTML	HTML file for online viewing	Default	PNG	96
LATEX [1]	LaTeX file	Default	PostScript	200
PDF	PDF file	Pearl	Native PDF	150
PRINTER	printable output in one of three different formats: PCL, PDF, or PS (PostScript)	MonochromePrinter for PCL, Printer for PDF and PS	Native PCL, Native PDF, or Native PostScript	150
RTF	output written in Rich Text Format for use with Microsoft Word 2000	RTF	EMF	200
Measured RTF		RTF	EMF	200

1. LATEX is an experimental tagset. Do not use this tagset in production jobs.

Specifying a Destination

To generate output from SAS, a valid ODS destination must be open. By default, the HTML destination is open. You can use an ODS destination statement, such as ODS PDF, to open a different destination. You can also specify options, such as the HTML file name or the path to an output directory, on the ODS destination statement.

ODS *destination* <*options*;>

The options available vary with the destination that is specified.

ODS Destination Statement Options

There are several destination statement options that you can use to control where your files or graphics should be written. Destination statement options enable you to specify a different style and the appropriate image resolution in DPI for your output images. The following ODS HTML statement:

```
ods html path="c:\myfiles\"
         gpath="c:\myfiles\images" (url="http://www.sas.com/images/")
         body="barGraph.htm"
         style=analysis;
```

This ODS statement does the following:

- It opens the HTML destination.
- It specifies that the images are to be written to the directory `C:\myfiles\images`.
- It specifies that the URL for the images is `http://www.sas.com/images/`*`image-filename`* in the HTML file.
- It specifies that the other output files (for example, the HTML file) are to be written to the directory `C:\myfiles\`.
- It specifies that the name of the initial HTML file that is displayed is `barGraph.htm`.
- It changes the style to Analysis.

The following ODS HTML statement specifies that the output is sent to the HTML destination. Because it does not specify either the PATH= or GPATH= options, all output is sent to the default SAS folder.

```
ods html body="barGraph.html";
```

The HTML output is written to the file specified by the BODY= option, barGraph.html. At start-up, the default folder is the WORK folder. When running SAS with the windowing environment in the Windows operating system, the current folder is displayed in the status bar at the bottom of the main SAS window. In SAS

Studio, the default folder is your working directory.For more information, see "Output in SAS Studio" on page 65.

If you do not specify a file name for your output, in the SAS windowing environment, SAS provides a default file that is determined by the ODS destination. This file is saved in the WORK folder. You can check the SAS log to verify the name of the file in which your output is saved. In SAS Studio, the output is displayed in the **RESULTS** tab. You can use the download buttons on the **RESULTS** tab to download the results as a file.

Options that you might want to specify on ODS destination statements are the following:

GPATH= (URL=*location* '*Uniform-Resource-Locator*' | NONE)
specifies the location for all graphics output that is generated while the destination is open. You can specify an external file or a fileref. You can use the URL= suboption to specify a URL that is used in links and references to output files. The GPATH= option is valid for the LISTING destination and the Markup family of destinations. If the GPATH option is not specified, the images are written to the location specified by the PATH option. For complete documentation on GPATH= option, see the ODS LISTING statement and the ODS MARKUP statement in *SAS Output Delivery System: User's Guide*.

PATH= (URL=*location* '*Uniform-Resource-Locator*' | NONE)
specifies the location of an external file or a SAS catalog for all markup files. You can specify an external file or a fileref. You can use the URL= suboption to specify a URL that is used in links and references to output files. The PATH= option is valid for the RTF, Measured RTF, and Markup family of destinations. If the PATH option is not specified, images are written to the WORK directory. For complete documentation on PATH= option, see the ODS MARKUP or TAGSET.RTF statements in *SAS Output Delivery System: User's Guide*.

IMAGE_DPI=
specifies the image resolution in DPI for the output images sent to HTML, LISTING, or RTF destinations. The default value is 96. For complete documentation on the IMAGE_DPI= option, see the statement documentation in *SAS Output Delivery System: User's Guide*.

DPI=
specifies the image resolution in DPI for the output images sent to PRINTER family destinations. The default value for the PRINTER destination is 150. For complete documentation on the DPI= option, see the valid ODS PRINTER statement in *SAS Output Delivery System: User's Guide*.

STYLE= *style-definition*
specifies a style to be used for the output. Each ODS destination has a default style for the formatting of output. The style specifies a collection of visual attributes that are used for the rendering of the output. The STYLE= option is valid for all ODS destinations except the Document destination and the Output destination. For complete documentation on the STYLE= option, see the ODS statements in *SAS Output Delivery System: User's Guide*. For more information about using the STYLE= option with SAS/GRAPH output, see Chapter 21, "Using ODS Styles, Device Parameters, and Options," on page 321.

Note: If you specify the PATH= or GPATH= options, the directory name that you specify is used to refer to images that are generated as part of your output. For example, if you are sending output to the HTML destination, and you specify `path="C:\myfiles\"`, then all HTML image tags use that path to refer to your images:

```
<img src="C:\myfiles\myoutput.png">
```

If your browser implements strict security regarding access to local files, you might have problems viewing the images. You can avoid these problems by specifying the URL= suboption.

ODS and Procedures That Support RUN-Group Processing

When you use ODS, it is wise to specify a QUIT statement at the end of every procedure that supports RUN-group processing. If you end every procedure step explicitly, rather than waiting for the next PROC or DATA step to end it for you, then the QUIT statement clears the selection list. Using this approach you are less likely to encounter unexpected results.

Controlling Titles and Footnotes

When you use ODS to send your graphs to an HTML, RTF, or PRINTER destination, you can control the location where titles and footnotes are printed or displayed. They can be displayed by ODS as part of the output file (such as the HTML body file) just as they are with tabular output. Or they can be rendered as part of the graphical image. Whether titles and footnotes are rendered as part of the image determines how you control their font, size, and color.

Controlling Where Titles and Footnotes Are Rendered

Where titles and footnotes are displayed depends on the device driver that you are using and on the setting of the ODS statement options GTITLE and GFOOTNOTE.

For the JAVAIMG, ACTIVEX, and ACTXIMG device drivers, titles and footnotes are created by ODS as part of the HTML or RTF file. The GTITLE and GFOOTNOTE options are ignored for these drivers. (These devices are not supported for the PRINTER destination.)

For all other devices, the GTITLE and GFOOTNOTE options determine where the titles and footnotes are displayed. The GTITLE and GFOOTNOTE options, by default, render titles and footnotes as part of the graphic image. You can specify that titles and footnotes appear as part of the output file (for example, the HTML body file) and not as part of the graphical image. To do this you must specify the NOGTITLE or NOGFOOTNOTE option, as in the following example.

```
/* direct titles and footnotes to the ODS output file */
ods html body="filename.htm" nogtitle nogfootnote;
```

Specifying NOGTITLE or NOGFOOTNOTE can increase the amount of space allowed for the procedure output area, which can increase the size of the graph, but the aspect ratio is preserved. Space that would have been used for the title or footnote is devoted instead to the graph. You might need to be aware of this possible difference if you are using annotate or map coordinates.

Note: When you specify NOGTITLE or NOGFOOTNOTE, the title or footnote text is not part of the GSEG and cannot be retrieved when the graph is replayed.

Using Graphics Options with ODS (USEGOPT)

When titles and footnotes are rendered as part of ODS output file instead of the graphic image, ODS does not recognize the settings for the following graphics options. ODS does recognize the settings if you also specify the ODS USEGOPT statement:

- CTEXT=
- CTITLE=
- FTEXT=
- FTITLE=
- HTEXT=
- HTITLE=

When titles and footnotes are rendered as part of the graphic image, these options are honored regardless of whether you specify the ODS USEGOPTS statement.

For example, the following code generates two graphs. The title for the first graph uses the text color and font as defined by the current style (Analysis). The title for the second graph uses the font size and color specified by the HTITLE and CTEXT options.

```
/* Specify an ODS output path */
filename odsout ".";

ods _all_ close;
ods html path=odsout file="myout.htm" style=analysis;
goptions reset=all dev=activex htitle=8 ctext="black";

ods nousegopt;
title "My title";
footnote "My footnote";
proc gchart data=sashelp.class;
   pie age / discrete legend;
run;

ods usegopt;
   pie age / discrete legend;
run;
```

```
quit;
ods nousegopt;
ods html close;
ods html; /* Not required in SAS Studio */
```

While ODS USEGOPT is in effect, the settings for these graphics options affect all of the titles and footnotes rendered by ODS. To turn off the use of these graphics option settings (for non-graphic output), specify the ODS NOUSEGOPT statement.

The default setting is ODS NOUSEGOPT.

Controlling the Text Font, Size, and Color

SAS looks for information about how to format titles and footnotes in the following order:

1. SAS looks for options on the TITLE and FOOTNOTE statement. For example, you can specify BOLD, ITALIC, FONT=, or HEIGHT= options on these statements.
2. When titles and footnotes are rendered as part of the ODS output file, and the ODS USEGOPTS statement is used, SAS looks for global options in the GOPTIONS statement. Such options are CTEXT= and FTITLE=. When titles and footnotes are rendered as part of the graphic image, SAS looks for options in the GOPTIONS statement. In this case using the ODS USEGOPTS statement has no effect.For more information, see “Using Graphics Options with ODS (USEGOPT)” on page 116.
3. SAS looks for information specified in the current style.

There are occasions when titles and footnotes are rendered as part of the ODS output file, and font sizes are specified as a percentage. These font sizes are interpreted as a percentage of the size specified by the current style. When titles and footnotes are rendered as part of the graphic file, fonts sizes that are specified as a percentage are interpreted as a percentage of the graphics output area. For more information about specifying fonts and font sizes, refer to the following topics:

- “FTEXT Graphics Option” on page 626 and “FTITLE Graphics Option” on page 627
- “HTEXT Graphics Option” on page 650 and “HTITLE Graphics Option” on page 651
- “GUNIT Graphics Option” on page 642
- “TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement” on page 507

11

SAS/GRAPH Output

Overview of SAS/GRAPH Output

About SAS/GRAPH Output

The result of most SAS/GRAPH procedures is the graphic display of data in the form of *graphics output*, which is distinct from *SAS output*. Whereas SAS output consists of text, graphics output consists of commands that tell a graphics device how to draw graphics elements. A *graphics element* is a visual element of graphics output (for example, a plot line, a bar, a footnote, the outline of a map area, or a border). This chapter discusses how to display, print, store, and export SAS/GRAPH output after you have created it.

SAS/GRAPH Output Terminology

The following terms are used when describing SAS/GRAPH output:

Graphics output file
: A file that contains bitmapped or vector graphic information. See “Supported Graphics Formats” on page 121.

Image file
: A file that contains bitmapped graphic information. Examples include PNG and GIF files. Image files are a subset of graphics output files.

Document file
: A file output by the Output Delivery System (ODS) that contains an image or is used to view an image. Examples include HTML, PDF, RTF, SVG, and PostScript files.

Supported Graphics Formats

You can export your SAS/GRAPH output in many different graphics file formats. SAS/GRAPH supports the following image file formats:

BMP	Windows Bitmap
GIF	Graphics Interchange Format
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
TIFF	Tagged Image Format File

SAS/GRAPH supports the following vector file formats:

CGM	Computer Graphics Metafile
EMF	Microsoft Enhanced Metafile
EPS	Encapsulated PostScript
PCL	Printer Control Language
PDF	Portable Document Format
PS	PostScript
SVG	Scalable Vector Graphics

The vector-based formats:

- are usually smaller than image files
- can be edited with third–party software (except for EPS)
- support system fonts
- support font embedding with the PDF, SVG, and PostScript devices
- provide a clear image on high-resolution devices.

The type of graphics file format that you choose depends on how you are going to use the output. For example, you are planning to import the graph into other software applications, such as Microsoft Excel, Word or PowerPoint, you might prefer to create an EMF file. The vector-based files are usually smaller than image files, they support TrueType fonts, and except for EPS, they can be edited with third-party software. In addition, they use device-resident fonts and provide a clear image on high-resolution devices.

If you want to display the graph on a web page, or import it into software that cannot accept vector graphics, you must create an image file such as PNG or GIF.

Most software applications that process graphics input can accept one or more of these file formats. Check the documentation for the hardware or software product to which you want to send the graph to determine what file formats it can use.

For a complete list of graphics file formats that are available with SAS/GRAPH in your operating environment, refer to the Device Help for SAS/GRAPH in the SAS Help facility.

Output Types

The SAS graphics procedures can generate the following types of output:

- a GRSEG (except for procedures GKPI, GTILE, and GAREABAR)
- a graphics output file that contains the graph (BMP, JPG, GIF, PNG, and so on)
- an HTML file that contains XML code that is consumed by the ActiveX control

In addition, the SAS Output Delivery System (ODS) creates document files, which include the following types of output:

- an HTML file that displays one or more graphs
- a PCL file that contains one or more graphs
- a PDF file that contains one or more graphs
- a PostScript file that contains one or more graphs
- an RTF file that contains one or more graphs
- an SVG file that contains one or more graphs

About GRSEGs

A GRSEG is a SAS catalog entry that contains graphics commands in a generic, device-independent format. There are few cases in which you would be concerned with the GRSEGs. One case for using the GRSEGs is when combining multiple graphs into a single graphics output file using the GREPLAY procedure. (See "Using the GREPLAY Procedure to Store Multiple Graphs in One Graphics Output File" on page 145.) Beyond this case, there are few reasons to use the GRSEGs. If you plan to use the GRSEGs, you must understand when they are generated and where they are stored.

GRSEGs are supported by the SAS/GRAPH procedures that use the graphics output devices with some exceptions. The procedures that are supported by only the Java and ActiveX devices, such as GKPI, GTILE, and GAREABAR, do not support GRSEGs.

A procedure that generates a GRSEG produces output in two steps:

1. It creates a GRSEG in a SAS catalog.
2. It uses a graphics output device to translate the commands from the GRSEG to commands that a particular graphics device understands. This is called device-dependent output.

This method enables you to produce graphics output on several types of graphics output devices.

A GRSEG is stored in a catalog in the SAS temporary directory. The graphics instructions that are contained in the GRSEG are understood only by the SAS/GRAPH software. You cannot use third-party graphics applications to view the graphic in a GRSEG. The SAS/GRAPH software provides devices that enable you

to convert a GRSEG to standard graphics formats such as GIF, PNG, and PDF, which you can view using third-party applications.

SAS/GRAPH software always assigns a name and a description to each GRSEG so that you can identify it. By default, the names and descriptions are determined by the procedure. For example, a GRSEG produced by the GCHART procedure is assigned the name GCHART and a description such as PIE CHART OF MONTH.

By default, SAS/GRAPH appends each new GRSEG to the catalog. If you create more than one graph with a procedure during a SAS session and the GRSEGs are stored in the same catalog, SAS/GRAPH software appends a number to the end of the name of subsequent GRSEGs. This number makes the names unique within the catalog. For example, if you create three graphs with the GCHART procedure during the same SAS session, the GRSEGs are named GCHART, GCHART1, and GCHART2. SAS/GRAPH software uses this naming convention whether GRSEGs are being stored in a temporary or permanent catalog.

You can supply a name and description when you create the graph by using the NAME= and DESCRIPTION= options. If you create more than one graph of the same name, the SAS/GRAPH software increments the specified name just as it does the default names.

What You Can Do with SAS/GRAPH Output

By default, output from SAS/GRAPH procedures that produce graphics output is sent to the ODS HTML destination using the HTMLBlue ODS style. In SAS Studio, the graphics output is sent to a special instance of the ODS HTML5, ODS RTF, and ODS PDF destinations. (See “Output in SAS Studio” on page 65.) Using the SAS ODS and the graphics options, you can direct graphics output to a variety of other destinations. Specifically, you can do the following with your graphics output:

- send it directly to a graphics hard-copy device, such as a printer. For details, see “Printing Your Graph” on page 151.
- save it in a temporary or permanent SAS catalog for later replay. See “Replaying Your SAS/GRAPH Output” on page 147.
- export it to a graphics output file using different graphics file formats. For example, you can save SAS/GRAPH output in formats such as CGM or PostScript for use with other software applications. For details, see “Exporting Your Output” on page 152.

Regardless of the destination of a graph, a GRSEG is created for those SAS/GRAPH procedures that support GRSEGs. The GRSEG is stored in the WORK.GSEG catalog unless you specify a different catalog with the GOUT= procedure option. To generate only GRSEGs and suppress all other forms of graphics output, use the NODISPLAY graphics option. See “DISPLAY, NODISPLAY Graphics Options” on page 614.

After your graphics output is saved in a catalog, you can do the following with your graphics:

- transport them in catalogs from one operating environment to another. For details, see Appendix 12, “Transporting and Converting Graphics Output,” on page 1725.

- convert them for use with a different version of SAS by converting the catalog containing the graphics output. For details, see “Converting Catalogs to a Different Version of SAS” on page 1729.
- export them to graphics output files using different graphics file formats. For details, see “Exporting Your Output” on page 152.

Specifying the Graphics Output File Type for Your Graph

About the Output Delivery System (ODS)

The SAS ODS sends your graph output to a default destination or a destination that you specify, such as your monitor, a printer, or a graphics output file. Each destination has a default style and graphics output device associated with it. You can use the STYLE= ODS option to specify a different style, and you can use the DEVICE= graphics option to specify a different device that is supported by the ODS destination that you are using.

See Chapter 10, “Managing Your Graphics with ODS,” on page 111 and Chapter 21, “Using ODS Styles, Device Parameters, and Options,” on page 321 for more information about using the ODS destinations, styles, and supported devices.

About the Graphics Output Devices

The Output That Each Device Generates

By default, the SAS/GRAPH ODS writes to the HTML destination, which displays your graph in a web browser window and creates a GRSEG in the catalog. You can specify a graphics output device other than the default PNG device for the ODS HTML destination, or you can specify a different ODS destination and device. For information about using ODS to manage your graphics output, see Chapter 10, “Managing Your Graphics with ODS,” on page 111.

The following table lists the common graphics output devices, and the default output that each generates.

Table 11.1 SAS/GRAPH Devices and the Output They Generate

Device	External files
ACTIVEX[1, 2, 3, 4, 5]	In SAS 9.4M7 and prior releases, an HTML file or an RTF file that contains XML code that is consumed by the SAS/GRAPH ActiveX control. This device must be used with the ODS HTML or ODS RTF destination. When the HTML or RTF file is viewed, the SAS/GRAPH output is displayed as an interactive ActiveX control. The user must install the ActiveX control to view the output of the ACTIVEX device. This device and the ACTXIMG device together are often referred to as ActiveX devices in this document.
ACTXIMG[1, 2]	In SAS 9.4M7 and prior releases, a PNG file that contains an image of the graph that is generated with the ACTIVEX device. The user does not need to install the SAS/GRAPH ActiveX control to view the output from the ACTXIMG device. This device and the ACTIVEX device together are often referred to as ActiveX devices in this document.
BMP	A BMP file that contains the graph.
CGM	A CGM file that contains the graph.
CGMOF97L	A CGM file suitable for inserting into Microsoft Word or PowerPoint presentations.
EMF	An EMF file that contains the graph. The file contains EMF Plus language extensions.
EMFDUAL	An EMF file that contains the graph. The file contains both traditional EMF and EMF Plus language extensions.
GIF	A GIF file that contains the graph.
IBMPCGX	Display device. This device is available on z/OS hosts only.
JAVAIMG[2]	A PNG file that contains a graph that is generated using Java technology. This device together with other devices is sometimes referred to as a Java device in this document.
JPEG	A JPG file that contains the graph. You can use the JPEGQUALITY= system option to control the image quality and compression of JPEG images. See *SAS System Options: Reference* for more information.
PCL5	A PCL file that contains one or more graphs and tables.
PDF	A PDF file that contains one or more graphs and tables.
PNG	A PNG file that contains the graph. This is often the default driver when using SAS both in Batch mode and in the window environment.

Device	External files
PSCOLOR	A PostScript file that contains one or more graphs.
PSL	A PostScript file that contains the graph in gray scale.
SASEMF	An EMF file that contains the graph. The file contains traditional EMF language.
SVG	An SVG file that contains the graph.
TIFF	A TIFF file that contains the graph in color. This device uses RGBA colors.
TIFFK	A TIFF file that contains the graph in color. This device uses CMYK colors.
WIN	Display device. This device is available on Windows hosts only.
XCOLOR	Display device. This device is available on UNIX hosts only.

1 Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

2 In the z/OS operating environment, starting with SAS 9.4M8, this device is no longer supported. If you attempt to use this device on z/OS, an error results.

3 For the RTF destination, see Problem Note 65142: Creating an RTF file by using the ACTIVEX device driver results in an RTF file that Microsoft Word cannot open.

4 For the HTML destination, see Usage Note 57554: Displaying an ActiveX graph in Microsoft Internet Explorer might result in a BEX error after the application of Microsoft patch KB3000850.

5 For detailed information about the ACTIVEX device, see Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

Graphics Output Files

When you export SAS/GRAPH output, you run the output through a device driver that creates a graphics output file. A graphics output file is a file that contains vector or bitmap graphics commands. Typically, you select a device that produces the type of graphics file format that you want, such as PNG, CGM, PS or EPS, GIF, or TIFF. You can select a device that sends the output directly to a printer or other hard-copy device without creating a graphics output file. You can specify the exact name and location of each file or assign a default location to which all files are sent.

You can also use the ODS to generate SAS/GRAPH output as HTML that you can view with a web browser. Details are discussed in Chapter 10, “Managing Your Graphics with ODS,” on page 111.

Once you have created a graphics output file, you can do the following:

- print the file using host commands
- view the file with an appropriate viewer or browser
- edit the file with the appropriate editing software
- import the file into other software applications

Note: A graphics output file is different from a SAS/GRAPH GRSEG. A graphics output file is a file that is independent of SAS, and a GRSEG is a type of SAS catalog file. Consequently, you use host commands to manipulate a graphics output file independent of SAS, whereas you must use SAS to manipulate SAS GRSEGs. The GREPLAY procedure can be used to replay graph entries stored in catalogs and display them in the GRAPH window.

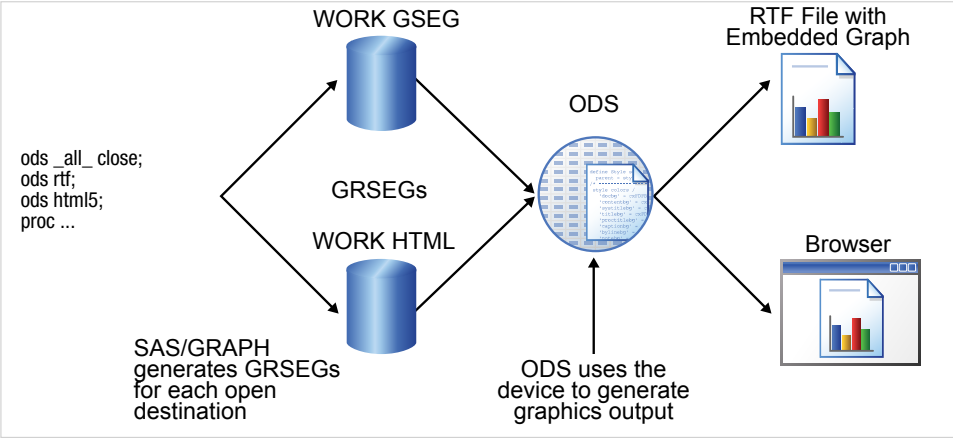
About File Extensions

When you send SAS/GRAPH output to an aggregate file storage location, SAS/GRAPH generates the name of the graphics output file. This is done by taking the GRSEG name and adding the appropriate file extension. Most devices provide a default extension. If a device does not generate an extension, then SAS/GRAPH uses the default extension .gsf. To specify a different extension from the one SAS/GRAPH provides, use the EXTENSION= graphics option. (For details, see "EXTENSION Graphics Option" on page 619).

The SAS/GRAPH Output Process

All Devices except JAVAIMG, ACTIVEX, and ACTXIMG

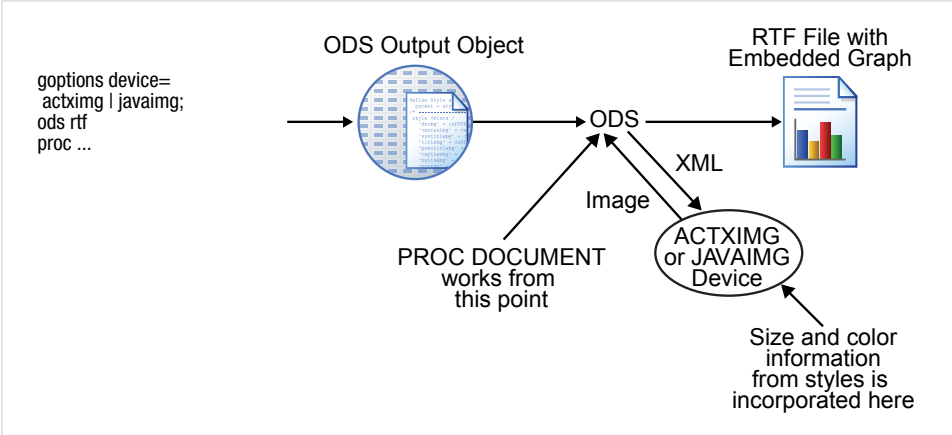
The following diagram illustrates the output process for all of the SAS/GRAPH graphics output devices except the Java and ActiveX devices.



Note: The image size, color, and font information is obtained from the device entry and incorporated into the GRSEG.

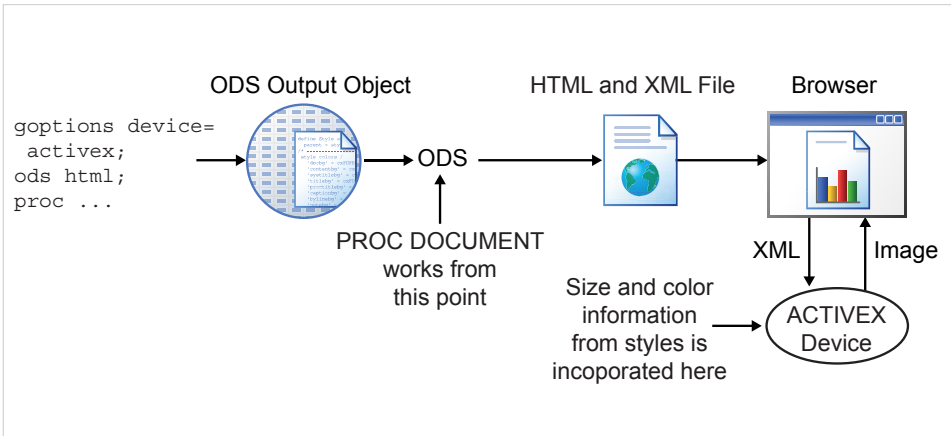
JAVAIMG or ACTXIMG Device

The following diagram illustrates the output process for the JAVAIMG and ACTXIMG graphics output devices.



The Output Process for the ACTIVEX Graphics Output Device

The following diagram illustrates the output process for the ACTIVEX graphics output device.



Note: For information about the ACTIVEX device, see Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

Setting the Size of Your Graph

Using the HSIZE= and VSIZE= Graphics Options to Set the Size of Your Graphics Area

You can use the HSIZE= and VSIZE= graphics options to change the default size of the graphics area for the device that you are using. The HSIZE= option sets the horizontal dimension while the VSIZE= option sets the vertical dimension. You can specify the dimension in inches (in), centimeters (cm), or points (pt). The default unit is inches (in).

Note: For the ODS PRINTER, ODS PDF, ODS PS, and ODS PCL destinations, the page area (easel) is determined by the printer's default paper size or by the PAPERSIZE= option.

Here is an example that creates a 15 centimeter wide by 10 centimeter high graph.

```
goptions reset=all hsize=15cm vsize=10cm;
proc gchart data=sashelp.cars;
   vbar Make
      where MPG_Highway >= 37;
   run;
quit;
```

Using the XPIXELS= and YPIXELS= Graphics Options to Set the Size of Your Graph

When you use a display device , Universal Printer device, or Universal Printer shortcut device with the ODS LISTING, ODS HTML, or ODS RTF destination, you can use the XPIXELS= and YPIXELS= graphics options to set the size of the graph without having to modify the device. Setting only the XPIXELS= and YPIXELS= options affects the size of the graph, but does not affect the resolution.

Note: The XPIXELS= and YPIXELS= graphics options are partially supported by the ACTIVEX device.

The ODS PRINTER, ODS PDF, ODS PS, and ODS PCL destinations do not honor the XPIXELS= and YPIXELS= graphics options. For these destinations, you must use the HSIZE= and VSIZE= graphics options to set the graph size. See “Using the HSIZE= and VSIZE= Graphics Options to Set the Size of Your Graphics Area” on page 129.

Here is an example that creates a 600 pixel wide by 800 pixel high graph.

```
goptions reset=all xpixels=600 ypixels=800;
proc gchart data=sashelp.cars;
   vbar Make;
      where MPG_Highway >= 37;
   run;
quit;
```

Notice that XMAX= and YMAX= are not set. In this example, the SAS/GRAPH software recomputes the XMAX= and YMAX= values to retain the original resolution for the new graph size.

Setting the Resolution of Your Graph

About Setting the Resolution

To set the resolution of your device-based graphics, use one of the following methods:

- Use the IMAGE_DPI= option in an ODS LISTING or ODS HTML statement to set the image resolution, and specify one of the following Universal Printer shortcut devices: PNG, PNGT, PNG300, SVG, SVGT, SVGVIEW, or SVGZ. The default resolution for the PNG300 device is 300 DPI while the default resolution for the remaining devices that are listed is 96 DPI.

  Note: Devices other than those listed do not honor the IMAGE_DPI= option.

- Use the IMAGE_DPI= option in an ODS RTF statement to set the image resolution, and specify one of the following Universal Printer shortcut devices: PNG, PNGT, or PNG300.
- Use the DPI= option in an ODS PDF, ODS PCL, ODS PS, or ODS PRINTER statement to set the image resolution. The default resolution for the ODS PDF, ODS PCL, and ODS PS destinations is 150 DPI. The default resolution for the ODS PRINTER destination is the default resolution of the printer that is used. For example, the ODS PRINTER destination produces output at 300 DPI by default when the PRINTER=PDF option is used, and it produces output at 96 DPI by default when the PRINTER=PNG option is used.
- Use a traditional SAS/GRAPH device and specify the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options to set the resolution for graphics formats that support variable resolution. See “Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” on page 131.
- Use a device variant to set the resolution of your graph to a specific resolution. See “Using a Device Variant to Set the Size or Resolution of Your Graph” on page 132.

Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices

For the traditional SAS/GRAPH devices, you can use the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options to set the resolution of your graph. Note the following:

- The XPIXELS=, YPIXELS=, XMAX=, and YMAX= graphics options are not supported by the default display devices and the Universal Printer devices, including the shortcut devices. These graphics options are partially supported by the ACTIVEX device.
- The resolution of GIF and BMP images is fixed and cannot be changed using this method.

The XPIXELS= and YPIXELS= graphics options set the number of pixels for the X and Y axes respectively. The XMAX= and YMAX= graphics options set the maximum boundaries of the output on the X and Y axes respectively. The SAS/GRAPH software computes the resolution as follows:

X-resolution = XPIXELS / XMAX

Y-resolution = YPIXELS / YMAX

Table 11.2 on page 131 summarizes the effect of the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options have on the image resolution.

Table 11.2 *Interactions of Graphics Options That Affect Resolution*

Options Specified	Options Not Specified	SAS/GRAPH Action
XPIXELS= and YPIXELS=	XMAX= and YMAX=	Changes the dimensions and recalculates the value of XMAX= and YMAX= in order to retain the resolution.
XMAX= and YMAX=	XPIXELS= and YPIXELS=	Changes the dimensions and recalculates the value of XPIXELS= and YPIXELS= in order to retain the resolution.
XMAX= and XPIXELS=		Changes the horizontal dimension and recalculates the resolution.
YMAX= and YPIXELS=		Changes the vertical dimension and recalculates the resolution.

For example, for the graphics option settings XPIXELS=800 and XMAX=8in, the resulting X resolution is 100 DPI.

You can set the X resolution, the Y resolution, or both. Here is an example that sets the resolution of a 1000-pixel-wide-by-1200-pixel-high TIFF image of a graph to 200 DPI.

```
ods _all_ close;
ods listing style=seaside;
goptions reset=all device=tiffp xpixels=1000 xmax=5in ypixels=1200
ymax=6in;
proc gchart data=sashelp.cars;
   vbar Make;
      where MPG_Highway >= 37;
   run;
quit;
ods listing close;
ods html;  /* Not required in SAS Studio */
```

Using a Device Variant to Set the Size or Resolution of Your Graph

Some of the graphics output devices have variants that produce graphics of a specific size or resolution for a given format. For example, the PNG300 and JPEG300 device variants produce 300 DPI images in the PNG and JPEG format respectively.

Note: The PNG300 and JPEG300 devices are not appropriate for use with the ODS HTML destination. These devices are used when a high-resolution graph (300 DPI) in the PNG or JPEG format is required for printing purposes. Because most browsers do not use the resolution value stored in the PNG or JPEG file, images produced by the PNG300 and JPEG300 devices appear very large when they are viewed in the browser.

See “Overview: Using Graphics Devices” on page 90.

Controlling Where Your Output Is Stored

About the ODS Default Output Directory

SAS Windowing Environment

With the exception of the ODS HTML destination, by default, ODS output is stored in the current SAS output directory, which is displayed on the status bar of the SAS window. A special case applies to the ODS HTML destination. When you start a new SAS session, the ODS HTML destination is open by default. This initial instance of the ODS HTML destination sends its output to the SAS WORK directory instead of the current directory. To find the path to the WORK directory, in the SAS Explorer window, you can right-click **Work** in the Libraries folder, and then select **Properties** from the pop-up menu.

When you close the initial instance of the ODS HTML destination and open a new instance, the output of the new instance and all other instances that are opened in that SAS session is sent to the current directory instead of the WORK directory. It is important that you keep this behavior in mind so that you can locate your ODS output, especially when using the ODS HTML destination.

SAS Studio

In SAS Studio, by default, output is generated in the HTML5, PDF, and RTF formats. You can download the output from SAS Studio to a file on your local machine or to a specified location. If you open your own ODS destination in SAS Studio, by default, the output is stored in your working directory. See “Output in SAS Studio” on page 65 for details.

Specifying the Name and Location of Your ODS Output

You can use the FILE= option in your ODS statement to specify where your ODS output files are stored. For the HTML destination, you can also use the PATH=, GPATH=, and the BODY= options to specify a different location for the HTML output file and the graphics output files. Here is an example that uses the FILE= ODS option with the PDF destination to send the PDF output to file mygraph.pdf.

```
goptions reset=all;
```

```
ods _all_ close;
ods pdf file=".\mygraph.pdf";
proc gchart data=sashelp.prdsale;
   vbar Product / sumvar=actual;
      title1 "First Quarter Sales in Canada";
      where Quarter=1 and Country="CANADA";
   run;
quit;
ods pdf close;
ods html; /* Not required in SAS Studio */
```

Here is an example that uses the PATH=, GPATH=, and the BODY= ODS options with the HTML destination to send the HTML output to file mygraph.html in the output directory, and the graphics output file to the **images** subdirectory. You can open the resulting HTML output file in your web browser.

```
/* Specify the ODS output path */
filename odsout ".";
goptions reset=all;
ods _all_ close;
ods html                  /* The PNG device is used by default */
   path=odsout (url=none) /* HTML output directory */
     gpath="./images"       /* Graphics output file location */
     body="mygraph.html";   /* HTML filename */
proc gchart data=sashelp.prdsale; /* Run procedure to generate graphs
*/
   vbar Product / sumvar=actual;
      title1 "First Quarter Sales in Canada";
      where Quarter=1 and Country="CANADA";
   run;
quit;
ods html close; /* Close the output file */
ods html;       /* Not required in SAS Studio */
```

You can use BY statements to create multiple graphs.

You can specify a device other than PNG with the DEVICE= option in the GOPTIONS statement.

Note: The **./images** subdirectory must already exist.

Note: With the SVG, PNG, GIF, and TIFF device, footnotes and titles are stored in the graphics output file by default. To move footnotes and titles out of the graphics output file and into the HTML file, specify the ODS HTML options NOGTITLE or NOGFOOTNOTE or both. See "Controlling Titles and Footnotes" on page 115.

For more information about the PATH=, GPATH=, and BODY= options, see *SAS Output Delivery System: User's Guide*.

Specifying the Name and Location of Your Graphics Output Files

When Graphics Output Files Are Created

Many of the graphics devices such as PNG and GIF write their graphics output to a disk file when used with the ODS LISTING or ODS HTML destination. (The ODS HTML destination uses the PNG device by default.) For the remaining ODS destinations, the graphics output is embedded in the ODS output. In that case, no graphics output file is created. For more information, see “About the Graphics Output Devices ” on page 124.

For the ODS LISTING and ODS HTML destinations, options are available that enable you to control the location and name of the graphics output files.

Controlling Graphics Output for ODS LISTING

For the ODS LISTING destination, you can use the GSFNAME= graphics option to specify the location of the output file. The GSFNAME= option requires a FILENAME statement that creates a file reference that points to a file or an aggregate file storage location. The syntax of the FILENAME statement is as follows:

```
FILENAME RefName  "DirectoryOrFile";
```

If the file reference points to an aggregate file storage location, the graphics output files are named according to the NAME= option, if specified, or the default naming convention. If the file reference points to a file, the file specified in the FILENAME statement is used, even if the NAME= option is specified. See “Summary of How Output Filenames and GRSEG Names Are Handled” on page 141.

Note: In SAS 9.4M6 and earlier releases, only the PNG and JPEG graphics devices support file references to the SAS Viya Files service (FILESRVC access method). Starting with SAS Viya 3.5 and SAS Studio 5.2, the GIF and SVG graphics devices also support FILESRVC file references.

Here is an example that shows how to send the output of the GCHART procedure to file mychart.png in subdirectory MyGraphs in the current working directory.

```
filename graphout "MyGraphs";
goptions reset=all device=png gsfname=graphout;
ods _all_ close;
ods listing;
proc gchart data=sashelp.cars;
   pie Make / name="MYCHART";
      where MSRP <= 15000;
```

```
   run;
quit;
ods listing close;
ods html; /* Not required in SAS Studio */
```

Note: The target directory MyGraphs should already exist. Otherwise, SAS creates a file named MyGraphs.

If a MYCHART GRSEG entry does not already exist in the temporary catalog, the device sends the output to file mychart.png in the Mygraphs directory. If a MYCHART GRSEG entry already exists, the device uses an incremented name such as MYCHART1. In the previous example, you can replace the aggregate file location with a file name in the FILENAME statement and omit the NAME= option and get the same result.

If you specify the file name in the FILENAME statement, you must include the proper file extension. See "About File Extensions" on page 127.

You can also store your output in a graphics output file on a remote host using FTP. Here is an example that uses FTP to store multiple PNG graphs in directory **/public/sas/graphs** on the remote UNIX host unixhost73.

```
filename grafout ftp "/public/sas/graphs" dir host="unixhost73" fileext
   user="anonymous";
ods _all_ close;
ods listing;
goptions reset=all device=png gsfname=grafout;

/* Create our data set by sorting sashelp.cars by type */
proc sort data=sashelp.cars out=work.cars;
   by type;
run;

/* Generate the graphs */
proc gchart data=work.cars;
   vbar Make;
      title1 "30 MPG or Better";
      where MPG_Highway >= 30;
      by type;
   run;
quit;
ods listing close;
ods html; /* Not required in SAS Studio */
```

This example creates four PNG files in directory **/public/sas/graphs** on host unixhost73. Since the GCHART procedure uses BY-group processing, the FILENAME statement includes the DIR option, which defines an aggregate file storage location. If you need to create only one graph, remove the DIR option and specify the absolute path to your graphics output file in your FILENAME statement.

Controlling Graphics Output for ODS HTML

For the ODS HTML destination, you can use the GPATH= option in the ODS HTML statement to specify the location of the graphics output files and the plot statement NAME= option to specify the base name. Here is an example that shows how to send the output of the GCHART procedure to file mychart.svg in the MyGraphs directory.

```
/* Specify the ODS output path */
filename odsout ".";
goptions reset=all device=svg;
ods _all_ close;
ods html path=odsout file="mychart.html" gpath="MyGraphs";
proc gchart data=sashelp.cars;
   pie Make / name="MYCHART";
      where MSRP <= 15000;
   run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */
```

About Filename Indexing

When duplicate names occur in graphics output file names, SAS/GRAPH procedures use indexing systems to determine unique names for new graphics output files. (Numbers are added to the end of the file name to create new file names). Two indexing systems are used: ODS Graphics indexing and catalog-based indexing. ODS Graphics indexing is used in all ODS Graphics output and by the procedures listed in Table 11.3 on page 137. All of the other procedures use catalog-based indexing.

Table 11.3 *Filename Indexing Systems Used by SAS/GRAPH Procedures*

Indexing System	How To Control Graphics Filenames	Procedure Name
Catalog-based	NAME= option in the procedure action statement	All procedures not listed below.
ODS Statistical Graphics		GAREABAR GKPI GTILE

Because two independent indexing systems are used by the SAS/GRAPH procedures, it is possible that graphics output files can be overwritten if the same graphics file name both for procedures that use catalog-based indexing and for

procedures that use ODS Graphics indexing. To avoid this problem, make sure that you specify different names for the procedures that use ODS Graphics indexing and the procedures that use catalog-based indexing. For example, if your application uses both the GCHART procedure and the GAREABAR procedure, and you are using the NAME= option to specify output file names, make sure you specify different file names for each procedure.

The current index value for both catalog-based indexing and ODS Graphics indexing can be reset. To reset the index value for catalog-based indexing, delete the entries from the GRSEG catalog. To reset the index value for ODS Graphics indexing, use the following statement:

```
ods graphics / reset=index;
```

In both cases, after the index value is reset, the new output overwrites output files that already exist.

Specifying the Catalog Name and Entry Name for Your GRSEGs

Using the Default Catalog and Entry Name

If you omit the NAME= and GOUT= options, the SAS/GRAPH software uses the default naming convention to name the GRSEG entry and stores the entry in the default WORK.GSEG catalog. The GRSEG naming convention uses up to eight characters of the default name for the procedure as the base name for the GRSEG. If the name generated by the procedure duplicates an existing GRSEG, the name is incremented such as GCHART, GCHART1, GCHART2, and so on. For details, see the description of the NAME= option for a specific procedure.

If you specify a file name for the graphics output file and omit the NAME= option, the graphics output file name is the name specified in the FILENAME statement, and the GRSEG entry name is the default procedure name. When you specify the file name, make sure that you include the appropriate file extension, such as .gif or .ps.

If you specify an aggregate file storage location instead of a specific file name and you omit the NAME= option, the name of both the GRSEG entry and the graphics output file is the default procedure name, and SAS/GRAPH supplies the appropriate file extension.

See “Summary of How Output Filenames and GRSEG Names Are Handled” on page 141.

Specifying a Name for Your GRSEG with the NAME= Option

You can use the NAME= option to change the name of your output. Here is an example that shows how to change the name of the GCHART procedure output to MYCHART.

```
filename outfile "./";
goptions reset=all device=png gsfname=outfile;
ods _all_ close;
ods listing;
proc gchart data=sashelp.cars;
   pie Make / name="MYCHART";
      where MSRP <= 15000;
   run;
quit;
ods listing close;
ods html; /* Not required in SAS Studio */
```

This example creates the file mychart.png in the SAS default output directory, and it creates the GRSEG Mychart in the SAS temporary catalog.

See "Summary of How Output Filenames and GRSEG Names Are Handled" on page 141 for additional information about output naming.

Specifying the Catalog and GRSEG Name with the GOUT= and NAME= Options

By default, GRSEGs are stored in the WORK.GSEG temporary catalog under the default name of the procedure that was used to generate the graph. The GRSEG name can be specified using the NAME= option, and the output catalog can be changed using the GOUT= procedure option. GRSEG names are limited to eight characters. If the NAME= option is set to a name that is more than eight characters in length, the GRSEG name is truncated to eight characters.

The name of the library and catalog in which the GRSEG is stored can be changed with the GOUT= procedure option. The GOUT= procedure option is assigned the catalog name in the format *libref.catalog* for the desired catalog. The name can be a one-level or two-level name. If a one-level name is used, the GRSEG is stored in the temporary WORK library under the specified catalog name. A two-level name can be used to specify a permanent catalog.

Here is an example that shows how to store a GRSEG generated by the GCHART procedure under entry MYCHART in the MYGRAPHS.CARS catalog.

```
LIBNAME Mygraphs "Mygraphs";
proc gchart data=sashelp.cars gout=Mygraphs.cars;
   vbar Make / name="Mychart";
      where MPG_Highway >= 37;
   run;
quit;
```

Table 11.4 on page 139 summarizes the location of the GRSEG based on the NAME= and GOUT= procedure using the GCHART procedure as an example.

Table 11.4 *How NAME= and GOUT= Affect the GRSEG Location*

NAME=	GOUT=	GRSEG Location
Not specified	Not specified	Gchart in WORK.GSEG

NAME=	GOUT=	GRSEG Location
Not specified	CARS	Gchart in WORK.CARS
Not specified	MYGRAPHS.CARS	Gchart in MYGRAPHS.CARS
MYCHART	Not specified	Mychart in WORK.GSEG
MYCHART	CARS	Mychart in WORK.CARS
MYCHART	MYGRAPHS.CARS	Mychart in MYGRAPHS.CARS

Where GRSEGs Are Stored When Multiple ODS Destinations Are Used

When you send output to multiple ODS destinations, a catalog is created for the GRSEGs for each of the destinations. If the GOUT= procedure option is not specified, by default, the GRSEGs for the first destination that was opened are sent to the WORK.GSEG catalog. The GRSEGs for the subsequently opened ODS destinations are sent to a catalog that is named after the destination itself. For example, if you open the ODS LISTING, HTML, and RTF destinations, in that order, the GRSEGs are stored in the catalogs that are shown in the following table.

Table 11.5 *Catalog Names for Default Catalog and Multiple ODS Destinations*

Catalog Name	Content
WORK.GSEG	The GRSEGs for ODS LISTING
WORK.HTML	The GRSEGs for ODS HTML
WORK.RTF	The GRSEGs for ODS RTF

In the default case, the GRSEGs for the first destination that is opened are stored in the WORK.GSEG catalog, regardless of the destination.

If you use the GOUT= procedure option to specify a catalog name, the GRSEGs for the first destination that you opened are sent to the catalog that is specified by the GOUT= procedure option. The GRSEGs for the subsequently opened ODS destinations are sent to a catalog that is named after the destination itself. For example, if you open the ODS HTML, LISTING, and RTF destinations, and you use the GOUT=MyGraphs.Sales procedure option, the GRSEGs are stored in the catalogs that are shown in the following table.

Table 11.6 Catalog Names for User-Specified Catalog and Multiple ODS Destinations

Catalog Name	Content
MYGRAPHS.SALES	The GRSEGs for ODS HTML
MYGRAPHS.LISTING	The GRSEGs for ODS LISTING
MYGRAPHS.RTF	The GRSEGs for ODS RTF

The GRSEGs for the first destination are stored in the catalog that is specified by the GOUT= procedure option.

Summary of How Output Filenames and GRSEG Names Are Handled

Table 11.7 on page 141 summarizes how SAS/GRAPH generates names for catalog entries and graphics output files, depending on 1) whether the NAME= option is used, and 2) the file reference specification in the FILENAME statement. This illustration assumes that the GCHART procedure is used with the DEVICE=GIF graphics option. It describes the case where a GRSEG and output file of the same name do not already exist, and the case where they do already exist.

Table 11.7 How SAS/GRAPH Generates Initial GRSEG Names and Filenames

NAME=	Condition	Result
NAME="FRED"	GSFNAME= points to a file named "MYGRAPH.GIF" and the catalog is empty.	GRSEG name: FRED external file name: MYGRAPH.GIF
NAME="FRED"	GSFNAME= points to an aggregate file storage location and the catalog is empty.	GRSEG name: FRED external file name: FRED.GIF
NAME="MYIMAGES"	The NAME= value is exactly eight characters in length. GSFNAME= points to an aggregate file storage location and the catalog is empty.	GRSEG name:MYIMAGES external file name:MYIMAGES.GIF
NAME="WEATHEROBS"	GSFNAME= points to an aggregate file storage location and the catalog is empty.	GRSEG name:WEATHERO

NAME=	Condition	Result
		external file name: WEATHEROBS.GIF
NAME= (not specified)	GSFNAME= points to a file named "MYGRAPH.GIF" and the catalog is empty.	GRSEG name: GCHART external file name: MYGRAPH.GIF
NAME= (not specified)	GSFNAME= points to an aggregate file storage location and the catalog is empty.	GRSEG name: GCHART external file name: GCHART.GIF

Note: When the file reference points to an aggregate file storage location, the name of the GRSEG *always* determines the name of the graphics output file. It does not matter whether the GRSEG name is the default name or a name assigned by the NAME= option.

CAUTION
If the graph created by your program already exists in the catalog, a new GRSEG with an incremented name is created. A new graphics output file might be created, which leaves your old graphics output file in place.

Although GRSEG names cannot be more than eight characters in length, the NAME= option supports long names. When the NAME= option is assigned a name of more than eight characters and the file reference points to an aggregate file location, the GRSEG name is the NAME= value truncated to eight characters, and the graphics output file name is the complete NAME= value. This is demonstrated by the NAME="WEATHEROBS" example in Table 11.7 on page 141.

When a GRSEG of the same name already exists in the catalog, the SAS/GRAPH software combines the NAME= option value with a number to create an incremented name of no more than eight characters. If the GSFNAME= graphics option is used and the file reference points to an aggregate file location, the new graphics output file name is also incremented using the same number that is used for the GRSEG name. The way in which the incremented graphics output file name is formed depends on the length of the NAME= value as follows:

- If the length of the NAME= value is less than eight characters or more than eight characters, the number that is used in the GRSEG name is appended to the NAME= value.
- If the length of the NAME= value is exactly eight characters, the number that is used in the GRSEG name is appended to the first seven characters of the NAME= value.

If the GSFNAME= graphics option points to a file, the graphics output file name remains the same and the original file is replaced with the new graph by default.

Table 11.8 on page 143 demonstrates how the SAS/GRAPH software increments the GRSEG name and the graphics output file names when a GRSEG and graphics output file of the same name already exist.

Table 11.8 How SAS/GRAPH Increments GRSEG Names and Filenames

NAME=	Condition	Result
NAME="FRED"	GSFNAME= points to a file named "MYGRAPH.GIF" and GRSEG FRED already exists.	GRSEG name: FRED1 external file name: MYGRAPH.GIF
NAME="FRED"	GSFNAME= points to an aggregate file storage location and GRSEG FRED already exists.	GRSEG name: FRED1 external file name: FRED1.GIF
NAME="MYIMAGES"	The NAME= value is exactly eight characters in length. GSFNAME= points to an aggregate file storage location and GRSEG MYIMAGES already exists.	GRSEG name:MYIMAGE1 external file name:MYIMAGE1.GIF
NAME="WEATHEROBS"	GSFNAME= points to an aggregate file storage location and GRSEG WEATHERO already exists.	GRSEG name:WEATHER1 external file name: WEATHEROBS1.GIF
NAME= (not specified)	GSFNAME= points to a file named "MYGRAPH.GIF" and GRSEG GCHART already exists.	GRSEG name: GCHART1 external file name: MYGRAPH.GIF
NAME= (not specified)	GSFNAME= points to an aggregate file storage location and GRSEGs GCHART and GCHART1 already exist.	GRSEG name: GCHART2 external file name: GCHART2.GIF

Notice that in the NAME="MYIMAGES" example the incremented external file name is formed by appending the number to the first seven characters of the NAME= value, while in the NAME="WEATHEROBS" and NAME="FRED" examples, the incremented external file name is formed by appending the number to the full NAME= value. Be aware that the way in which the incremented graphics output file name is formed is different when the length of the NAME= value is exactly eight characters than when it is more or less than eight characters.

You cannot replace individual GRSEGs in a catalog. To replace a GRSEG, you must delete the GRSEG, and then re-create it. Therefore, even though the contents of the

graphics output file are replaced, the GRSEG is not. Each time you submit the program, a new GRSEG is created, and the GRSEG name is incremented.

Replacing an Existing Graphics Output File Using the GSFMODE= Graphics Option

You can use the GSFMODE= graphics option to replace an existing graphics output file with a new graph. To replace an existing graphics output file, the GSFMODE= option must be set to REPLACE, which is the default value for this option. When you run a SAS program that creates a graphics output file and the graphics option GSFMODE=REPLACE is used, the existing graphics output file is replaced with the new graph. However, a unique GRSEG is still generated each time you run the procedure.

See “GSFMODE Device Parameter and Graphics Option” on page 638.

Storing Multiple Graphs in a Single Graphics Output File

Using Graphics Options to Store Multiple Graphs in One Graphics Output File

You can use the GSFMODE=APPEND and the GSFNAME= graphics options to store multiple graphs in one graphics output file. When the GSFMODE= graphics option is set to APPEND and the GSFNAME= option points to a file, if the graphics output file specified by the GSFNAME= option already exists, the SAS/GRAPH software appends the new graph to the graphics output file. Otherwise, it creates the graphics output file and stores the graph in it.

Note: Although a file can contain multiple graphs, some viewers can view only one graph. This can make it appear that a file containing multiple graphs contains only one graph.

The GSFMODE=APPEND option is useful only with the GIF graphics output device, and the PCL and PostScript Universal Printer devices. When used with other devices, such as PNG, SVG, or PDF, only the first graph is displayed in the viewer.

A common application of the GSFMODE=APPEND option is in the production of animated GIFs. See “Animating GIF Images in SAS/GRAPH” on page 228.

Using the GREPLAY Procedure to Store Multiple Graphs in One Graphics Output File

You can use the GOUT= procedure option with the GREPLAY procedure to store multiple graphs in one graphics output file. This involves the following steps:

1. Create a file reference for your output file. For example:

   ```
   filename myfile "MyOutputFile.pdf";
   ```

2. Run the procedure to generate your charts and store them in a catalog.

3. Add the GSFNAME=*FileRefName* to your GOPTIONS statement.

4. Run the GREPLAY procedure as follows:

   ```
   proc greplay
      igout=<CatalogName>
         replay _all_;
      run;
   quit;
   ```

   Replace <CatalogName> with the name of the catalog in which your graphs are stored. The REPLAY _ALL_ action statement replays all of the entries in the catalog.

Here is an example that replays five graphs to one PDF file for printing.

```
/* Specify graphics output file name */
filename pdfout "multicharts.pdf";

/* Specify the graphics options */
goptions reset=all device=png nodisplay;

/* Generate the graphs */
proc gchart data=sashelp.cars gout=Work.Mygraphs;
   vbar Make;
      title1 "30 MPG or better";
      where MPG_Highway > 30;
   run;

      vbar Make;
      title1 "Between 25 MPG and 29 MPG";
      where MPG_Highway >= 25 AND MPG_Highway <= 29;
   run;

   vbar Make;
      title1 "Between 20 MPG and 24 MPG";
      where MPG_Highway >= 20 AND MPG_Highway <= 24;
   run;

   vbar Make;
      title1 "Between 15 MPG and 19 MPG";
```

```
      where MPG_Highway >= 15 AND MPG_Highway <= 19;
   run;

   vbar Make;
      title1 "Less than 15 MPG";
      where MPG_Highway < 15;
   run;
quit;

/* List the graphs in Work.Mygraphs */
proc greplay igout=Work.Mygraphs nofs;
   list igout;
run;
quit;

/* Enable display, and then replay all of the graphs to psout */
goptions display;

/* Close the currently open destinations */
ods _all_ close;

/* Open ODS PDF */
ods pdf file=pdfout;

/* Replay the graphs */
proc greplay
   igout=Work.Mygraphs nofs;
      replay _all_;
   run;
quit;

/* Close the PDF destination and open the HTML destination */
ods pdf close;

/* Clear fileref pdfout */
filename pdfout clear;

/* Clear catalog Work.Mygraphs */
proc greplay igout=Work.Mygraphs nofs;
   delete _all_;
run;
quit;

ods html; /* Not required in SAS Studio */
```

Replaying Your SAS/GRAPH Output

Replaying Your Output Using the GREPLAY Procedure

For the SAS/GRAPH procedures that support GRSEGs, you can use the GREPLAY procedure to replay your graph GRSEGs without having to rerun your DATA step and procedures. You can replay all of your graphs or only the ones that you select. When you replay your graphs, use the same device that you used when you generated the original graphs. If you use a different device, your replayed graphs might be distorted.

You can replay your graphs to the GRAPH window for viewing or to a graphics output file. Here is an example that replays all of the graphs in the WORK.GSEG catalog for viewing:

```
goptions reset=all;
proc greplay igout=work.gseg nofs;
   replay _all_;
run;
quit;
```

You can also use the GREPLAY procedure to replay multiple graphs to a single file for the graphic and document formats that support multiple images per file. See "Using the GREPLAY Procedure to Store Multiple Graphs in One Graphics Output File" on page 145 and Chapter 16, "Generating Animations," on page 227.

For information about the GREPLAY procedure, see Chapter 44, "GREPLAY Procedure," on page 1415.

Replaying Output Using the DOCUMENT Procedure

About the DOCUMENT Procedure

For all of the SAS/GRAPH procedures, you can use the DOCUMENT procedure to replay output that you created. Use the ODS DOCUMENT destination, without having to rerun your DATA step and procedures. The ODS DOCUMENT destination creates ODS output objects for your output. You can replay the output objects at any time to your monitor or to a different device.

Creating Your ODS Document

To create an ODS document for your output, do the following in your SAS program:

1. Open ODS DOCUMENT and specify the name of the output catalog with Write permissions.
2. Close ODS HTML.
3. Open the ODS destinations that you want to send your output to.
4. Specify the device that you want to use using the DEVICE= graphics option.
5. Generate your chart.
6. Close the ODS destinations that you opened in step 3.
7. Close ODS DOCUMENT.
8. Open ODS HTML.

Here is an example that shows how to create an ODS document containing three pie charts and how to store it in catalog Mygraphs.Mydocs. The vertical bar charts are generated with the PNG device.

```
/* Create the Mygraphs catalog */
LIBNAME Mygraphs "./";

/* Close the currrently open ODS destinations */
ods _all_ close;

/* Open the DOCUMENT destination. Specify catalog */
/* Mygraphs.Mydocs for the output and give it write permission */
ods document name=Mygraphs.Mydocs(write);

/* Open the HTML destination, and specify the PNG device. */
ods html style=seaside;
goptions reset=all device=png xpixels=540 ypixels=405;

/* Generate the charts */
proc gchart data=sashelp.cars gout=Mygraphs.Mydocs;
   vbar Type / subgroup=Origin;
      title1 "30 MPG or Better";
      where MPG_Highway >= 30;
   run;
   vbar Type / subgroup=Origin;
      title1 "Between 20 MPG and 29 MPG";
      where MPG_Highway < 30 and MPG_Highway >=20;
   run;
   vbar Type / subgroup=Origin;
      title1 "19 MPG or less";
      where MPG_Highway < 20;
   run;
quit;
```

```
/* Close the HTML and DOCUMENT destinations */
ods html close;
ods document close;

/* Reopen the HTML destination (not required in SAS Studio) */
ods html;
```

Replaying Your ODS Document

After you create your ODS document, use the DOCUMENT procedure to replay it. You can replay all of the graphs in your document or only those that you select. To see a list of the graphs in an ODS document, use a LIST statement with the DOCUMENT procedure. Here is an example that shows how to list the graphs in Mygraphs.Mydocs.

```
proc document name=Mygraphs.Mydocs;
    list / levels=all;
run;
quit;
```

A list of the graphs in the document is displayed in the Output window as shown in the following figure.

Listing of: \Mygraphs.Mydocs\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\GChart#1	Dir
2	\GChart#1\GCHART#1	Graph
3	\GChart#1\GCHART1#1	Graph
4	\GChart#1\GCHART2#1	Graph

In this example, the graphs are listed in the order in which they were inserted into the catalog. To replay individual graphs, you must know the path to the graphs, which is shown in the Path column.

Here is an example that shows how to play the first and third graphs in the Mygraphs.Mydocs catalog to the ODS PDF destination using the PNG300 device.

```
ods _all_ close;                        /* 1 */
goptions reset=all device=png300;                       /* 2 */
ods pdf file="myreport.pdf";                            /* 3 */
proc document name=Mygraphs.Mydocs;                     /* 4 */
   replay \Gchart#1\Gchart#1  / levels=all dest=pdf;
   replay \Gchart#1\Gchart2#1 / levels=all dest=pdf;
   run;
quit;
ods pdf close;                                          /* 5 */
ods html;                                               /* 6 */
```

1 Close the currently open ODS destinations.

2. Use the DEVICE= graphics option to specify the graphics output device that you want to use to generate the graphs.
3. Open the ODS destinations to which you want to send the output.
4. Run the DOCUMENT procedure with one or more REPLAY statements to replay your graphs. Specify the path to each graph, and use the DEST= option to specify the output destination.
5. Close the ODS destinations that you opened in step 2.
6. Open the ODS HTML destination (not required in SAS Studio).

To replay all of the graphs in the catalog, in the DOCUMENT procedure step, use one REPLAY statement that does not specify a path. For example:

```
ods _all_ close;
goptions reset=all device=png300;
ods pdf file="myreport.pdf";
proc document name=Mygraphs.Mydocs;
   replay / levels=all dest=pdf;
   run;
quit;
ods pdf close;
ods html; /* Not required in SAS Studio */
```

For more information about using the ODS DOCUMENT destination, see *SAS Output Delivery System: User's Guide*. For details about the DOCUMENT procedure, see *SAS Output Delivery System: Procedures Guide*.

Previewing Output

If you want to preview how a graph is going to appear on another device before you send it to that device, use the TARGETDEVICE= graphics option. For example, to preview output on your display as it would appear on a color PostScript printer, include TARGETDEVICE= in a GOPTIONS statement and specify the device for the printer:

```
goptions targetdevice=pscolor;
```

How output is displayed on your screen depends on the following:

- the orientation of the target device. As a result, the graph might not cover the entire display area of the preview device.
- the values of either the LCOLS and LROWS pair or the PROWS and PCOLS pair, depending on the orientation of the target device.
- the default color list of the target device.
- the values of the HSIZE and VSIZE device parameters for the target device. The HSIZE and VSIZE values are scaled to fit the display device, but they retain the target device aspect ratio.
- the value of the CBACK device parameter for the target device.

All other device parameter values, including the destination of the output, come from the current device entry. Therefore, the output displayed by TARGETDEVICE= might not be an exact replication of the actual output, but it is as close as possible.

See “TARGETDEVICE Graphics Option” on page 700 for a complete description of TARGETDEVICE=.

Printing Your Graph

About Printing Graphs

You can print your SAS/GRAPH output on hard-copy devices such as a printer. Regardless of the destination, you can create a hard copy of your graph in one of the following ways:

- Print the SAS/GRAPH program output directly to a hard-copy device.
- Print the SAS/GRAPH program output by creating a graphics output file, HTML file, or PDF file, and then printing the file using host commands or host application commands.
- Print the displayed graph directly from the GRAPH or Results Viewer window or the Graphics Editor window.
- Print the displayed graph directly from a browser that supports the SVG format.

Sending Your Graph Directly to a Printer

You can send graphics output directly to a hard-copy device by sending the graphics commands directly to the device. On most systems, you can use any of the following methods to print directly to a device:

- Use the ODS PRINTER destination to send your output directly to the default printer. Use the PRINTER= option if you want to direct your output to a printer other than the default printer or if a default printer is not defined.

  See the *SAS Output Delivery System: User’s Guide* for information about the ODS PRINTER statement.

  See *SAS 9.4 Universal Printing* for information about how to define a default printer for the Universal Printer.
- Use a FILENAME statement, a GOPTIONS statement, and a SAS/GRAPH device. The FILENAME statement defines a file reference that points to the print commands to send your output to any available hard-copy device. The GOPTIONS statement references the file reference, assigns the device, and specifies any additional parameters.
- Use the GDEVICE procedure to modify a SAS/GRAPH device entry to spool output directly to a printer. See Chapter 38, “GDEVICE Procedure,” on page 1163 for information about adding host commands to a device entry.
- Use the Universal Printing interface.

For detailed instructions on each of these methods, refer to the SAS Help facility for SAS/GRAPH.

Saving and Printing Your Graph

You can save your graph to a graphics output file, and then print the file using host commands. You can perform these two steps separately or combine them by incorporating the host printing commands into your program or graphics output device. In any case, you must choose a graphics file format that is compatible with your printer. For example, if you are using a PostScript printer, be sure to create a PostScript file using the appropriate device for the printer.

You can use any of the following methods to create and print a graphics output file:

- Use FILENAME and GOPTIONS statements to create the graphics output file, and then use a host command to spool the file to a spooler for the device.
- Use an ODS PRINTER statement to produce a PostScript, PDF, PCL, SVG, PNG, or GIF file. Then use a host command or a host application command to send the file to the printer.
- Use the GDEVICE procedure to modify a SAS/GRAPH device to save the output to a graphics output file and spool the output directly to a printer. See Chapter 38, “GDEVICE Procedure,” on page 1163 for information about modifying device entries.
- Use the Universal Printing interface.

  Note: On Windows platforms, the ODS PRINTER destination uses the Universal Printing interface in addition to the Windows system printers.

For detailed instructions on each of these methods, refer to the SAS Help facility for SAS/GRAPH.

Exporting Your Output

You can export your SAS/GRAPH output to other formats or to other software applications such as Microsoft Office. See the following topics for more information.

- “Replaying Output Using the DOCUMENT Procedure” on page 147
- Chapter 19, “Writing Your Graphs to a PDF File,” on page 289
- Chapter 18, “Using SAS/GRAPH Output with Microsoft Office Products,” on page 281

12

Plotting a Cloud Analytic Services (CAS) In-Memory Table

Accessing In-Memory Tables

Starting with SAS 9.4M5, the SAS/GRAPH procedures can access an in-memory table through a CAS engine libref. In that case, the data is downloaded from SAS Cloud Analytic Services to the SAS client through the CAS engine libref, and then it is processed on the SAS client by the SAS/GRAPH procedure. The amount of data that is read from an in-memory table is limited by system option CASDATALIMIT=. See "CASDATALIMIT= System Option" in *SAS Cloud Analytic Services: User's Guide*. It can also be limited by the DATALIMIT= CAS engine LIBNAME statement option or the DATALIMIT= data set option. See the DATALIMIT= LIBNAME statement option and the DATALIMIT= data set option.

If the data limit is exceeded, the procedure step is terminated, and an error message is written to the SAS log. Although you can use one of these options to increase the limit, processing a large amount of data locally might degrade SAS performance. For a very large in-memory table, use a procedure that is supported by the CAS server or use a CAS action to summarize or otherwise reduce the data, and then plot the results using SAS/GRAPH. See "Preprocessing Data in CAS and Plotting the Results Using SAS/GRAPH" on page 154.

For information about how to use a CAS engine libref to access in-memory tables, see "CAS LIBNAME Statement" in *SAS Cloud Analytic Services: User's Guide*.

Preprocessing Data in CAS and Plotting the Results Using SAS/GRAPH

Here is an example that shows how to summarize an in-memory table in CAS and how to plot the results using the GCHART procedure. For large in-memory tables, it is recommended that you preprocess the data in CAS to reduce the size of the data before you plot it using a SAS/GRAPH procedure. This example uses the data in SAS data set Sashelp.Cars as sample data. The MDSUMMARY procedure is used to summarize the data before it is plotted. The MDSUMMARY procedure is supported by the CAS server and can process a large amount of data. For information about the MDSUMMARY procedure, see "MDSUMMARY Procedure" in *SAS Cloud Analytic Services: User's Guide*.

Here is the SAS code for this example.

```
*options cashost="cloud.example.com" casport=5570;   /* 1 */
cas casauto sessopts=(caslib=casuser);               /* 2 */

libname mycas cas;                                   /* 3 */

proc casutil;                                        /* 4 */
  load data=sashelp.cars casout="cars";
quit;

proc mdsummary data=mycas.cars;                      /* 5 */
  var mpg_highway;
  groupby type;
  output out=mycas.cars_summary;
run;

proc print data=mycas.cars_summary;                  /* 6 */
  var type _mean_;
run;

goptions reset=all border xpixels=420 ypixels=340;   /* 7 */
axis1 label=(angle=90 "Average Highway MPG");
axis2 label=("Vehicle Type");
title "Average Highway MPG and Vehicle Type";
proc gchart data=mycas.cars_summary;
   vbar3d type / sumvar=_mean_ sum
      raxis=axis1 maxis=axis2;
   format _mean_ 5.2;
run;
quit;

cas casauto terminate;
```

1 System options CASHOST= and CASPORT= specify the host name and port for the CAS server. You must set these options if they are not already set. See

system options CASHOST= and CASPORT= in *SAS Cloud Analytic Services: User's Guide*.

2 The CAS statement establishes a connection with the CAS server. Session option CASLIB=CAUSER in the CAS statement specifies CASUSER as the active caslib. See CAS Statement and "CASLIB= Session Option" in *SAS Cloud Analytic Services: User's Guide*.

3 The LIBNAME statement creates CAS engine libref MYCAS, which enables the SAS client to access in-memory tables in the active caslib, CASUSER. See "CAS LIBNAME Statement" in *SAS Cloud Analytic Services: User's Guide*.

4 The CASUTIL procedure loads SAS data set Sashelp.Cars into in-memory table Cars. See "CASUTIL Procedure" in *SAS Cloud Analytic Services: User's Guide*.

5 The MDSUMMARY procedure summarizes variable Mpg_Highway, groups the results by variable Type, and then stores the results in in-memory table Cars_Summary in caslib CASUSER. See "MDSUMMARY Procedure" in *SAS Cloud Analytic Services: User's Guide*.

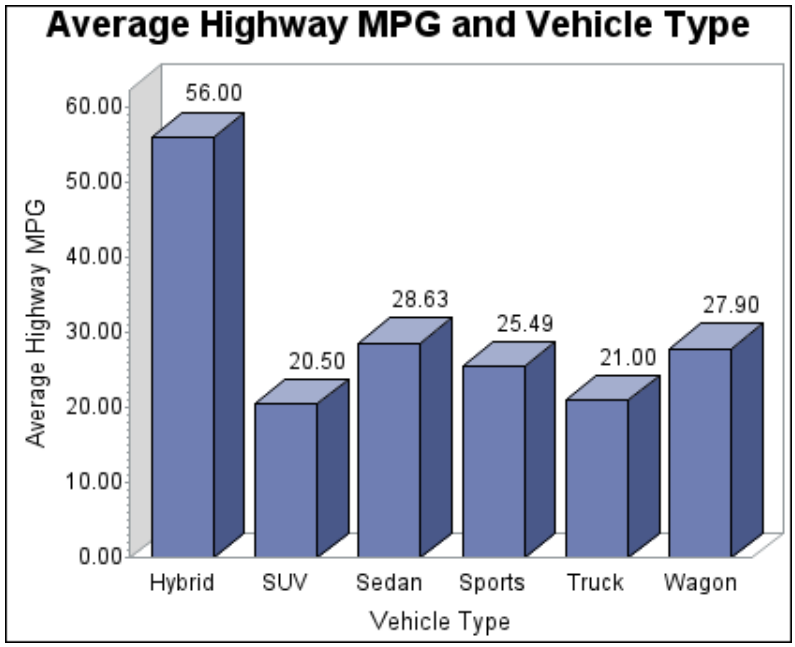
6 Optional: The PRINT procedure prints columns Type and _Mean_ in table Cars_Summary.

7 The GCHART procedure creates a bar chart from the data in in-memory table Cars_Summary. The VBAR3D statement creates a vertical bar chart of variable Type using variable _Mean_ as the sum variable and display the sum at the top of each bar.

The summarized data generated by the MDSUMMARY procedure contains only 6 observations as shown in the following image.

Obs	Type	_Mean_
1	SUV	20.5
2	Hybrid	56
3	Sedan	28.629770992
4	Truck	21
5	Sports	25.489795918
6	Wagon	27.9

Here is the resulting bar chart.


Average Highway MPG and Vehicle Type
Average Highway MPG
60.00
50.00
40.00
30.00
20.00
10.00
0.00
56.00
20.50
28.63
25.49
21.00
27.90
Hybrid
SUV
Sedan
Sports
Truck
Wagon
Vehicle Type

PART 3

Generating Output with SAS/GRAPH

13

Using SVG Graphics

Introduction to Scalable Vector Graphics

Scalable Vector Graphics (SVG) is an XML language for describing two-dimensional vector graphics.

SAS creates SVG files based on the World Wide Web Consortium (W3C) recommendation for SVG files. SAS SVG files are created using the UNICODE standard encoding.

Advantages of SVG Graphics

SVG graphics are vector graphics, so they can be resized without losing quality. A single SVG file can be scaled to any size or transformed to any resolution without compromising the clarity of the graphic. Bitmap images such as PNG and GIF lose quality anytime they are resized.

Also, if you need to display the same graphic at multiple sizes or resolutions, you would need multiple bitmap images, but only one SVG file. SVG files are displayed clearly at any size in any viewer or browser that supports SVG. Using browser controls, the user can zoom in to view details in a complicated SVG graphic.

An SVG file might also be smaller in size than the same graphic created by a bitmap (image) device such as GIF or PNG.

SVG files are ideal for producing documents of any size. The documents can be displayed on a computer monitor, PDA, or cell phone, or be printed or delivered as PDF files.

Universal Printers versus SAS/GRAPH Devices

You can create SVG files by using either the SAS/GRAPH SVG devices or the Base SAS SVG universal printers. The SVG devices enable you to generate SAS/GRAPH output using the Universal Printing system without specifying ODS statements or an OPTIONS PRINTERPATH= statement.

The SAS/GRAPH SVG devices are primarily for use with the LISTING and HTML destinations. With the PRINTER destination, it is required that you use the SVG universal printers directly.

The information provided here is limited to creating SVG files using SAS/GRAPH devices in the LISTING and HTML destinations. “Creating SVG Documents Using Universal Printing” in *SAS 9.4 Universal Printing*.

See also Table 9.12 on page 102 for examples showing the difference in using a shortcut device and a universal printer.

For detailed information about the SVG standard, see the W3C documentation at http://www.w3.org/TR/SVG.

The SVG Devices and the Output That They Create

There are five SVG devices:

SVG
: produces SVG 1.1 files. When used in the HTML destination, if your procedure produces multiple graphs, the SVG device produces separate SVG files for each graph. When used in the LISTING destination, the SVG device produces one SVG file, and the pages are in a continuous layout.

  When you also specify ANIMATION=START in the OPTIONS statement, the SVG device produces animated output.

SVGT
: produces SVG 1.1 files that are transparent (no background). These files are useful when you want to overlay several graphs on top of each other and you want all of the graphs to be visible. The SVGT device is intended for use when a procedure produces multiple graphs and is best used in conjunction with the ODS PRINTER destination.See "Creating Overlaid Transparent SVG Documents" in *SAS 9.4 Universal Printing* for more information.

SVGZ
: produces compressed SVG 1.1 files, which are useful when file size is an issue. However, some browsers do not support compressed SVG files, and you cannot view these files in a text editor. (See also "Browser Support for Viewing SVG Files" on page 169.)

SVGVIEW
: produces SVG 1.1 files with navigational controls when the SVG file contains multiple pages. This device is primarily for use in the LISTING destination with procedures that produce multiple graphs. The navigational controls enable you to page through the graphs. See "Example: Generating a Single SVG File with Multiple Pages and Page Controls" on page 166. When used in the HTML destination, the SVGVIEW device produces separate SVG files for each graph, just like the SVG device.

SVGANIM
: produces SVG 1.1 animated output. You can modify the animation timing by specifying animation system options.

  Note: You can also produce animated output by using the SVG device along with the ANIMATION=START system option. This documentation uses the SVG device in its examples.

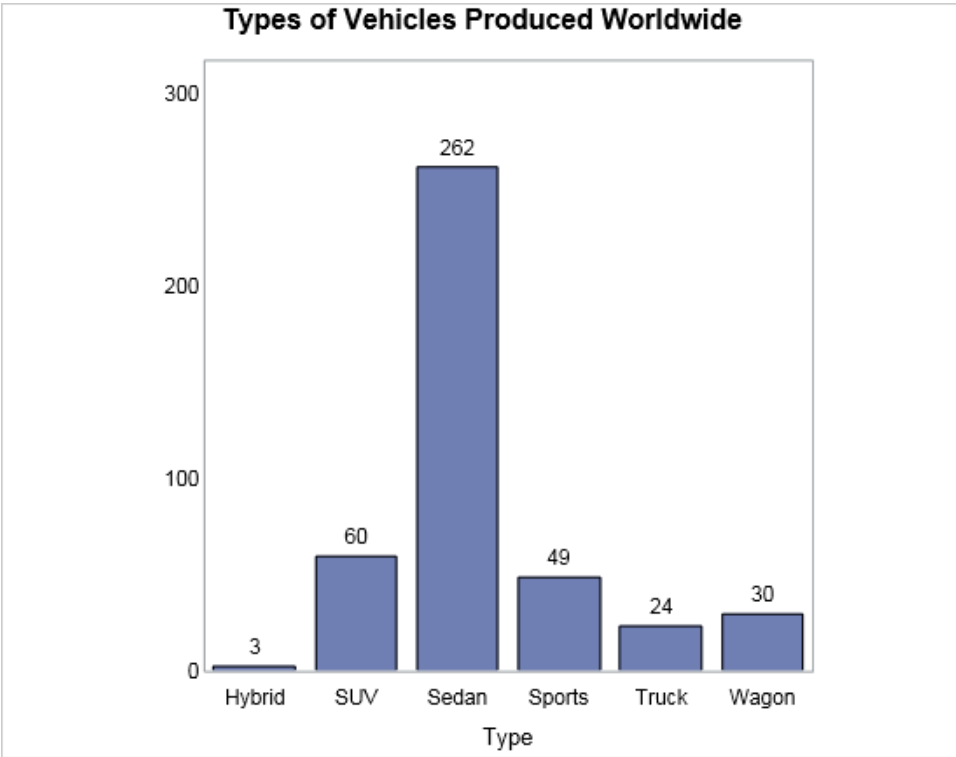
Example: Creating an SVG File

The following code produces an SVG file named `carstype.svg` and an HTML file named `cars.htm`:

```
/* Specify the output path */
filename svgout ".";

/* Specify the graphics options */
goptions reset=all device=svg
   hsize=5.75in vsize=4.5in
   gsfname=svgout;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Open the ODS HTML destination */
ods html path=svgout file="cars.htm";

/* Generate the bar chart.  The NAME= option */
/* specifies the name of the SVG file.       */
title h=2 "Types of Vehicles Produced Worldwide";
axis1 label=none major=none minor=none;
proc gchart data=sashelp.cars;
   vbar type / raxis=axis1 outside=freq
               name="carType";
run;
quit;

/* Close the ODS HTML destination */
ods html close;
ods html; /* Not required in SAS Studio */
```

You can view the SVG graphic by opening `cars.htm` in your SVG-enabled browser. If your browser does not render the graphic, see “Browser Support for Viewing SVG Files” on page 169.

You can view the SVG coding by opening the SVG file, `cartype.svg`, in a text editor.



Example: Generating Stand-Alone SVG Files in HTML5

In the HTML destination, SVG graphics are created as stand-alone files by default. In the HTML5 destination, SVG graphics are created as inline graphics in the HTML file. You can generate a stand-alone SVG image file in the ODS HTML5 destination by specifying `svg_mode='embed'` in the ODS HTML5 statement. This option embeds the SVG image file into the HTML file using the `<embed>` tag.

In the following example, the code shown in "Example: Creating an SVG File" on page 162 is modified to use this option.

> **TIP** When using the ODS HTML5 destination, it is important to close the HTML destination first. The following code closes all open destinations before opening the HTML5 destination.

```
/* Specify the output path */
filename svgout ".";

/* Specify the graphics options */
goptions reset=all device=svg
   hsize=5.75in vsize=4.5in
   gsfname=svgout;
```

```
/* Close the currently open ODS destinations */
ods _all_ close;

/* Open the ODS HTML5 destination */
ods html5 options(svg_mode="embed")
   path=svgout file="cars.htm";

/* Generate the bar chart.  The NAME= option */
/* specifies the name of the SVG file.       */
title h=2 "Types of Vehicles Produced Worldwide";
axis1 label=none major=none minor=none;
proc gchart data=sashelp.cars;
   vbar type / raxis=axis1 outside=freq
               name="carType";
run;
quit;

/* Close the ODS HTML5 destination */
ods html5 close;
ods html; /* Not required in SAS Studio */
```

In the SAS windowing environment, the map appears in the SAS Results viewer. The code produces an SVG image file named **cartype.svg** and an HTML file named **cars.htm**. You can view the SVG graphic by opening **cars.htm** in your web browser.

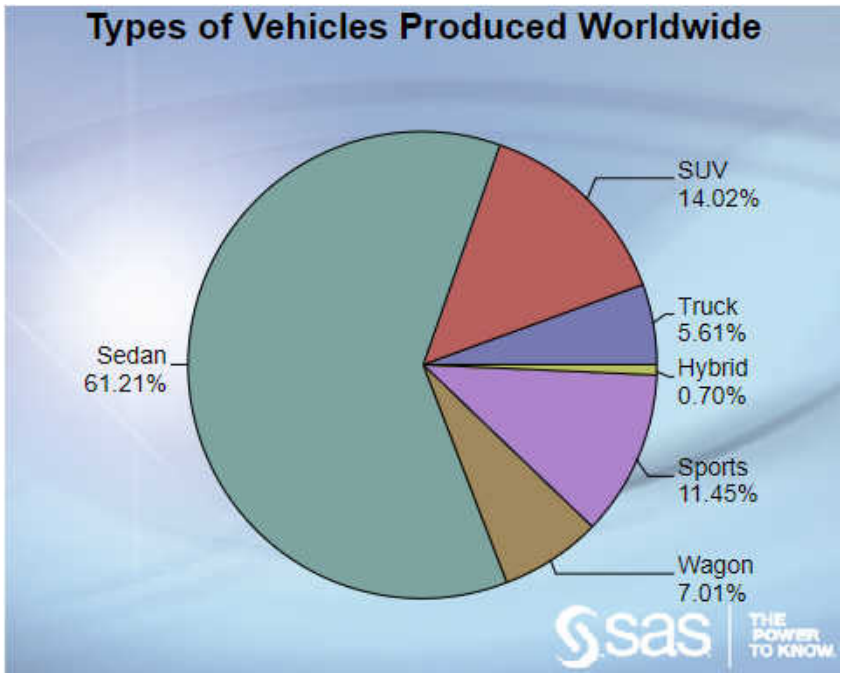
Example: Placing an Image behind a Graph

You can use the IBACK= or IFRAME= graphics options in the GOPTIONS statement to specify the graphics file that you want to be placed behind the SVG graphic. From the file that you specify with either of these options, SAS/GRAPH embeds a base64-encoded PNG image directly into the SVG file. For more information, see "Images in SVG Documents" in *SAS 9.4 Universal Printing*.

```
/* Specify the output file and the background image file */
filename odsout "file-path";
filename bgimage "file-path/images/svg_background_light.png";

/* Close the currently open ODS destinations */
ods _all_ close;

/* Specify the background image with the IBACK=  */
/* option. Replace external-image-file with the  */
/* name of an image that resides on your system. */
goptions reset=all device=svg hsize=5in vsize=4in
              gsfname=svgout htext=2
              imagestyle=fit iback=bgimage;

/* Specify the SVG viewport size */
```

```
options svgwidth=5in svgheight=4in;

/* Open the HTML5 destination */
ods html5 path=odsout file="carTypes.html";

/* Generate the pie chart. */
title height=3 "Types of Vehicles Produced Worldwide";
proc gchart data=sashelp.cars;
   pie type / other=0
      radius=35
      midpoints="Truck" "SUV" "Sedan" "Wagon" "Sports" "Hybrid"
      value=none
      percent=arrow
      slice=arrow
      noheading;
run;
quit;

/* Close the HTML destination */
ods html5 close;

/* Reset system options SVGWIDTH and SVGHEIGHT */
options svgwidth="" svgheight="";

/* Clear filerefs odsout and bgimage */
filename odsout clear;
filename bgimage clear;
```

The program creates a file named carTypes.svg. To view the results, open the SVG file in your browser.



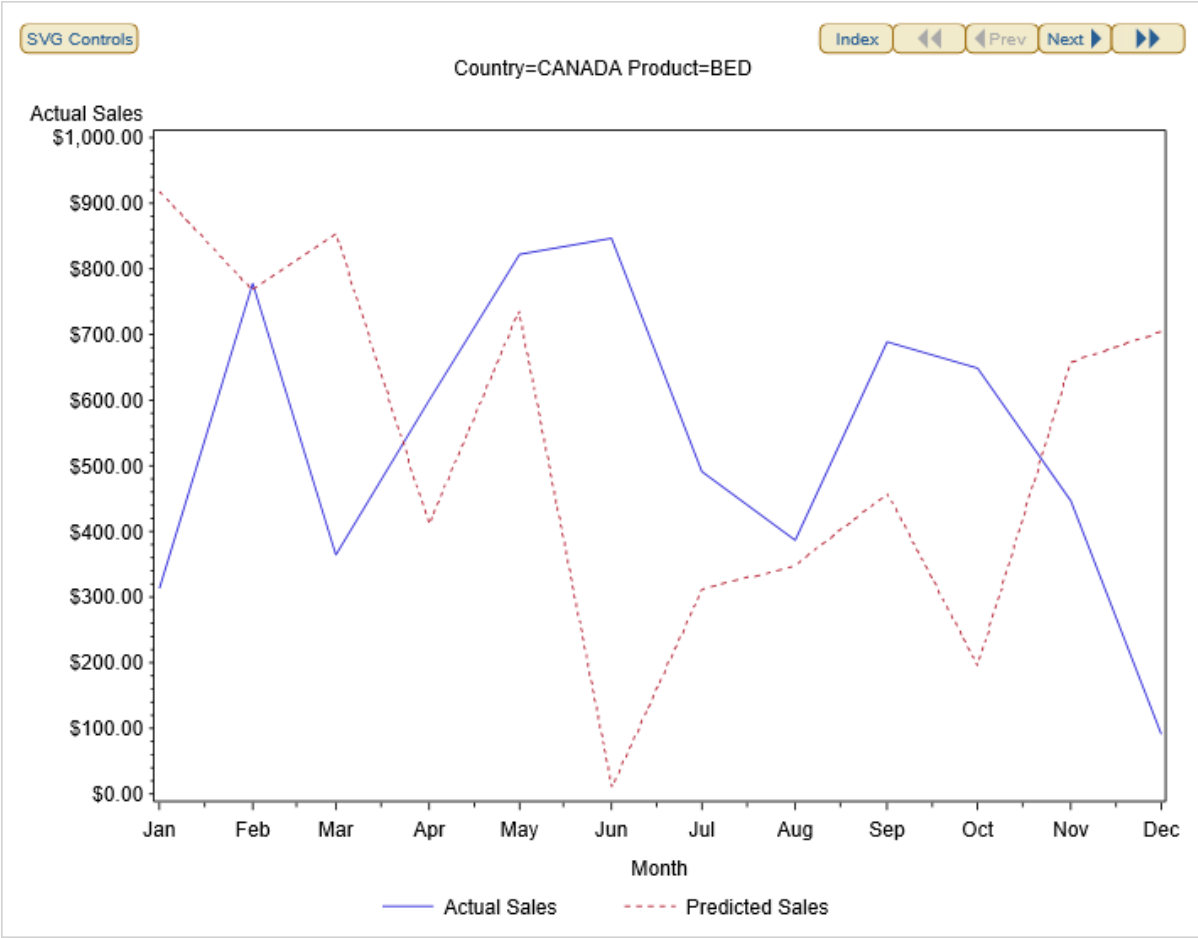
For additional information, see “Displaying an Image in a Graph Background” on page 385.

If your browser does not render the graphic, see “Browser Support for Viewing SVG Files” on page 169.

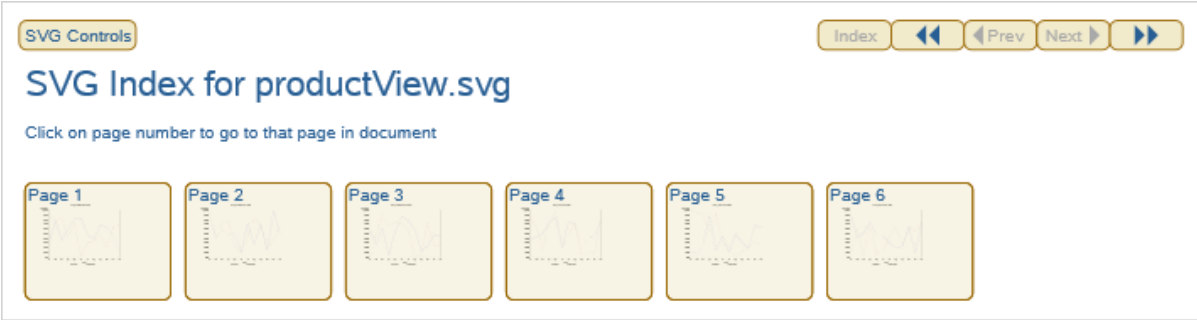
Example: Generating a Single SVG File with Multiple Pages and Page Controls

The SVGVIEW device is designed to be used when in the LISTING destination. It is useful when a single procedure produces multiple graphs, such as with BY-group processing. When used in the LISTING destination, the SVGVIEW device creates a single SVG file with multiple pages. Each graph produced by the procedure is on a different page. The SVG file, by default, has control buttons that enable you to navigate forward and backward through the graphs. You can also display an index page that shows a thumbnail image of each page.

For example, the following display shows the initial graph that is produced by the program in Example Code 13.1 on page 167. The program produces six graphs. You can page through them by clicking on the **Prev** and **Next** buttons.



The **Index** button displays a page of thumbnail images. There is one thumbnail for each page in the SVG file that, when clicked, take you to that page.



The program that generates this SVG file is as follows:

Example Code 13.1 *Program Code: Using the SVGVIEW Device with BY-Group Processing*

```
/* Specify the output path and filename */
filename mysvg "./productView.svg";

/* Subset the data set SASHELP.PRDSALE. */
/* Output the subset to WORK.PRODSUB.   */

data prodsub;
  set sashelp.prdsale;
  where year=1994 and
     (country = "U.S.A." or country = "CANADA")
      and region="EAST" and division="CONSUMER" and
     (product in ("SOFA", "TABLE", "BED"));
run;

/* Sort WORK.PRODSUB. */

proc sort data=prodsub;
   by country product;
run;

/* Close the currently open ODS destinations, and */
/* open the LISTING destination.   */
ods _all_ close;
ods listing;

/* Use the GSFNAME= option to send the  */
/* output of the LISTING destination to */
/* the MYSVG fileref.                   */

goptions reset=all device=svgview
         gsfmode=replace gsfname=mysvg;

/* Join the data points and change the */
/* line style for the predicted sales  */
/* to a dashed line.                   */

symbol1 interpol=join line=1 color=_style_;
symbol2 interpol=join line=2 color=_style_;
legend1 label=none;

/* Generate a graph for each unique    */
```

```
/* combination of country and product. */

proc gplot data=work.prodsub;
   by country product;
   plot actual*month predict*month /
        overlay legend=legend1;
run;
quit;

/* Close the LISTING destination,   */
/* and reopen the HTML destination. */
ods listing close;
ods html; /* Not required in SAS Studio */
```

The program creates a file named productView.svg. To view the results, open the SVG file in your browser.

When used in the HTML destination, the SVGVIEW device produces separate SVG files for each graph, just like the SVG device.

For additional information, see "Multi-Page SVG Documents in a Single File" in *SAS 9.4 Universal Printing* and "Creating Separate Files for Multi-Page SVG Documents" in *SAS 9.4 Universal Printing*.

If your browser does not render the graphic, see "Browser Support for Viewing SVG Files" on page 169.

Web Server Content Type for SVG Files

If the mime content type setting for your web server does not have the correct setting for SVG files, the results can be unpredictable. Your web browser might render SVG files as text files, or SVG files might be unreadable.

To ensure that SVG files are rendered correctly, you can configure your web server to use this mime content type:

```
image/svg+xml
```

Height, Width, and Scaling of SVG Files

In HTML5 output, SVG and SVGZ file height and width attributes are set to the size of the SVG. The reason is that many browsers do not scale an SVG file to 100% of the container. The Google Chrome and Safari browsers do scale the SVG file to 100% of the container.

To enable scaling of SVG files that can be scaled to the size of the container, you set the SVGHEIGHT= and SVGWIDTH= system options to 100%.

```
options svgheight="100%" svgwidth="100%";
```

Browser Support for Viewing SVG Files

Browsers That Support SVG Files

In order to view SVG files, you need a viewer or browser that supports Scalable Vector Graphics. Some browsers, such as Mozilla Firefox, have built-in support for SVG files.

Note: SVGZ graphics that you create in the ODS HTML5 destination can be viewed only with the Google Chrome or Opera web browsers.

The following table lists some browsers and viewers that support SVG files.

Table 13.1 *SVG Browser Support*

Browser or Viewer	Company
Batik SVG Toolkit	Apache Software Foundation
eSVG Viewer and IDE	eSVG Viewer for PC, PDA, Mobile
Google Chrome [1, 2]	Google Inc.
GPAC Project	GPAC
Microsoft Edge[1]	Microsoft
Mozilla Firefox [1]	Mozilla Foundation
Opera	Opera Software ASA
Safari, including iPad [1, 2, 3]	Apple, Inc.
TinyLine	TinyLine

1 This browser is supported by SAS.
2 See "Height, Width, and Scaling of SVG Files" on page 168.
3 When an animated SVG file is opened in Safari, the **Refresh** button does not work. You need to refresh the browser.

Using Mozilla Firefox to View SVG Graphics

- Compressed SVG files using the SVGZ Universal Printer are not supported.
- Zooming and panning features are not currently implemented.
- If you select **View** ⇨ **Page Style** ⇨ **No Style**, all graphs appear as a black rectangle.
- Firefox does not support font embedding. To avoid font mapping problems in your SVG file, you can set the NOFONTEMBEDDING system option. If the FONTEMBEDDING option is set when an SVG file is created and the SVG file is subsequently viewed in Firefox, Firefox uses the default font setting that is defined on the **Contents** tab in the Firefox Options dialog box.

Using an iPad to View SVG Graphics

To be able to view SVG graphics on a iPad and rotate the graphic to fit within the screen, you might need to do the following:

- Set the SVGHEIGHT= and SVGWIDTH= options. See "Height, Width, and Scaling of SVG Files" on page 168 for more information.
- Create an HTML file that embeds an SVG file and view the HTML file. See "Example: Generating Stand-Alone SVG Files in HTML5" on page 163 for more information.

Starting with SAS 9.4M2, the SVG, SVGT, and SVGView devices add the ROLE and ARIA-LABEL attributes in the HTML output when custom data tips are specified for hot-spots and the ODS HTML5 destination is used. These attributes make these custom data tips accessible to iPad users who are using the Voiceover screen reader. The SVG devices do not generate data tips by default. For information about how to add data tips to your SVG graphs, see "Data Tips in PNG, GIF, SVG, SVGT, SVGView, and SVGZ Presentations" on page 254.

See Also

"Browser Support for Viewing SVG Documents" in *SAS 9.4 Universal Printing* for information about browser support for viewing SVG documents.

Controlling Graph Resolution with the SVG Devices

The default resolution for the SVG devices is 96 DPI. Because the SVG devices are Universal Printer shortcut devices, you cannot change the resolution using options in the GOPTIONS statement.

If you are sending output to the HTML or LISTING destinations, you can control the resolution with the IMAGE_DPI= option on the ODS destination statement. For example:

```
ods listing image_dpi=600;
```

Controlling Graph Size with the SVG Devices

The default graph size for the SVG output is 800 x 600 pixels. You can change the size of your graph with the HSIZE= and VSIZE= graphics options. You can change the paper size by specifying the XMAX= and YMAX= or the XPIXELS= and YPIXELS= graphics options. Specifying a value for the XMAX=, YMAX=, XPIXELS=, or YPIXELS= graphics options changes the setting of the PAPERSIZE= system option. See "SAS System Options and SVG Output" on page 171 and "Graphics Options and Device Parameters Dictionary" on page 577.

SAS System Options and SVG Output

You can use the following system options to set the environment for creating SVG documents:

Table 13.2 *SAS System Options That Affect SVG Documents*

Task	System Option
Specify the name of an SVG printer to create a stand-alone SVG document.	PRINTERPATH=
Embed a comment in the SVG document	COLOPHON

Task	System Option
Options to set the SVG document size	
Set the paper size to use for Universal Printing.	PAPERSIZE=
Set the height of the SVG document. If the SVG document has embedded SVG documents, the height value affects only the outermost SVG document.	SVGHEIGHT=
Set the width of the SVG document. If the SVG document has embedded SVG documents, the width value affects only the outermost SVG document.	SVGWIDTH=
Set the x-axis coordinate for the lower left corner of an embedded SVG document.	SVGX=
Set the y-axis coordinate for lower left corner of an embedded SVG document.	SVGY=
Specify the X and Y coordinates, and the width and height that are used to set the viewBox for the outermost SVG document; specify the coordinates of the area of the document that is displayed in the viewport.	SVGVIEWBOX=
Specify whether to force uniform scaling of an SVG document.	SVGPRESERVEASPECTRATIO=
Options that modify the SVG document appearance	
Set the title that appears in the title bar of the SVG document.	SVGTITLE=
Specify whether to display navigational controls in a multi-page SVG document.	SVGCONTROLBUTTONS
Specify whether to display the magnify tool in SVG documents.	SVGMAGNIFYBUTTON
Options for animating SVG documents	
Start or stop creating an animation file.	ANIMATION
Set the amount of time that each frame of an animated document is held in view.	ANIMDURATION
Specifies whether to loop through the animation continuously or to play it one time.	ANIMLOOP

Task	System Option
Specifies whether to overlay frames in the animation or to play them sequentially.	ANIMOVERLAY
Specifies whether to immediately start an animation when an SVG document appears in the web browser.	SVGAUTOPLAY
Sets the number of seconds for a frame to fade into view.	SVGFADEIN
Specifies whether a frame in the animation overlaps the previous frame or if each frame is played sequentially when a frame is specified to fade in and out.	SVGFADEMODE
Sets the number of seconds for a frame to fade out of view.	SVGFADEOUT

For more information, see *SAS System Options: Reference*.

Overview of Animating Output with the SVG Device

You can use the SVG device to create a single SVG file that displays a sequence of images. Various SAS system options enable you to affect how the SVG devices generate output. Step-by-step details about creating an animated SVG image file are described in "Main Steps for Animating SVG Output" on page 230.

14

Generating SVG, PNG, GIF, and TIFF Graphics

What is a Non-Interactive Graph?

A non-interactive graph is a graph that is permanently fixed after it is displayed. You can view a non-interactive graph but you cannot manipulate it as you view it in a browser. Examples of non-interactive graphs include GIF and PNG images. To generate a non-interactive graph, in your SAS program, run a SAS graphics procedure and specify with the DEVICE= graphics option on one of the following devices:

SVG	TIFF
GIF	ACTXIMG
PNG	JAVAIMG

See Chapter 13, “Using SVG Graphics,” on page 159 for information about SVG devices and their output.

Variants of some of the devices are also available for special purposes. The GIF device by default creates images with dimensions of 800 x 600 pixels.

However, using the XPIXELS= and YPIXELS= graphics options with the GIF device is the recommended way to change the default size of your GIF graph to whatever size you need. See “Using the XPIXELS= and YPIXELS= Graphics Options to Set the Size of Your Graph” on page 129.

You can also use the following additional variants:

PNG300	produces PNG images with 300 DPI resolution
PNGT	provides support for transparency in PNG images
SVGZ	produces compressed SVG images
SVGT	provides support for transparency in SVG images
SVGVIEW	provides navigational control buttons for multipage SVG images

See “Using Graphics Devices” on page 89 for more information about these devices.

When you send your graph output to the ODS HTML destination, you can add data tips and drill-down links to your non-interactive graph. See Chapter 17, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” on page 249.

Creating a Non-Interactive Graph

You can use a GOPTIONS statement with a device type of SVG, PNG, GIF, and TIFF to create a non-interactive graphic output file from one or more SAS/GRAPH procedures. SAS first creates a GRSEG entry in a graphics catalog in your WORK

library, and then creates a graphics output file of the specified type from the GRSEG entry.

Follow these steps to generate one or more non-interactive graphs using the ODS LISTING destination:

1. Add a FILENAME statement to create a file reference for the location of the output files. To generate only one output file, specify the file reference, file name, and storage location as follows:

   ```
   filename mygif1 "C:\mysas\images\barchart.gif"; /* Path to output file */
   ```

   The file reference can be up to eight characters in length. To generate multiple images in a single program, specify a file reference for the path only, as follows:

   ```
   filename imageout "C:\mysas\images"; /* Path to output directory */
   ```

   When you generate multiple image output files, the SAS/GRAPH software automatically generates the names of the graphics output files, as described in "Summary of How Output Filenames and GRSEG Names Are Handled" on page 141.

2. Add a GOPTIONS statement to specify the output format using the DEVICE= graphics option, and the file reference using the GSFNAME= graphics option as follows:

   ```
   goptions reset=all device=gif  gsfname=mygif1;
   ```

   The value of the GSFNAME= graphics option is the name of your previously defined file reference, whether that file reference references a file name or a directory. If you do not specify a value for the GSFNAME= graphics option, the SAS/GRAPH software uses default names for your graphics output files as described in "Summary of How Output Filenames and GRSEG Names Are Handled" on page 141.

3. Close the currently open ODS destinations, and then open the ODS LISTING destination.

   ```
   ods _all_ close;
   ods listing;
   ```

4. Run the procedure that generates the graph. For example:

   ```
   proc gchart data=sashelp.class;
           hbar3d sex / sumvar=height type=mean;
    run;
   quit;
   ```

   The output is stored in the format specified by the DEVICE= graphics option, and in the output location specified by the GSFNAME= graphics option. For example, **`C:\mysas\images\barchart.gif`** is an output file for a bar chart.

5. Close the ODS LISTING destination, and then open ODS HTML (not required in SAS Studio).

   ```
   ods listing close;
   ods html; /* Not required in SAS Studio */
   ```

To create an HTML file that links to the image, use the ODS HTML destination with the following options:

BODY=
: The file name of the output HTML file (FILE= is a synonym for BODY=).

PATH=
The location (path, directory, or file reference) of the HTML file and non-interactive graph file.

GPATH=
The location of the graphics output file that is created.

Note: You must specify a value for the GPATH= option only if you specify the FILE= option as a *complete path* and file name, and you do not specify the PATH= option. You can specify FILE= as just a file name (and extension). If you also specify PATH=, then both the HTML file and the graphics output file are written to the same location (as specified by PATH=).

STYLE=
The style to be applied. If you do not specify a style, the default style is applied.

For samples, see "Examples: Generating Static Graphics" on page 183.

For complete information about these options, see *SAS Output Delivery System: User's Guide*.

SVG, PNG, GIF, and TIFF Devices Compared to ACTXIMG and JAVAIMG Devices

SVG, PNG, GIF, and TIFF Devices

When you use the graphics option DEVICE=SVG, PNG, GIF, and TIFF with a SAS/GRAPH procedure, an ODS style is applied to your graph by default. In other words, the GSTYLE system option is on by default. You can apply any of the ODS styles to your graph when the GSTYLE system option is on. You cannot apply an ODS style if the NOGSTYLE system option is on.

For complete information about the GSTYLE system option, see *SAS System Options: Reference*.

When you send your output to the ODS HTML destination, you can add data tips to your graph that are displayed when the cursor is over a portion of the image. You can also add drill-down links to other images or to other URLs. See "Links in SVG, PNG, and GIF Presentations" on page 261.

ACTXIMG and JAVAIMG Devices

When you use the graphics option DEVICE=JAVAIMG with a SAS/GRAPH procedure, an ODS style is always applied to your graph by default. In SAS 9.4M7 and earlier releases, the same is true when you specify DEVICE=ACTXIMG. This is the same behavior as with the SVG, PNG, GIF, and TIFF devices. You can apply any of the ODS styles to your graph. Unlike the SVG, PNG, GIF, and TIFF devices, the ACTXIMG and JAVAIMG style is not affected by the GSTYLE and NOGSTYLE system options. (See "Output from Different Devices and the GSTYLE/NOGSTYLE System Options" on page 330.)

You can also add data tips to your graph (see "Data Tips in ActiveX and Java Presentations" on page 254) and drill-down links to other URLs.

Developing Web Presentations with the PNG, SVG, and GIF Devices

You can use the SVG, PNG, and GIF devices to create web presentations that consist of non-interactive graphs. You can also add data tips and drill-down links to your graphs.

About the SVG, PNG, and GIF Devices

The SVG, PNG, and GIF devices enable you to generate non-interactive graphs for your web presentation. You can use these devices when you are sending output to the ODS HTML destination in order to generate an HTML file to display one or more graphs. For details, see "Controlling Graphics Output for ODS HTML" on page 137.

Enhancements that are available to SVG, PNG, and GIF web presentations include adding drill-down links or tooltip functionality. Styles other than the default ODS style can be applied with the STYLE= ODS option. Drill-down links can be added to the following elements:

- the chart elements (such as the bars, plot markers, or GMAP areas) using the URL= or the HTML= option
- the legend using the HTML_LEGEND= option
- the titles and footnotes using the LINK= option in the title or footnote statement
- the annotated text or graphs or both that use the HTML= variable in the annotate data set

For details, see "Adding Drill-Down Links to Web Presentations Generated with a Static-Graphic Device" on page 182.

When to Use the SVG, PNG, and GIF Devices

The SVG, PNG, and GIF devices are most suitable for web presentations that consist of non-interactive graphs and graphs with simple drill-down capabilities. In SAS 9.4M7 and earlier releases, if you need more interactivity, or if you want to compute responses to drill-down actions when the graph is viewed, then generate a presentation using the ACTIVEX device. The GIF device provides only 256 colors, which might be suitable for many presentations. The SVG and PNG devices provide TruColor support and are better suited for web presentations that contain color-intensive graphics.

For sample code that generates an HTML output file using the PNG device, see "Specifying the Name and Location of Your ODS Output" on page 133.

Developing Web Presentations with the JAVAIMG and ACTXIMG Devices

About the JAVAIMG and the ACTXIMG Devices

You can use the JAVAIMG device to create web presentations that include static graphs that are generated with Java technology. In SAS 9.4M7 and earlier releases, you can use the ACTXIMG device to create web presentations that include static graphs that are generated with the ACTIVEX device. However, graphs created with the ACTXIMG device do not have all of the interactivity that ACTIVEX graphs provide. You can add data tips and drill-down links to your JAVAIMG and ACTXIMG graphs.

The JAVAIMG and ACTXIMG devices enable you to generate web presentations that display a snapshot of one or more graphs in the PNG format. In SAS 9.4M7 and earlier releases, the ACTXIMG device works on PC hosts only. On all other hosts, the ACTXIMG device defaults to using the JAVAIMG device. Starting with SAS 9.4M8, the ACTXIMG device is no longer supported and always defaults to using the JAVAIMG device.

When you run a program that specifies the ACTXIMG device, the SAS/GRAPH ActiveX Control runs in the background to generate the PNG files. This means that you must install the SAS/GRAPH ActiveX Control on your computer before you can use the ACTXIMG device. For information about installing the ActiveX control, see "Installing the SAS/GRAPH ActiveX Control" on page 197. SAS/GRAPH procedures that can be used with the ACTXIMG device are the same as those that can be used with the SAS/GRAPH ActiveX Control, as listed in Table 15.1 on page 193. The procedures that can be used with the JAVAIMG device are listed in Table 15.1 on page 193.

The resulting PNG files can be viewed in any supported browser that supports the PNG format—neither Java nor ActiveX is required to view them.

The PNG files created by the ACTXIMG device are identical in appearance to the graphs created with the DEVICE=ACTIVEX graphics option as they are initially displayed in a browser. The latter graphs are interactive and can be manipulated by a user viewing them in a browser. However, the PNG files created by the ACTXIMG device are non-interactive and their appearance cannot be changed after they are created.

Note: With the JAVAIMG and ACTXIMG devices, the titles and footnotes are always stored in the HTML file. They are not stored in the graphics output files regardless of whether the GTITLE and GFOOTNOTE options are set. See "Controlling Titles and Footnotes" on page 115.

When to Use the ACTXIMG Device

You might not need interactivity such as changing the chart type or style. In this case, in SAS 9.4M7 and earlier releases, the ACTXIMG device provides several advantages over the interactive presentations that are generated with the ACTIVEX device. Because PNG image files are generated, the web clients are not required to install the SAS/GRAPH ActiveX Control to display the graphs. Also, web performance improves because the PNG image files are smaller in size than the HTML files that are required to run an ActiveX control.

Some of the SAS/GRAPH procedures, such as the GKPI and GAREABAR procedures, support only the JAVAIMG, ACTIVEX, and ACTXIMG devices. For these procedures, you must use the ACTXIMG device if you need a non-interactive image.

Finally, in some cases such as plots generated with the G3D procedure, the ACTXIMG device provides a better non-interactive image than the other devices.

Note:

- Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.
- In the z/OS operating environment, starting with SAS 9.4M8, the ACTXIMG and JAVAIMG devices are no longer supported. If you specify the ACTXIMG device, an error results.
- When SAS is installed on a server, the ACTXIMG device is limited by the display capabilities of the server on which it runs. The number of colors that the server is capable of displaying is just such a limitation. Consequently, the ACTXIMG PNG snapshot might not look as good as what you get from the ACTIVEX device. Therefore, it is better to use the ACTIVEX device if the server's display settings are less than optimal.

Using JAVAIMG in the z/OS Environment

Starting with SAS 9.4M8, the JAVAIMG device is no longer supported in the z/OS operating environment. If you specify the JAVAIMG device on z/OS, an error results. In SAS 9.4M7 and prior releases, if you specify the DEVICE=JAVAIMG graphics option, then you must specify FILESYSTEM=HFS. This is because HFS file space is needed to write the graphics output files. You might also need to increase the amount of memory that is allotted for your session so that SAS can run Java in the background. The suggested region size is 400 megabytes. For a batch job, add either REGION=400M or REGION=409600K to the JOB card. For a TSO session, specify SIZE(409600). For more information, refer to your JCL reference manual.

Generating an HTML Output File Using the JAVAIMG or the ACTXIMG Device

The procedure for generating an HTML output file for viewing JAVAIMG or ACTXIMG device output is similar to the procedure for generating the same file for the SVG, PNG, or GIF devices. For an example, see “Example: Using the ACTXIMG Device” on page 184.

Adding Drill-Down Links to Web Presentations Generated with a Static-Graphic Device

You can add drill-down links to web presentations that are generated with an SVG, PNG, GIF, JAVAIMG, or an ACTXIMG device. For information about the default configurations of these web presentations, see “Example: GIF Output with Drill-Down Links” on page 187.

You can add drill-down links to the following elements:

- graph elements or legend elements or both. See “Example: GIF Output with Drill-Down Links” on page 187.
- graph elements specified in an Annotate data set. See “Generating Drill–Down Links with the Annotate Facility” on page 266.
- titles and footnotes using the LINK= option in the TITLE or FOOTNOTE statement.

Implementing Data Tips and Drill-Down Links with the SVG Devices

You can implement data tips and drill-down links in SVG files that are generated in the HTML and LISTING destinations. In both cases, you use the HTML= option or the HTML_LEGEND= option (or both options) to specify variables in your input data that define the drill-down URLs. See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255 for information about implementing drill-down links, including defining link variables. For information about adding data tips, see "Data Tips for Web Presentations" on page 253.

Implementing data tips and drill-down links in SVG files that are generated in the LISTING destination has an additional requirement: you must specify the IMAGEMAP= option in the PROC statement. This option makes the image map generated by the procedure available to the SVG device. For example:

```
proc gchart data=sashelp.prdsale imagemap=myimgmap;
```

Note: You can also implement drill-down links and data tips in SVG output when you are using the ODS PRINTER destination. However, when you are using the ODS PRINTER destination, you are using a universal printer, not a device.

Examples: Generating Static Graphics

About These Examples

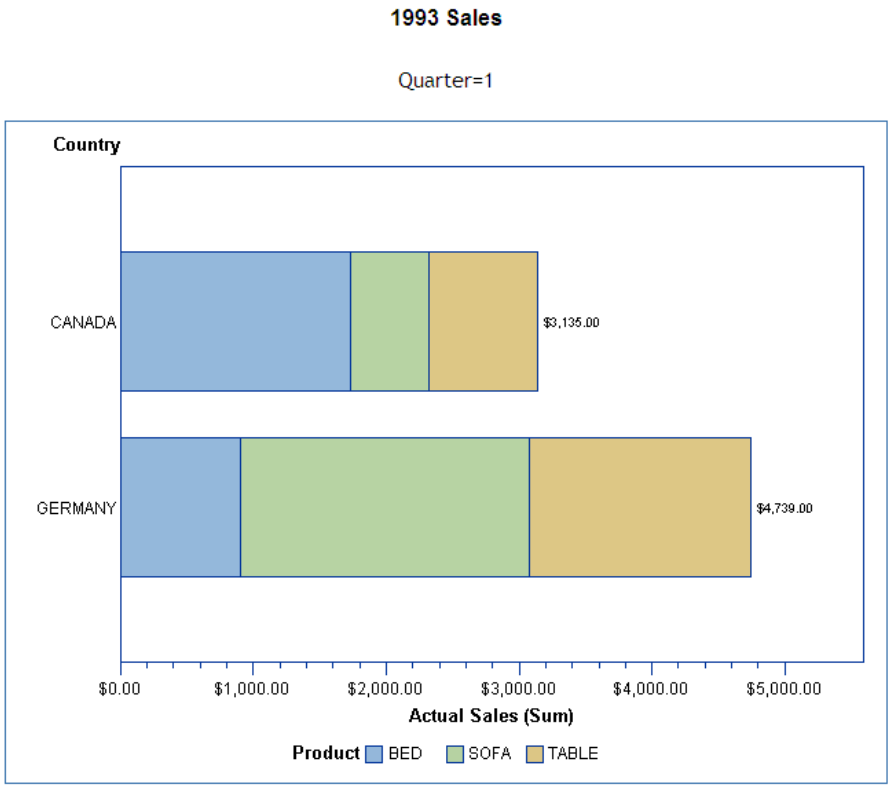
The examples in this section demonstrate how to create non-interactive images for a web presentation. The examples include:

Example: Using the ACTXIMG Device

Here is an example that uses the ODS HTML destination to create an HTML file in SAS 9.4M7 and earlier releases. (This example does not apply to SAS 9.4M8 and later releases.) The HTML file references four PNG files that are created by the GCHART procedure with the DEVICE=ACTXIMG graphics option. Because the ACTXIMG device invokes a SAS/GRAPH ActiveX Control, you can run this example only in a Windows environment.

The GCHART procedure in this example uses BY-group processing to display the results of each of the four quarters of the year. Consequently, the procedure produces four separate PNG files. Only the first graph is shown here. To see all of the PNG images in the output, you must scroll down the page in your browser.

Figure 14.1 *Using ODS with the ACTXIMG Device*



Here is the complete SAS code for this example. In this example, the output files are sent to the default working directory. If you want to send the output files to a different location, change the path in the ODS HTML statement PATH= option. If you want to send the PNG files to a separate location, add the GPATH= option to the ODS HTML statement to specify the new location for the PNG files.

See "ODS HTML Statement" in *SAS Output Delivery System: User's Guide* for details.

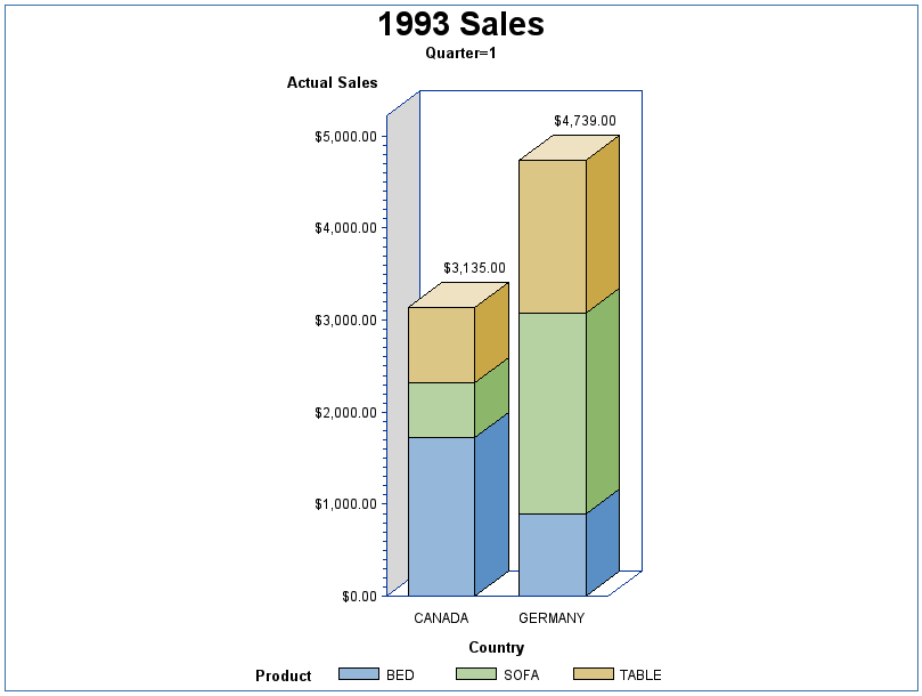
```
 /* Specify the ODS output path */
filename odsout ".";
```

```
/* Create data set from sashelp.prdsale */
data prdsummary;
  set sashelp.prdsale;
  where year=1993 and (country = "GERMANY" or country = "CANADA")
        and region="EAST" and division="CONSUMER" and
        (product="SOFA" or product="TABLE" or product="BED");
run;

/* Sort the data set by quarter */
proc sort data=work.prdsummary;
    by quarter;
run;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Specify the SEASIDE style for the HTML output */
ods html path=odsout file="sales.htm" style=seaside;

/* Specify device as actximg */
goptions reset=all device=actximg border;

title1 "1993 Sales";

/* Chart total 1993 sales for each country by quarter */
proc gchart data=work.prdsummary;
   hbar country / sumvar=actual subgroup=product sum;
   by quarter;
run;
quit;

/* Close ODS HTML and then reopen it to restore the default style. */
ods html close;
ods html; /* Not required in SAS Studio */
```

Example: Generating PNG Output

Here is an example that uses ODS to create an HTML file that references four PNG files that are created by a SAS/GRAPH procedure. The GCHART procedure in this example uses BY-group processing to display the results of each of the four quarters of the year. Consequently, the procedure produces four separate PNG files. Only the first graph is shown here. To see all of the graphs, you must scroll down the page in your browser.

Figure 14.2 Generating PNG Output Using ODS



Here is the complete SAS code for this example. In this example, the output files are sent to the default working directory. If you want to send the output files to a different location, change the path in the ODS HTML statement PATH= option. If you want to send the PNG files to a separate location, add the GPATH= option to the ODS HTML statement to specify the new location for the PNG files.See "ODS HTML Statement" in *SAS Output Delivery System: User's Guide* for details.

```
/* Specify the ODS output path */
filename odsout ".";

/* Create data set from sashelp.prdsale */
data prdsummary;
    set sashelp.prdsale;
    where year=1993 and (country = "GERMANY" or country = "CANADA")
        and region="EAST" and division="CONSUMER" and
        (product="SOFA" or product="TABLE" or product="BED");
run;

/* Sort the data set by quarter */
proc sort data=work.prdsummary;
    by quarter;
run;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Generate the graph */
ods html path=odsout file="sales.htm" style=seaside;
goptions reset=all border;
title1 "1993 Sales";
proc gchart data=prdsummary(where=(year=1993));
    vbar3d country / sumvar=actual subgroup=product sum;
```

```
    by quarter;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */
```

Notice that a device is not specified in the GOPTIONS statement in this example. ODS uses the PNG device as the default device for the HTML destination.

Example: GIF Output with Drill-Down Links

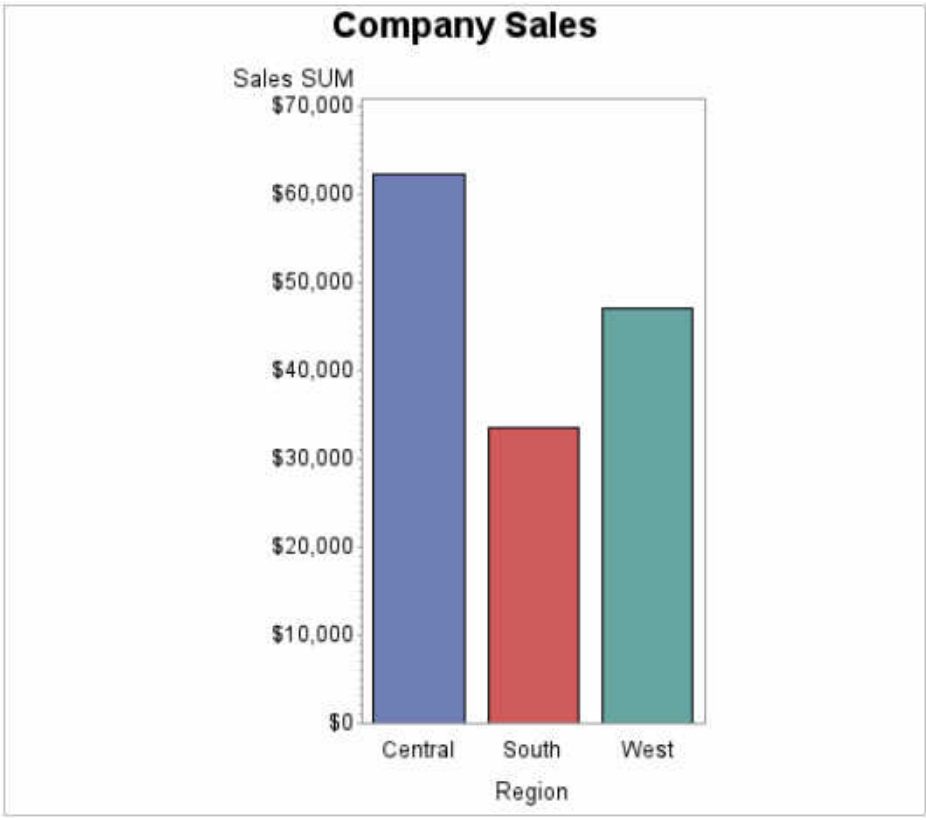
Here is an example that generates web output with drill-down functionality using the GIF device.

(See also Chapter 17, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” on page 249.)

In this example, the DEVICE=GIF graphics option generates image output files and the ODS HTML statement generates an HTML output file. The URL= option identifies a link variable that provides drill-down URLs. The values of the link variables are added to the data set with IF/THEN statements. ODS inserts the drill-down URLs into an image map that it generates in the HTML output file. Because the link variable contains the URL string only, the URL= option must be used to identify the link variable. The HTML= option cannot be used in this case.

When you display the HTML output file in a web browser, the following chart is displayed.

Figure 14.3 *Vertical Bar Chart with Drill-Down Links*



If you click one of the three blocks in the chart, you see a table of the data for that block. For example, if you click the Central bar, the following table is displayed.

Central Sales

State	Sales
IL	$18,038
IN	$13,611
OH	$11,084
MI	$19,660

Here is the example code.

```
/* Specify the ods output path. */
filename odsout ".";

 /* Create the data set REGSALES. */
data regsales;
   length Region State $ 8;
   format Sales dollar8.;
   input Region State Sales;

 /* Initialize the link variable. */
   length rpt $40;

/* Assign values to the URL variable. */
if Region="Central" then
     rpt="central.html";
   else if Region="South" then
     rpt="south.html";
   else if Region="West" then
     rpt="west.html";

   datalines;
West CA 13636
West OR 18988
West WA 14523
Central IL 18038
Central IN 13611
Central OH 11084
Central MI 19660
South FL 14541
South GA 19022
;

/* Close the currently open ODS destinations. */
ods _all_ close;

 /* Set graphic options. */
goptions reset=all border device=gif xpixels=500 ypixels=440;

 /* Open the HTML destination for ODS output. Specify the */
 /* path and filename. */
```

```
ods html5 path=odsout body="company.html"
   options(bitmap_mode="inline");

 /* Create a chart that uses the link variable.  */
title1 "Company Sales";
proc gchart data=regsales;
   vbar region / sumvar=sales
   patternid=midpoint
   url=rpt; /* Set the URL variable to rpt */
run;
quit;
ods html5 close;

 /* Create the Central sales page */
ods html path=odsout body="central.html";

title1 "Central Sales";
proc print data=regsales noobs;
   var state sales;
   where region="Central";
run;
quit;

/* Create the Southern sales page */
title1 "Southern Sales";

ods html body="south.html";

proc print data=regsales noobs;
   var state sales;
   where region="South";
run;
quit;

/* Create the Western sales page */
title1 "Western Sales";
ods html body="west.html";

proc print data=regsales noobs;
   var state sales;
   where region="West";
run;
quit;

 /* Close ODS HTML to close the output file, and then reopen ODS HTML.
*/
ods html close;
ods html; /* Not required in SAS Studio */
```

The code for this example is also available in the SAS Sample Library under the name GWBDRILL. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

15

Generating Interactive ActiveX Graphics

Overview of Creating Interactive Output for ActiveX

Note:

- Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.
- Starting with SAS 9.4M8, SAS/GRAPH ActiveX Control is not available from SAS. If you order SAS/GRAPH 9.4M8 or a later version, SAS/GRAPH ActiveX Control is not included in that order. If you have an existing installation of SAS/GRAPH ActiveX Control in your environment and plan to upgrade or migrate to SAS 9.4M8 or later, SAS recommends that you first uninstall SAS/GRAPH ActiveX Control. See “Uninstalling the SAS/GRAPH ActiveX Control” on page 200.

  For more information about uninstalling retired products, see Unconfiguring and Uninstalling Retired Products in *SAS Guide to Software Updates and Product Changes*.

In SAS 9.4M7 and earlier releases, you can use the ACTIVEX device to generate interactive output for ActiveX. This device is used with the ODS HTML or the ODS RTF destination to create an HTML file or an RTF file that contains XML code that is consumed by the SAS/GRAPH ActiveX Control. The SAS/GRAPH ActiveX Control provides user interactivity in Microsoft Office products in the Windows operating environment. Interactive features include the ability to change graph types (a bar chart to a pie chart, for example), display data tips at the point of the cursor, rotate and zoom, reassign variable roles, and modify axes, legends, colors, and text fonts.

For your web users who have installed SAS locally, the control is run automatically when the HTML output file is displayed in Internet Explorer or in Edge when reloaded in Internet Explorer mode, if that feature is enabled. For your web users who have not installed SAS installed, and who have not already installed the SAS/GRAPH ActiveX Control, you can configure your HTML output file to prompt them to install the control at display time, as described in “Installing the SAS/GRAPH ActiveX Control” on page 197.

You can enhance your ActiveX presentations by adding drill-down links (see “Configuring Drill-Down Links with ActiveX” on page 207) and configuring interactive features (see “Specifying Parameters and Attributes for ActiveX” on page 1644).

In addition to HTML output, you can use the SAS/GRAPH ActiveX Control to display interactive graphs in Object Linked Embedded (OLE) documents, and in applications written in Visual Basic, C++, and JavaScript. You can also include them in Microsoft Office Products, such as Word, Excel, and PowerPoint. See Chapter 18, "Using SAS/GRAPH Output with Microsoft Office Products," on page 281.

The following table lists the procedures and statements that generate output that can be displayed in the SAS/GRAPH ActiveX Control.

Table 15.1 *Procedures and Statements That Generate Output for the SAS/GRAPH ActiveX Control*

Procedure	Statements
GAREABAR[1]	HBAR, VBAR
GBARLINE	BAR, PLOT
GCHART	BLOCK, HBAR, HBAR3D, VBAR, VBAR3D, PIE, PIE3D, DONUT
GCONTOUR	PLOT
GMAP	CHORO, BLOCK, PRISM
GPLOT	BUBBLE, BUBBLE2, PLOT, PLOT2
GRADAR	CHART
G3D	PLOT, SCATTER
GTILE	FLOW, TILE, TOGGLE

1 Starting with SAS 9.4M8, because the ACTIVEX and ACTXIMG graphics devices are obsolete, the GAREABAR procedure is also obsolete. If you attempt to run the GAREABAR procedure, an error message is written to the SAS log.

Note the following:

- Displaying an ActiveX graph in Microsoft Internet Explorer might result in a BEX error. For more information, see Usage Note 57554: Displaying an ActiveX graph in Microsoft Internet Explorer might result in a BEX error after the application of Microsoft patch KB3000850 on support.sas.com.
- If you want to use the SAS/GRAPH ActiveX control in RTF output, see Problem Note 65142: Creating an RTF file by using the ACTIVEX device driver results in an RTF file that Microsoft Word cannot open.
- The SAS/GRAPH ActiveX Control is not compatible with the RTF output format when UTF-8 character encoding is used.
- Using PROC GMAP to generate a highly detailed map might create a large HTML output file, which might cause problems on certain web browsers. If this is the case, you can use PROC GREDUCE to remove some of the complexity and produce a more usable map.

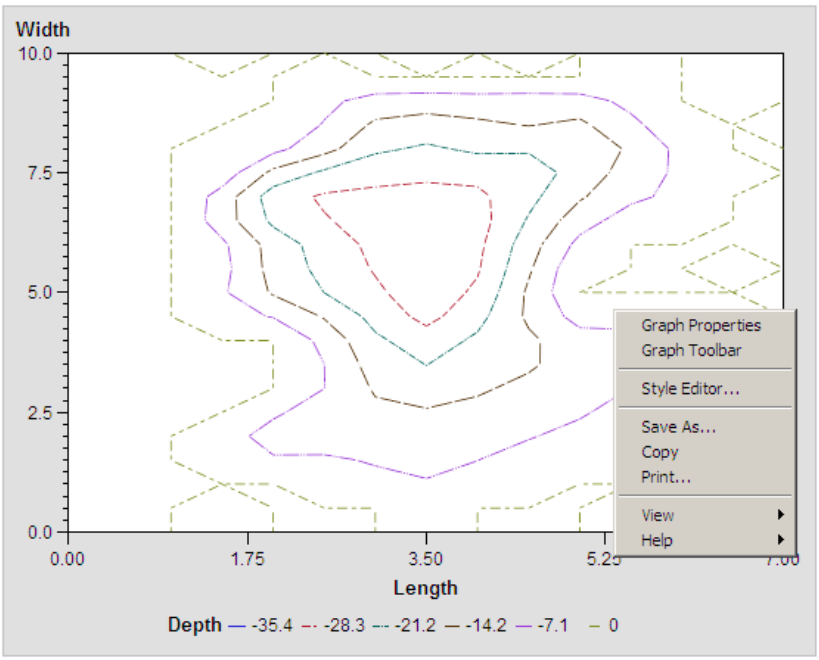
- The SAS/GRAPH ActiveX Control does not enable 8-bit gray scale images. If you use images for backgrounds or chart elements, make sure that they are 24-bit images.
- The ACTIVEX device does not support character values on the response axis. When generating plots with the ACTIVEX device, at least one variable must be numeric. When a numeric value is specified for the category axis and a character variable is specified for the response axis, the axis values are swapped in order to place the character value on the category axis. If both values are character values, the plot is not generated.

SAS/GRAPH ActiveX Control Display Features

The SAS/GRAPH ActiveX control displays the output of most of the SAS/GRAPH procedures and many of the annotate functions. It enables extensive interactive features via a pop-up menu. The pop-up menus enable you to move, rotate (3-D graphs only), and zoom, and to control the properties of graphs such as its colors, legends, and axes.

You can enable data tips and drill-down links with presentations created for the ActiveX control. Figure 15.1 on page 194 shows output from the GCONTOUR procedure as displayed by the ActiveX control. (You can open the pop-up menu for the ActiveX control by right-clicking the graph.)

Figure 15.1 *Sample ActiveX Presentation*



The ActiveX control can be viewed only in Windows operating environments with Microsoft Internet Explorer on a PC with the ActiveX control installed. The output can be inserted into Word, PowerPoint, and Excel documents as well.

To create a graph to be displayed by ActiveX, specify DEVICE=ACTIVEX on your GOPTIONS statement. See "Generating Output for ActiveX" on page 201 for more information.

For more information about SAS/GRAPH graphics output devices, see "Using Graphics Devices" on page 89.

About Using the SAS/GRAPH ACTIVEX Device in SAS Studio

Default Output in SAS Studio

By default, SAS Studio generates output as HTML, PDF, and RTF files. HTML is the primary output format in SAS Studio. When you run a SAS program in SAS Studio, any HTML output that is generated is displayed in the Results tab. PDF and RTF output, if requested, is also generated but is not displayed. You must download any PDF and RTF output from the **Results** tab. For information about results in SAS Studio, see *SAS Studio: User's Guide* for your version of SAS Studio. You can find the documentation for all versions of SAS Studio on the SAS Studio documentation page on support.sas.com.

Generating ActiveX HTML Output in SAS Studio

The HTML output in SAS Studio is generated using a customized ODS HTML5 destination that generates images in the PNG or SVG format. Using the GOPTIONS DEVICE= option to specify a device has no effect on the default HTML output in SAS Studio. In SAS 9.4M7 and earlier releases, to generate ActiveX output in HTML in SAS Studio, use your own ODS HTML destination as follows:

1. Close the currently open ODS destinations:

   ```
   ods _all_ close;
   ```

2. Open your ODS HTML destination, and specify the location of the output files:

   ```
   ods html path="path-to-output" file="html-filename.html" <options>
   ```

   where:

   path="*path-to-output*"
   specifies a writable path for the output file that is accessible to the SAS Studio server.

file="*html-filename.html*"
specifies a name for the HTML output file.

<options>
specifies one or more additional ODS options. For information about additional ODS options, see *SAS Output Delivery System: User's Guide*.

3 Use SAS/GRAPH procedures such as GCHART or GPLOT to generate your output.

4 When you have finished generating your output, close your ODS HTML destination:

```
ods html close;
```

The examples in this book follow this general pattern so that you can run the examples in SAS Studio or in the SAS windowing environment.

Generating ActiveX RTF Output in SAS Studio

Unless your SAS Studio is configured to use UTF-8 character encoding, you can generate ActiveX RTF output using the default output in SAS Studio. Otherwise, you can generate only static RTF output using the ACTXIMG device. If your SAS Studio is configured to use an encoding other than UTF-8, generate your RTF output using the ACTIVEX device as you normally would in SAS Studio, and then download the RTF output from the Results tab. Alternatively, use a pattern similar to that described in "Generating ActiveX HTML Output in SAS Studio" on page 195 to open your own ODS RTF destination.

Data Tip and Drill-Down Support in ActiveX

When generating graphs using the SAS/GRAPH ActiveX Control, you can enhance your graphs by doing the following:

- displaying pop-up text when the mouse pointer is over a portion of the diagram. See "Data Tips for Web Presentations" on page 253.
- adding drill-down links that enable you to link to other web resources. See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255.

Note: The SAS/GRAPH ActiveX Control does not support chart descriptions.

Installing the SAS/GRAPH ActiveX Control

About the ActiveX Control Installation

In SAS 9.4M7 and earlier releases, the SAS/GRAPH ActiveX Control is installed silently when you install SAS/GRAPH. When you create a web presentation using the SAS/GRAPH ACTIVEX device, by default, the resulting presentation is configured to prompt users to install the SAS/GRAPH ActiveX Control if it is not already installed. The SAS/GRAPH software configures the presentation by setting the CODEBASE= option in the HTML file as shown in the following example:

```
CODEBASE="http://www2.sas.com/codebase/graph/v94/sasgraph.exe#version=9,4"
```

No files are installed without the user's permission. Your users can refuse installation by refusing the licensing agreement at the beginning of the installation process. Also note that the installation program does not run if the control has already been installed. You can customize the installation process as described in “Customizing the SAS/GRAPH ActiveX Control Installation Process” on page 197.

Customizing the SAS/GRAPH ActiveX Control Installation Process

Rather than using the default installation process, you can customize the installation by downloading the installation program and running it manually on your local computers. You can also make the installation program available to all of your users on your local network for installation on an as-needed basis. To download the installation program, do the following:

1. Open the SAS All Downloads page in your web browser:

   http://support.sas.com/demosdownloads/

2. Click **SAS/GRAPH Software** in the SAS software list.
3. Click **SAS/GRAPH Control for ActiveX** under the appropriate SAS version.
4. In the SAS/GRAPH Control for ActiveX download table, click the executable file under Request Download for your Windows platform.
5. If the Sign In page is displayed, enter your credentials, and then click **Sign In**.

Note: You must be signed in on your account in order to download files. If you do not have an account, click **Create Profile** under New User? to create your account profile.

6 Click the executable file under Request Download for your Windows platform again.

7 Read the license agreement, and then click **Accept** to accept the agreement.

Note: To cancel the download, click **Do Not Accept**.

8 Download the file to your local computer.

To install the SAS/GRAPH ActiveX on a local computer, run the Control for ActiveX installation program as Administrator, and then and follow the installation prompts. The installation program installs the SAS/GRAPH ActiveX Control files in the folder shown in the following table.

Computer Platform	ActiveX Control Platform	Installation Folder
Win32	Win32	`c:\Program Files\SASHome\Graph\V9`
Win64	Win32	`c:\Program Files (x86)\SASHome\Graph\V9`
	Win64	`c:\Program Files\SASHome\Graph\V9`

Installation requires about 15 megabytes of disk space for the 32-bit version and about 30 megabytes for the 64-bit version.

If you want to configure your SAS/GRAPH ActiveX web presentations to install the SAS/GRAPH ActiveX Control as-needed from your local network rather than the SAS website, copy the installation program to a shared location on your network. The shared location can be a shared directory or public folder on your file system, or a web server on your network. After you have made the installation program available to your users, you can then configure your new and existing ActiveX presentations to install the SAS/GRAPH ActiveX Control from your local network. See "Configuring Your ActiveX Presentation to Install the SAS/GRAPH ActiveX Control from Your Local Network" on page 199 and "Configuring an Existing ActiveX Presentation to Install the SAS/GRAPH ActiveX Control from Your Local Network" on page 200.

Configuring Your ActiveX Presentation to Install the SAS/GRAPH ActiveX Control from Your Local Network

To configure a SAS/GRAPH ActiveX web presentation to install the SAS/GRAPH ActiveX control from your local network, do the following:

1. If you have not already done so, download the SAS/GRAPH ActiveX Control installation program and copy it to a shared location as described in "Customizing the SAS/GRAPH ActiveX Control Installation Process" on page 197.

2. In the SAS program for your presentation, use the CODEBASE= option with the ODS HTML statement to configure the HTML output file to reference the installation program when the HTML file is opened. For example:

   ```
   ods html body="myGraph.html"
   codebase="http://www.ourco.com/sasweb/graph/
   installation-filename.exe#version=9,4";
   ```

   where *installation-filename*.exe is the name of the Control for ActiveX installation program.

   If the installation program is stored on your file system rather than on a web server, then you can use a file specification as the value of the CODEBASE attribute. For example:

   ```
   ods html body="myGraph.html"
   codebase="/sasweb/graph/installation-filename.exe#version=9,4";
   ```

3. (Optional) If SAS or the SAS/GRAPH ActiveX Control is not installed on a local computer, open your web presentation and verify the following:

   - A prompt to install the SAS/GRAPH ActiveX control appears.
   - The installation is successful.
   - The presentation is displayed properly.

   Note: If you want to remove the SAS/GRAPH ActiveX Control after the test, follow the instructions in "Uninstalling the SAS/GRAPH ActiveX Control" on page 200.

Configuring an Existing ActiveX Presentation to Install the SAS/GRAPH ActiveX Control from Your Local Network

To configure an existing SAS/GRAPH ActiveX web presentation so that it installs the SAS/GRAPH ActiveX Control from your local network, do the following:

1. If you have not already done so, download the SAS/GRAPH ActiveX Control installation program and copy it to a shared location as described in "Customizing the SAS/GRAPH ActiveX Control Installation Process" on page 197.
2. In a text editor, open the initial HTML file of your web presentation.
3. In the OBJECT tag, insert the CODEBASE= attribute. The attribute references the location of the installation program. The following CODEBASE value references a public directory:

   ```
   CODEBASE="/sasweb/graph/installation-filename.exe#version=9,4"
   ```

   If the installation program is stored on a web server, use an HTTP reference. For example:

   ```
   CODEBASE="http://www.ourco.com/sasweb/graph/
   installation-filename.exe#version=9,4"
   ```
4. Save the HTML file and close the editor.
5. (Optional) If SAS or the SAS/GRAPH ActiveX Control is not installed on a local computer, open the modified web presentation and verify the following:
   - A prompt to install the SAS/GRAPH ActiveX control appears.
   - The installation is successful.
   - The presentation is displayed properly.

   Note: If you want to remove the SAS/GRAPH ActiveX Control after the test, see "Uninstalling the SAS/GRAPH ActiveX Control" on page 200.

With the file thus modified, displaying the HTML file gives users who need it the option of installing the control in the default location on their local computers.

Uninstalling the SAS/GRAPH ActiveX Control

On Windows, you can uninstall the SAS/GRAPH ActiveX Control from Control Panel. To manually uninstall the SAS/GRAPH ActiveX Control on Windows 10, for example, do the following:

1. On the taskbar click **Start**, type `control panel`, and then select **Control Panel App**.

Note: Alternatively, you can click the search icon, type `control panel`, and then select **Control Panel App**.

2 Under Programs, click **Uninstall a program**.

3 Click **SAS/Graph ActiveX Control version 9.4** in the list of currently installed programs.

4 Click **Uninstall**, and then follow the prompts to remove the program.

5 Close the Control Panel window.

For information about how to manually uninstall this program on other versions of Microsoft Windows, refer to the Microsoft documentation.

Generating Output for ActiveX

In SAS 9.4M7 and earlier releases, the SAS/GRAPH ActiveX Control displays interactive charts, maps, and plots in Microsoft Internet Explorer, Word, PowerPoint, and Excel. Table 15.1 on page 193 lists the SAS/GRAPH procedures that generate output for ActiveX.

Follow these steps to generate a default web presentation that runs the SAS/GRAPH ActiveX Control.

1 Reset the graphics options and specify the ACTIVEX device:

```
goptions reset=all device=activex;
```

2 Close the currently open ODS destinations:

```
ods _all_ close;
```

3 Open the ODS HTML or ODS RTF destination. Use the STYLE= option to specify an ODS style (see Chapter 21, “Using ODS Styles, Device Parameters, and Options,” on page 321), and use the PATH= and BODY= options to specify an output file name other than the default. Use the NOGTITLE option to write the graph title outside the graph borders.

Note: When the ACTIVEX device is used, the graph title is always written outside the graph border. The GTITLE option is ignored in that case.

For example:

```
ods html style=analysis path="output-path" file="your_file.htm"
   nogtitle;
```

4 Run a procedure or procedures that are supported by the ACTIVEX device (see Table 15.1 on page 193):

```
title "Chart of Average Weight and Height by Sex";
proc gchart data=sashelp.class;
   vbar sex / type=mean sumvar=height group=age subgroup=sex space=0;
run;
```

```
quit;
```

5 Close the ODS destination that you opened in step 3 to close the HTML file. If you are using the SAS windowing environment, reopen the ODS HTML destination. For example:

```
ods html close;
ods html; /* Not required in SAS Studio. */
```

The preceding program assumes that your web users have installed the SAS/GRAPH ActiveX Control in advance. If the SAS/GRAPH ActiveX Control is not already installed on a user's computer, your web presentation automatically prompts the user to install it. For information about prompting new users to start the SAS/GRAPH ActiveX Control installation process, see “Configuring Your ActiveX Presentation to Install the SAS/GRAPH ActiveX Control from Your Local Network” on page 199. For further troubleshooting information, see Chapter 20, “Troubleshooting Web Output,” on page 311. For information about enhancing the default web presentation, see “Configuring Drill-Down Links with ActiveX” on page 207.

About Languages in ActiveX

For international audiences, the SAS/GRAPH ActiveX Control has a graphical user interface that can appear in the following languages:

Arabic	French	Korean
Chinese, simplified	German	Polish
Chinese, traditional	Hebrew	Russian
Danish	Hungarian	Spanish
Dutch	Italian	Swedish
English	Japanese	Turkish

In general, web-based ActiveX devices must use a language-specific operating environment and web browser in order to display a translated graphical user interface. For more information, contact your on-site SAS support personnel.

About Special Fonts and Symbols in ActiveX

The ACTIVEX device supports only system fonts. You can also use characters from many of the fonts that you have installed on your computer. The ACTIVEX device supports most of the symbols that you can use with the VALUE= option in the SYMBOL statement. (See “SYMBOL Statement” on page 470.) The following table shows the symbols that are supported by ActiveX and the value that you must specify in the SYMBOL statement VALUE= option for each.

Table 15.2 Symbols That Are Supported by ActiveX

Value	Symbol	Value	Symbol
ARROW	↑	MARS or >	♂
ARROWDOWN	↓	MARSFILLED	♂
ARROWLEFT	←	PLUS	+
ARROWRIGHT	→	POINT	.
ASTER	✳	SALTIRE	✖
ASTERISK	✳	SALTIREFILLED	✖
CHAIN	⌶	SPADE or " (double quote)	♤
CIRCLE	○	SPADEFILLED	♠
CIRCLEFILLED	●	SPHERE	●
CLUB or %	♧	SPIRAL	✢
CLUBFILLED	♣	SQUARE	□
CONE	▲	SQUAREFILLED	■
CROSS	✚	STAR or =	✳
CROSSFILLED	✚	HEX	⌑
CUBE	■	STARFILLED	★
CYLINDER	■	TACK	⊤
DIAMOND or $	◇	TILDE	∿
DIAMONDFILLED	◆	TRIANGLE	△
DOT	●	TRIANGLEDOWN	▽

Value	Symbol	Value	Symbol
EARTH or +	⊕	TRIANGLEDOWNFILLED	▼
EARTHFILLED	●	TRIANGLEFILLED	▲
FEMALEFIGURE	🚺	TRIANGLELEFT	◁
FEMALEFIGUREFILLED	🚺	TRIANGLELEFTFILLED	◀
FLAG	⚐	TRIANGLERIGHT	▷
FLAGFILLED	⚑	TRIANGLERIGHTFILLED	▶
GREATERTHAN	>	UNION	∪
HASH	#	VENUS or * (asterisk)	♀
HEART or #	♡	VENUSFILLED	♀
HEARTFILLED	♥	WEAVE	⌗
HEX	⌘	WIDEARROW	⇧
HEXAGON	⬡	WIDEARROWDOWN	⇩
HEXAGONFILLED	⬢	WIDEARROWDOWNFILLED	⬇
HOME	⌂	WIDEARROWFILLED	⬆
HOMEDOWN	⛉	WIDEARROWLEFT	⇦
HOMEDOWNFILLED	⛊	WIDEARROWLEFTFILLED	⬅
HOMEFILLED	⛊	WIDEARROWRIGHT	⇨
HOURGLASS	⌛	WIDEARROWRIGHTFILLED	➡
HOURGLASSFILLED	⌛	X	×

Value	Symbol	Value	Symbol
IBEAM		Y	
LESSTHAN		Z	
MALEFIGURE			
MALEFIGUREFILLED			

The following table lists the symbols that are not supported by ActiveX.

Table 15.3 *Symbols That Are Not Supported by ActiveX*

Value	Symbol	Value	Symbol
PAW		<	
_ (underscore)		, (comma)	
&		/	
' (single quote)		?	
- (hyphen)		(	
@		)	
. (period)		:	

SAS Formats Supported by ActiveX

The ActiveX devices support the SAS character, numeric, and date and time formats that are listed in the following tables. For more information about the formats, see the *SAS System Options: Reference*.

Table 15.4 *Character Formats Supported by ActiveX*

$	$ASCII	$BINARY	$BYVAL	$CHAR

$EBCDIC	$HEX	$OCTAL	$QUOTE	$REVERJ
$REVERS	$UPCASE	$XPORTCH		

Table 15.5 Numeric Formats Supported by ActiveX

BEST	BESTX	BINARY	COMMA	COMMAX
D	DOLLAR	DOLLARX	E	EURO
EUROX	F	FLOAT	FRACT	HEX
IB	IBR	IEEE	IEEER	LOGPROB
MINGUO	MRB	NEGPAREN	NUMX	OCTAL
ODDSR	PB	PCPIB	PERCENT	PERCENTN
PIB	PIBR	PK	PVALUE	RB
ROMAN	S370FF	S370FHEX	S370FIB	S370FIBU
S370FPD	S370FPDU	S370FPIB	S370FRB	S370FZD
S370FZDL	S370FZDS	S370FZDT	S370FZDU	SIZEK
SIZEKB	SIZEKMG	SSN	VAXRB	WORDF
WORDS	XPORTFLT	XPORTINT	YEN	Z
ZD				

Note: The ACTIVEX and ACTXIMG devices do not honor a numeric format that is applied to the values of a base 10 logarithmic axis.

Table 15.6 Date and Time Formats Supported by ActiveX

DATE	DATEAMPM	DATETIME	DAY	DDMMYY
DDMMYYB	DDMMYYC	DDMMYYD	DDMMYYN	DDMMYYP
DDMMYYS	DOWNAME	DTDATE	DTMONYY	DTWKDATX
DTYEAR	DTYYQC	HHMM	HOUR	JULDATE
JULDAY	JULIAN	MDYAMPM	MMDDYY	MMDDYYB
MMDDYYC	MMDDYYD	MMDDYYN	MMDDYYP	MMDDYYS
MMSS	MMYY	MMYYC	MMYYD	MMYYN
MMYYP	MMYYS	MONNAME	MONTH	MONYY

NENGO	PDJULG	PDJULI	QTR	QTRR
TIME	TIMEAMPM	TOD	WEEKDATE	WEEKDATX
WEEKDAY	WORDDATE	WORDDATX	XYYMMDD	YEAR
YYMM	YYMMC	YYMMD	YYMMDD	YYMMDDB
YYMMDDC	YYMMDDD	YYMMDDN	YYMMDDP	YYMMDDS
YYMMN	YYMMP	YYMMS	YYMON	YYQ
YYQC	YYQD	YYQN	YYQP	YYQR
YYQRC	YYQRD	YYQRN	YYQRP	YYQRS
YYQS	YYQZ			

Note: The ACTIVEX and ACTXIMG devices do not support nested formats that include user-defined formats in the nesting. If you want to use a custom format with these devices and the custom format nests other formats, make sure that it nests only the formats that are provided by SAS.

Note: The ACTIVEX and ACTXIMG devices do not honor the DATATYPE=DATE option in the FORMAT procedure PICTURE statement.

Configuring Drill-Down Links with ActiveX

About ActiveX Drill-Down Links

ActiveX parameters provide a way to implement drill-down functionality and to configure interactive features. The purpose and syntax of these parameters are defined in “ActiveX Control Parameter Dictionary” on page 1643. In the ODS HTML statement, ActiveX parameters are specified with the PARAMETERS= option, as described in “ActiveX Control Parameter Dictionary” on page 1643.

The SAS/GRAPH ActiveX Control enables the URL, HTML, and Script drill-down modes for charts and maps. Drill-down functionality is not enabled for contour plots. For information about implementing these drill-down modes, see “Specifying the Drill-Down Mode” on page 210.

The following table lists the procedures and statements that generate output that can be used in ActiveX presentations with drill-down functionality.

Table 15.7 Statements Enabled for Drill-Down Functionality in ActiveX

Procedure	Statements
GBARLINE	BAR, PLOT
GCHART	HBAR, HBAR3D, VBAR, VBAR3D, PIE, PIE3D, DONUT
GPLOT	PLOT, BUBBLE
GMAP	CHORO, BLOCK, PRISM
G3D	PLOT, SCATTER

Using ODS Parameter Tags

You can use parameter tags in the ODS HTML statement to specify drill-down behavior for the ActiveX control in the ODS HTML statement. Parameters are specified in the ODS destination statement with the PARAMETERS= option as follows:

```
ODS HTML PARAMETERS=(options);
```

See "Specifying Parameters and Attributes for ActiveX" on page 1644 for a detailed description of the parameter tags and attributes that are available for use with ActiveX and Java.

Using Drill-Down Tags

You can use the following tags to specify drill-down behavior for the ActiveX control. The following table defines the drill-down tags and explains the types of graphs to which the tags can be applied.

Table 15.8 Drill-Down Tags Used by the ActiveX Control

Tag Name	Description	Definition of the Value That Follows the Tag	Applied in
G_COLOR	Use new colors for the graph elements	Name of the new color variable	Scatter plots
G_COLORV	Use the color variable from the preceding level	None	Scatter plots

Tag Name	Description	Definition of the Value That Follows the Tag	Applied in
G_DEP	Use a new dependent variable	Name of the new dependent variable	All charts
G_DEPV	Use the dependent variable from the previous level	None	All charts
G_DEPTH	Use a new depth variable	Name of the new depth variable	Vertical bar charts and scatter plots
G_DEPTHV	Use the depth variable that was used in the previous level	None	Vertical bar charts and scatter plots
G_GROUP	Use a new group variable	Name of the new group variable	Bar charts
G_GROUPV	Use the group variable that was used in the previous level	None	Bar charts
G_INDEP	Use a new independent variable	Name of the new independent variable	Charts and maps
G_INDEPV	Use the independent variable that was used in the previous level	None	Charts and maps
G_LABEL	Use a new label	Name of the new label (mapID) variable	Maps
G_LABELV	Use the same label that was used in the previous level	None	Maps
G_SUBGR	Use a new subgroup variable	Name of the new subgroup variable	Bar charts and scatter plots
G_SUBGRV	Use the same subgroup variable that was used in the previous level	None	Bar charts and scatter plots

When you specify a variable name after a tag, the name must be specified exactly how it appears in the data set. Variable names are case-sensitive in JavaScript. To find out how a variable was defined in the data set, use the CONTENTS procedure.

Specifying the Drill-Down Mode

To enable a given drill-down mode, specify a value for the parameter DRILLDOWNMODE. The DRILLDOWNMODE parameter is specified in an ODS statement. The following syntax sets the DRILLDOWNMODE parameter in the ODS statement:

ODS HTML PARAMETERS= ("DRILLDOWNMODE"="SCRIPT" | "URL" | "HTML" | “ANY”);

Here is a description of the DRILLDOWNMODE values.

SCRIPT
calls a JavaScript method that you specify in your SAS/GRAPH program. You provide the JavaScript that responds to the selected area. The data passed to the JavaScript method determines the graphic portion selected, and the appropriate action.

See “Configuring Script Drill-Down Mode” on page 214

Example “Example: Script Drill-Down Mode with ActiveX” on page 220

URL
displays URLs that are provided by the HTML= variable. The URLs identify HTML files.

Restriction If the graphics procedure that generates the graph specifies the HTML= option, then the value of the DRILLDOWNMODE= parameter is automatically set to URL. All modes specified in ODS are overridden.

See “HTML” on page 1650

HTML
generates drill-down URLs based on a substitution pattern that you specify in your SAS/GRAPH program. The ActiveX control completes the URL by inserting the specified data from the selected graph element.

```
ods html file="statepop.htm"
   parameters=("DRILLDOWNMODE"="HTML"
   "DRILLPATTERN"=%nrstr("http://www.state.{&statename}.us"));
```

The data set variable value STATENAME completes the drill-down URL.

Note Define the variable with the partial URL when creating the graphic.

Tip Use the %NRSTR macro function to mask the URL-pattern string so that SAS does not interpret variable substitution strings such as &STATENAME as macro variables.

See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 255

ANY
attempts to implement the four drill-down modes in succession until a valid web destination is found. The order is Script, URL, and HTML.

See "Specifying Parameters and Attributes for ActiveX" on page 1644 for a complete list of ODS parameters.

Understanding Variable Roles

The assignment of roles to variables determines the appearance of the resulting graph. The assignment of roles takes place in the SAS/GRAPH statement that generates the graph. One variable is always assigned the role of independent variable, and another is always assigned the role of dependent variable. Once the initial graph has been displayed in the ActiveX control, web users can change the variable roles using menu options.

Variable roles are used to configure the Local, HTML, and Script drill-down modes. The roles are assigned with parameters, using the PARAMETERS= option in the ODS statement. In the specification of a parameter, the assignment of roles is done with drill-down tags.

Removing Blank Spaces from Data Values in Substitution Strings

The drill-down modes Script (see "Configuring Script Drill-Down Mode" on page 214) and HTML (see "Configuring the Drill-Down Response in HTML and URL Modes" on page 213) use substitution strings to generate a response to drill-down actions. The substitution strings are replaced with data values. Blank spaces in those data values can produce unexpected results. To remove blank spaces from data values when those values are to be used in a substitution string, specify the PATTERNSTRIP= parameter as follows in the ODS statement:

ODS HTML FILE=*fileref-or-external-file*
PARAMETERS=("DRILLDOWNMODE"="SCRIPT" | "URL"
"PATTERNSTRIP"= | "NONE" | "YES" | "COMPRESS");

NONE
: is the default value. Any blank spaces in the data value are inserted into the substitution string.

YES
: removes all blank spaces from the end of the data value, but retains blank spaces elsewhere.

COMPRESS
: removes all blank spaces from the data value, wherever they occur.

Using Variables as Substitution Strings

When you specify a variable name as a substitution string in the HTML drill-down mode, the ActiveX control replaces the entire string with the value of the variable as

it is specified in the selected graph element. The syntax of the substitution string is as follows:

```
{&variable_name}
```

Because JavaScript is case sensitive, the name of the variable must be exactly the same as it is in the data set.

A variable name substitution string might look like this:

```
http://ourweb.com/uspop/{&statename}/poptable.htm
```

The substitution string above could be used in a web presentation that begins with a map of the United States. In response to a drill-down action in HTML mode, the value of the STATENAME variable for the selected state would be substituted into the URL. The resulting URL would point to a web page that contains a table of population information for the selected state.

In the HTML drill-down mode, you can specify variable roles or labels as substitution strings, using drill-down tags, as described in "Understanding Variable Roles" on page 211. The syntax of these substitution strings is as follows:

{&*drill-down-tag*}

where *drill-down-tag* specifies a variable role or label in the initial graph. The ActiveX control replaces the substitution string by deriving a variable name from the role or label and applying the value of that variable to the URL. The value is taken from the data that is associated with the selected graph element.

For example, a web presentation could be configured using this URL:

```
http://ourweb.com/regstaff/{&G_INDEPV}/stafflist.htm
```

When a web user selects a data element with the independent variable REGION, if the value of REGION is East, the ActiveX control displays this URL:

```
http://ourweb.com/regstaff/East/stafflist.htm
```

The default value for the DRILLPATTERN parameter is as follows:

```
{&G_INDEPV,f}{&G_GROUPV,f}{&G_SUBGRV,f}.html
```

The URL that is created points to an HTML file that is in the same directory as the top level HTML file. The name of the file is a concatenation of formatted values for the first independent, group, and subgroup variables defined in the data set.

Configuring HTML Drill-Down Mode

Using the DRILLDOWNMODE, DRILLPATTERN, PATTERNSTRIP, and DRILLTARGET Parameters

You can use the parameters DRILLDOWNMODE, DRILLPATTERN, PATTERNSTRIP, and DRILLTARGET to configure the HTML drill-down mode for the ActiveX control.

In the HTML drill-down mode, the ActiveX control responds to drill-down actions by constructing a Uniform Resource Locator (URL) using the data in the selected graph element. The URL is passed to the web browser for display.

The parameter DRILLDOWNMODE (see "ActiveX Control Parameter Dictionary" on page 1643) establishes the HTML drill-down mode. The PATTERNSTRIP parameter (see "Removing Blank Spaces from Data Values in Substitution Strings" on page 211) can be used to selectively remove blank spaces from data values before those values are applied to the URL. The DRILLTARGET parameter (see "Using Variables as Substitution Strings" on page 211) enables you to specify where you want the drill-down graph to appear in the browser.

Specify the DRILLPATTERN parameter in the ODS statement as shown in the following example:

ODS HTML
    PARAMETERS=("DRILLDOWNMODE"="HTML"
        "DRILLPATTERN"="*URL-with-substitution-strings*");

An example of this statement might look like this:

```
ods html file="statepop.htm"
   parameters=("DRILLDOWNMODE"="HTML"
      "DRILLPATTERN"=%nrstr("http://www.state.{&statename}.us"));
```

In this example, the value of the data set variable STATENAME completes the drill-down URL.

When ODS is configured as shown above, the ActiveX control dynamically generates URLs in response to drill-down actions. The ActiveX control replaces the substitution strings with data values from the graph element that was selected in the drill-down action. The *URL-with-substitution-strings* value can include multiple substitution strings. Substitution strings can include combinations of variable names, variable roles or labels, and drill-down tags. For details, see "Using Variables as Substitution Strings" on page 211. All substitution strings are enclosed in braces ({ and }) and begin with an ampersand character (&). The %NRSTR macro function can be used to mask the URL-pattern string so that SAS does not interpret the variable substitution strings as macro variables.

If a variable name is specified as a substitution string in drill-down mode; the ActiveX control replaces the string with the value of the selected data item.

Configuring the Drill-Down Response in HTML and URL Modes

In the HTML and URL drill-down modes, you can specify the parameter DRILLTARGET to specify where you want the web browser to display drill-down graphs. By default, the ActiveX control displays drill-down graphs in a new web browser window.

Specify the DRILLTARGET parameter in the ODS statement as shown in the following example:

ODS HTML PARAMETERS=(PARAMETERS=("DRILLDOWNMODE"="HTML" "DRILLTARGET"="_blank" | "_self" | "_parent" | "_top" | *any_named_target*);

Note: The predefined target values must be entered in lowercase.

Each value does the following:

_blank
: displays the drill-down graph in a newly opened, unnamed browser window.

_self
: displays the drill-down graph in the same frame or window as the initial graph. This is the default behavior in most browsers.

_parent
: displays the drill-down graph in the parent frame in a frame set. If no frames are defined, this value is the same as _SELF.

_top
: displays the drill-down graph in the full browser window, thereby replacing any frames that were defined in that window.

target_name
: displays the drill-down graph in the frame or browser window named *target_name*.

Configuring Script Drill-Down Mode

Using the DRILLDOWNMODE, DRILLFUNC, PATTERNSTRIP, and DRILLTARGET Parameters

You can use the parameters DRILLDOWNMODE, DRILLFUNC, PATTERNSTRIP, and DRILLTARGET to configure the Script drill-down mode for the ActiveX control. The Script drill-down mode enables you to execute a JavaScript callback method in response to drill-down actions. You use PUT statements to write the callback method into the HTML output file. Some experience with JavaScript is therefore required.

The syntax used to implement the Script drill-down mode is specified in the ODS statement as shown in the following example:

ODS HTML PARAMETERS=("DRILLDOWNMODE"="SCRIPT" "DRILLFUNC"="*method*");

The parameter DRILLDOWNMODE (see “Specifying the Drill-Down Mode” on page 210) establishes the Script drill-down mode. The DRILLFUNC parameter specifies the name of the JavaScript callback method that is executed in response to drill-down actions.

In response to a drill-down mode. The DRILLFUNC parameter specifies the name of the JavaScript callback method that is executed in response to drill-down actions.

In response to a drill-down action, the ActiveX control generates an array of arguments that is to be passed into the callback method. The array contains all of

the data in the array as it generates its output. As the callback method terminates, it might return an object. The ActiveX control ignores this object.

To invoke the callback method, the ActiveX control issues netscape.javascript.JSObject.call() in the following form:

PUBLIC OBJECT CALL(STRING *method-name*, OBJECT *argument-array-name*[])

The *method-name* argument is the name of the callback method that you define in JavaScript in your program. The ActiveX control supplies the *argument-array-name*.

Working with the Array of Elements

Understanding the structure of the array of arguments is important for you to be able to access those elements in your callback method. The elements in the array represent all of the variables and values that are represented by the graph element that was selected in the drill-down action. The data is labeled in the array using drill-down tags. The tags identify variable roles or labels and values. For details, see "Using Drill-Down Tags" on page 208 and "Understanding Variable Roles" on page 211.

The first element in the array of arguments is the name of the ActiveX control. The second element in the array is the name of a file. The name of that file is derived from the variable roles in the graph at the preceding drill-down level, using the following substitution string:

```
{&G_INDEPV,f}
{&G_GROUPV,f}
{&G_SUBGRV,f}.html
```

The file name is a concatenation of the formatted values of the independent, group, and subgroup variables in the graph at the preceding drill-down level.

Note: The file name and file type are provided as a convenience. If you use this file name and file type, then you must create the actual file and provide its content.

The remaining elements in the array consist of drill-down tags, and the data that is associated with those tags in the graph element that was selected in the drill-down action. Each variable is represented by triplet pairs of arguments in the array:

tag variable_name

*tag*V *variable_value*

*tag*V,F *formatted_value*

Implementing Script Drill-Down Mode

To implement Script drill-down mode, use PUT statements in a DATA step to write a JavaScript callback method into the HTML output file. For information about writing JavaScript, refer to JavaScript tutorials that are available on the internet.

Formatting Data Values in Script Drill-Down Mode

For Script drill-down mode only, you can specify that data values are to be formatted or not formatted. By default, the values of the variables are not formatted. If the characters `,f` are appended to the end of the tag, then those values are presented in formatted form. This parameter tag specifies that the values of the independent variable **cost** are to appear in formatted form.

```
{g_inep,f}{cost}
```

The format is applied using the FORMAT statement in the DATA step or graphics procedure that generated the data for the graph. Formatted values are specified in the statement that generated the original graph. Formatted values are used for axis labels, legends, and data tips that are displayed when the mouse is positioned over a graph data element.

Configuring Data Tips with ActiveX

For output generated with the ACTIVEX graphics output device, data tips are created by default. The data tips are displayed when the mouse pointer is positioned over a graph data element. Use the TIPS=NONE parameter to suppress data tips for ActiveX. For example:

```
ODS HTML parameters=("Tips"="NONE")
```

For SAS/GRAPH procedures that support the HTML= option, you can add custom data tips to the output as described in “Adding Custom Data Tips with the HTML= Option” on page 253. To determine whether a procedure supports the HTML= option, refer to the description for that procedure in *SAS/GRAPH: Reference*.

Note: The output from the ACTIVEX graphics output device does not display custom data tips when the number of observations in the graph data exceeds 5000.

Examples: Creating Interactive ActiveX Output

About These Examples

The examples in this section describe how to create interactive graphs using the ACTIVEX device. They are valid in SAS 9.4M7 and earlier releases. Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

The examples include:

- “Example: Generating an Interactive Area Bar Chart in ActiveX” on page 217
- “Example: Generating an Interactive Contour Plot in ActiveX” on page 219
- “Example: Script Drill-Down Mode with ActiveX” on page 220

The following additional samples are available in the Sample Library:

- GWBAXBLK—Generating an Interactive Block Diagram
- GWBAXCON—Generating an Interactive Contour Plot
- GWBAXMAP—Generating an Interactive Map for the web
- GWBDRACT—ActiveX Map with Script-mode drill-down
- GWBDRAC2—Another ActiveX Map with JavaScript

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH documentation page on support.sas.com.

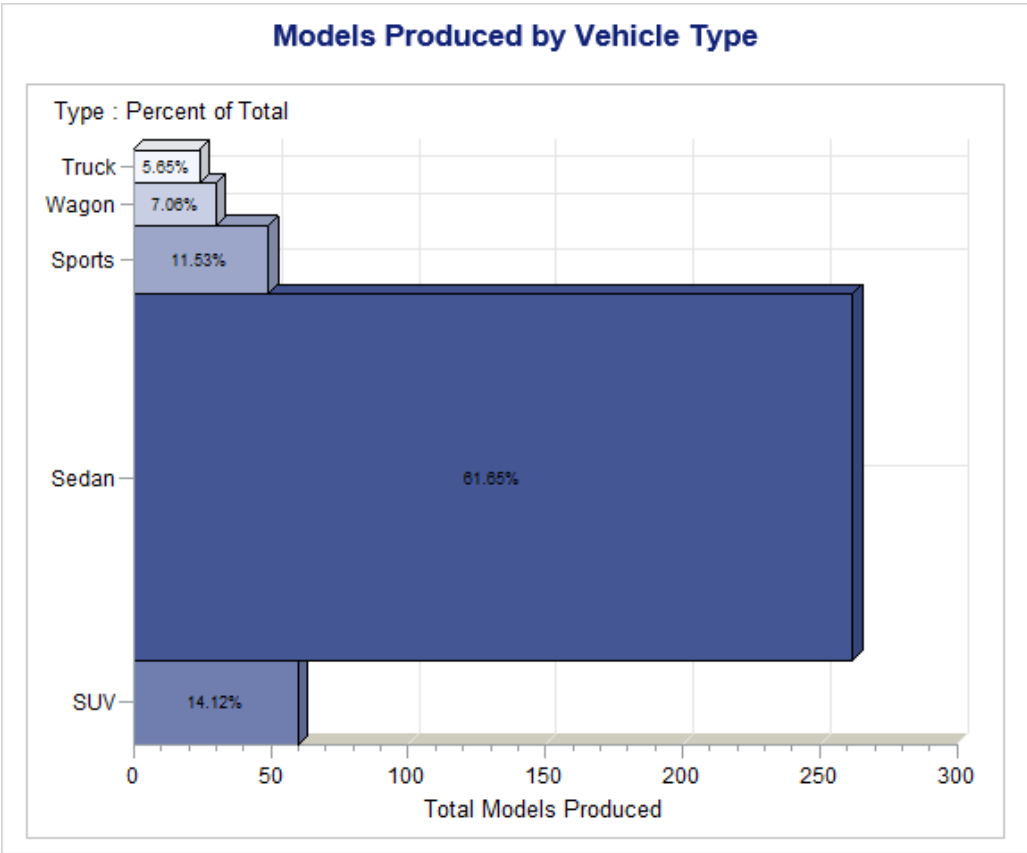
Example: Generating an Interactive Area Bar Chart in ActiveX

Here is an example that demonstrates how the ODS HTML statement can be combined with the ACTIVEX device to generate an interactive area bar chart. Here is the example program code.

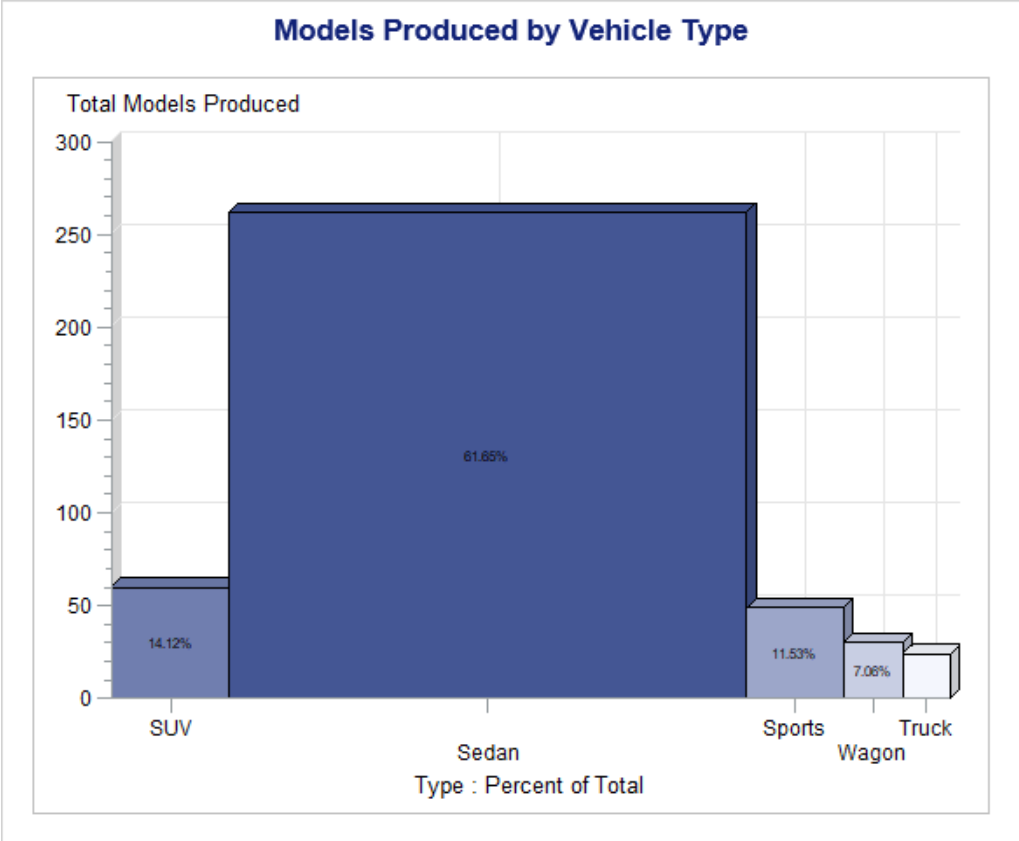
```
/* Get the model count by vehicle type */
proc freq data=sashelp.cars order=data noprint;
   tables Type / out=work.carTypes;
   where type ne "Hybrid";
run;
```

```
/* Create the area bar chart */
ods _all_ close;
goptions reset=all dev=activex xpixels=520 ypixels=400 border;
title "Models Produced by Vehicle Type";
ods html path="." file="carmodels.html";
proc gareabar data=work.carTypes;
   hbar3d type*percent /
      sumvar=count widthstat=percent;
   label count="Total Models Produced" percent="Percent of Total";
run;
quit;
ods html close;

ods html; /* Not required in SAS Studio */
```

The following figure shows the result.



The SAS/GRAPH ActiveX Control provides a pop-up menu that enables you to change many aspects of the graph, including the chart type. For example, to change the pie chart to a bar chart, right-click the graph, and then select **Chart Type** ⇨ **Vertical Bar** in the pop-up menu. The chart changes from a horizontal area bar chart to a vertical area bar chart as shown in the following figure.



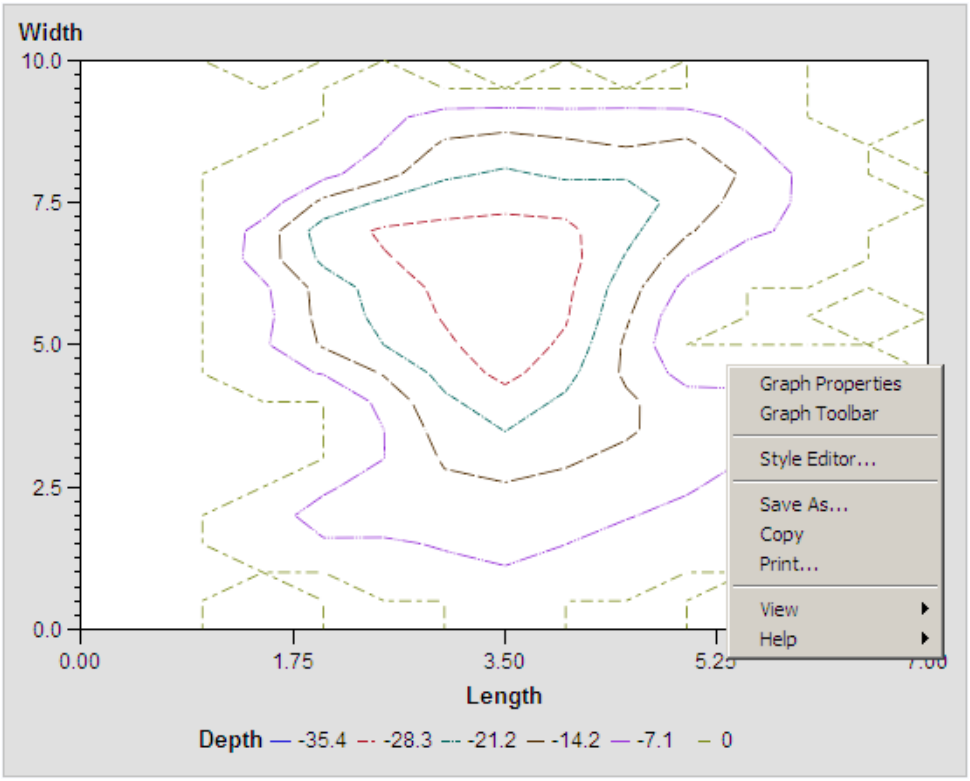
Example: Generating an Interactive Contour Plot in ActiveX

Here is an example that displays a contour plot of water depth in a lake. The SAS/GRAPH ActiveX Control lets you manipulate many of the aspects of the plot using the pop-up menu that is displayed when you right-click.

Here is the example program code.

```
goptions reset=all border device=activex;
ods _all_ close;
ods html path="output-path" file="contour.html" style=default;
proc gcontour data=sashelp.lake;
   plot width * length = depth;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */
```

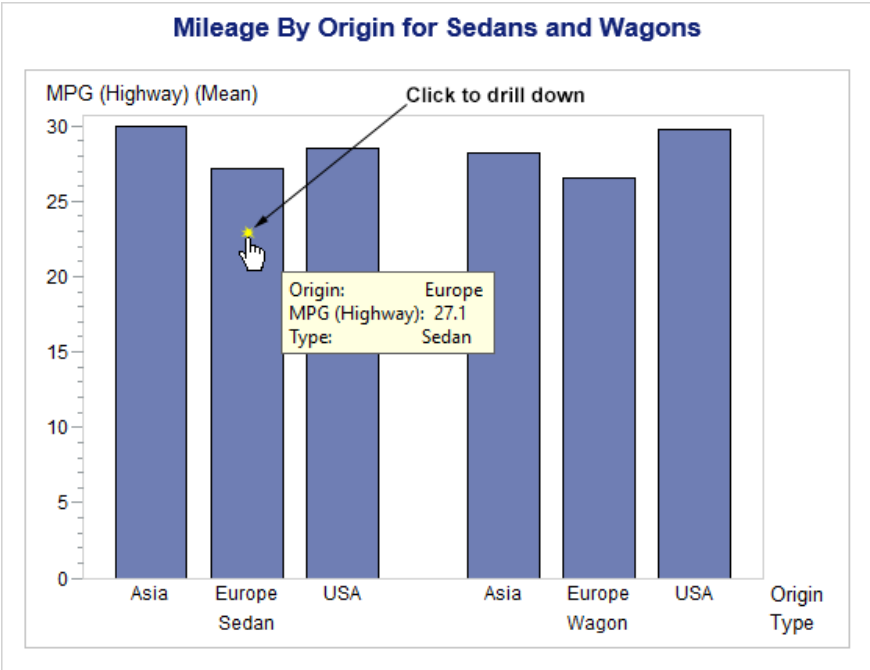
The following figure shows the result and the pop-up menu that appears when you right-click the graph.



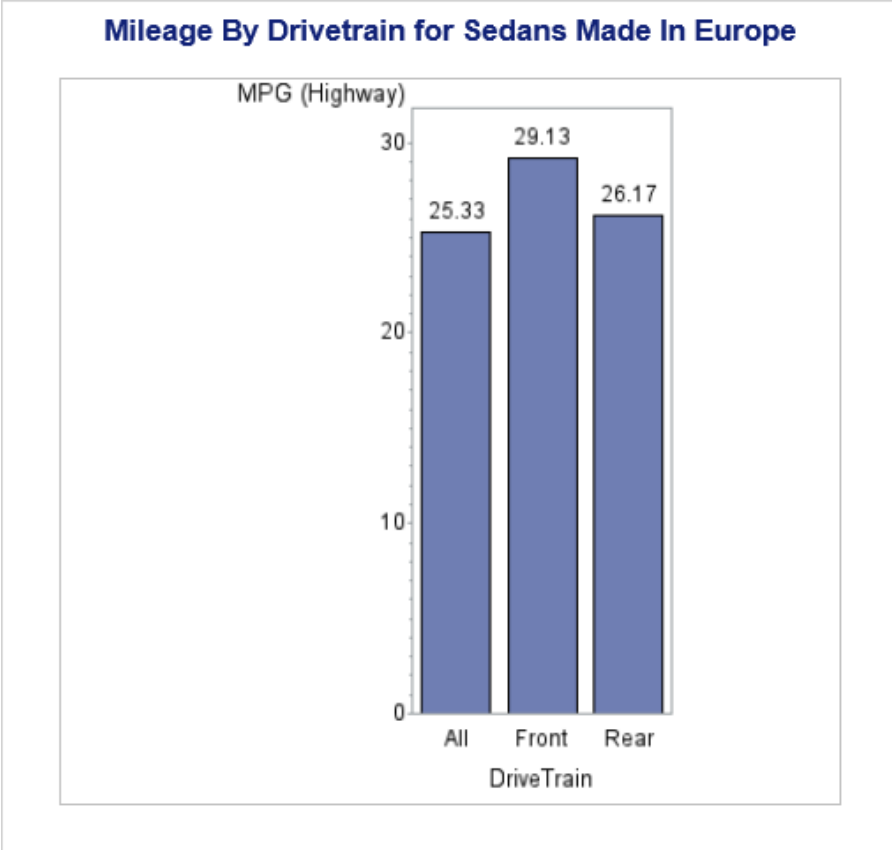
Example: Script Drill-Down Mode with ActiveX

Here is an example that shows how to implement the script drill-down mode in the SAS/GRAPH ActiveX Control. This example generates the drill-down vertical bar chart shown in the following figure.

Figure 15.2 *ActiveX Drill-down Graph of Mean Highway MPG for Sedans and Wagons by Origin*



SAS/GRAPH provides data tips by default. These data tips are displayed when the cursor is over a bar. When a bar is clicked, a horizontal bar chart of mean highway mileage for sedans and wagons by drivetrain is opened in a new browser window. The following figure shows the drill-down chart that opens when the Europe bar for sedan is clicked.



To implement JavaScript drill-down functionality, PUT statements are used in a DATA step to insert code for the JavaScript function GRAPHDRILL into the drill-down chart HTML file after the file is generated. The GRAPHDRILL function generates the URL to the drill-down chart and opens the URL in a new browser window when a bar is clicked. Here is the SAS code for this example.

```
/* Specify the output path and HTML filename. */
filename gout "output-path";
%let fileurl=url-to-files; /* Terminate with a path separator such as
'/'. */
%let htmlfn=mileage.html;

/* Extract data for sedans and wagons from Sashelp.Cars. */
data cars;
   set sashelp.cars(where=(type in ("Sedan" "Wagon")));
run;

/* Generate the drill-down graphs. */
%macro gengraph(origin,type);
   ods _all_ close;
   ods html5 file="mpg_&origin._&type..html" path=gout(url="&fileurl")
      options(bitmap_mode="inline");
   title "Mileage By Drivetrain for &type.s Made In &origin";
   proc gchart data=cars(where=(origin eq "&origin" and type eq
"&type"));
      hbar3d drivetrain / sumvar=mpg_highway type=mean mean
name="MPG_&origin";
   run;
   quit;
```

```
   ods html5 close;
%mend gengraph;

goptions reset=all device=actximg gsfname=gout
   ypixels=250 xpixels=380 nodisplay;
%gengraph(USA,Sedan);
%gengraph(USA,Wagon);
%gengraph(Europe,Sedan);
%gengraph(Europe,Wagon);
%gengraph(Asia,Sedan);
%gengraph(Asia,Wagon);

/* Generate the main drill-down graph. */
ods _all_ close;
goptions reset=all device=activex nodisplay
   xpixels=500 ypixels=350;
ods html5 file="&htmlfn" path=gout(url="&fileurl")
     parameters=("DRILLDOWNMODE"="Script"
                 "EXTERNALNAME"="GIDX"
                 "DRILLTARGET"="_self"
                 "DRILLFUNC"="GraphDrill")
     attributes=("NAME"="GIDX")
     options(bitmap_mode="inline");

title "Mileage By Origin for Sedans and Wagons";
proc gchart data=cars(where=(type in ("Sedan" "Wagon")));
vbar3d origin / sumvar=mpg_highway type=mean
   group=type;
run;
quit;
ods html5 close;

/* Insert JavaScript function GRAPHDRILL into the HTML file. Function
   GRAPHDRILL generates the URL to the drill-down graph and opens
   the graph in a new browser window when a bar is clicked. */
data _null_ ;
file "%sysfunc(pathname(gout))/&htmlfn" mod;
put " " ;
put "<SCRIPT LANGUAGE='JavaScript'>" ;
put "function GraphDrill( ref )";
put "{";
put "   /* Get Origin and Type. */";
put "   for(i = 2; i < GraphDrill.arguments.length; i += 2)";
put "   {";
put "      if (GraphDrill.arguments[i] == 'G_INDEPV,f' )";
put "         var origin = GraphDrill.arguments[i+1];";
put "      if (GraphDrill.arguments[i] == 'G_GROUPV,f' )";
put "         var type = GraphDrill.arguments[i+1];";
put "   };";
put " ";
put "   /* Open the drill-down graph in a new browser window. */";
put "   window.open('mpg_' + origin + '_' + type + '.html',";
put "               ' ', 'height=350, width=550');";
put "};";
put "</SCRIPT>";
run ;
```

```
goptions reset=all;
ods html; /* Not required in SAS Studio. */
```

Example: URL Drill-Down Mode with ActiveX

The behavior is the same. Clicking a bar in the drill-down graph opens a drill-down graph in a new browser windows as shown in Figure 15.2 on page 221. Instead of using a JavaScript function to generate drill-down URLs, in this example, the URLs are added to the plot data in a DATA step. The HTML= option is then used in the VBAR3D statement to specify the link URL variable. Here is the code for this example.

```
/* Specify the output path and HTML filename. */
filename gout "output-path";
%let fileurl=url-to-files; /* Terminate with a path separator such as
'/'. */
%let htmlfn=mileage.html;

/* Extract data for sedans and wagons from Sashelp.Cars
   and add a URL variable. */
data cars;
   length url $80;
   set sashelp.cars(where=(type in ("Sedan" "Wagon")));
   url="href='&fileurl.mpg_" || strip(origin) || "_" || strip(type) ||
".html'";
run;

/* Generate the drill-down graphs. */
%macro gengraph(origin,type);
   ods _all_ close;
   ods html5 file="mpg_&origin._&type..html" path=gout(url="&fileurl")
      options(bitmap_mode="inline");
   title "Mileage By Drivetrain for &type.s Made In &origin";
   proc gchart data=cars(where=(origin eq "&origin" and type eq
"&type"));
      hbar3d drivetrain / sumvar=mpg_highway type=mean mean
name="MPG_&origin";
   run;
   quit;
   ods html5 close;
%mend gengraph;

goptions reset=all device=actximg gsfname=gout
   ypixels=250 xpixels=380 nodisplay;
%gengraph(USA,Sedan);
%gengraph(USA,Wagon);
%gengraph(Europe,Sedan);
%gengraph(Europe,Wagon);
%gengraph(Asia,Sedan);
%gengraph(Asia,Wagon);

/* Generate the main drill-down graph. */
ods _all_ close;
```

```
goptions reset=all device=activex nodisplay
   xpixels=500 ypixels=350;
ods html5 file="&htmlfn" path=gout
   parameters=("DRILLDOWNMODE"="URL")
   options(bitmap_mode="inline");

title "Mileage By Origin for Sedans and Wagons";
proc gchart data=cars(where=(type in ("Sedan" "Wagon")));
vbar3d origin / sumvar=mpg_highway type=mean
   group=type html=url;
run;
quit;
ods html5 close;

goptions reset=all;
ods html; /* Not required in SAS Studio. */
```

Example: HTML Drill-Down ActiveX

The behavior is the same. Clicking a bar in the drill-down graph opens a drill-down graph in a new browser windows as shown in Figure 15.2 on page 221. Instead of using a JavaScript function to generate drill-down URLs or adding URLs to the plot data, in this example, the specified URL pattern incorporates the value of the Origin and Type variables in the plot data to generate the drill-down links dynamically. Because variable Origin is the independent variable, the origin is obtained from parameter G_INDEPV. Because variable Type is the group variable, type is obtained from parameter G_GROUPV. The %NRSTR macro function is used to mask &G_INDEPV and &G_GROUPV in the URL-pattern string so that SAS does not interpret them as macro variables. Here is the code for this example.

```
/* Specify the output path and HTML filename. */
filename gout "output-path";
%let fileurl=url-to-files; /* Terminate with a path separator such as
'/'. */
%let htmlfn=mileage.html;

/* Extract data for sedans and wagons from Sashelp.Cars. */
data cars;
   set sashelp.cars(where=(type in ("Sedan" "Wagon")));
run;

/* Generate the drill-down graphs. */
%macro gengraph(origin,type);
   ods _all_ close;
   ods html5 file="mpg_&origin._&type..html" path=gout(url="&fileurl")
      options(bitmap_mode="inline");
   title "Mileage By Drivetrain for &type.s Made In &origin";
   proc gchart data=cars(where=(origin eq "&origin" and type eq
"&type"));
      hbar3d drivetrain / sumvar=mpg_highway type=mean mean
name="MPG_&origin";
   run;
   quit;
   ods html5 close;
```

```
%mend gengraph;

goptions reset=all device=actximg gsfname=gout
   ypixels=250 xpixels=380 nodisplay;
%gengraph(USA,Sedan);
%gengraph(USA,Wagon);
%gengraph(Europe,Sedan);
%gengraph(Europe,Wagon);
%gengraph(Asia,Sedan);
%gengraph(Asia,Wagon);

/* Generate the main drill-down graph. */
ods _all_ close;
goptions reset=all device=activex nodisplay
   xpixels=500 ypixels=350;
ods html5 file="&htmlfn" path=gout
   parameters=("DRILLDOWNMODE"="HTML"
               "DRILLPATTERN"=
%nrstr("mpg_{&G_INDEPV,f}_{&G_GROUPV,f}.html"))
   options(bitmap_mode="inline");

title "Mileage By Origin for Sedans and Wagons";
proc gchart data=cars(where=(type in ("Sedan" "Wagon")));
vbar3d origin / sumvar=mpg_highway type=mean
   group=type;
run;
quit;
ods html5 close;

goptions reset=all;
ods html; /* Not required in SAS Studio. */
```

16

Generating Animations

Animating Images in SAS/GRAPH

You can create either an animated GIF or SVG image output file. The following sections describe what an animated image is, the steps involved in animating your output, the system options available for altering animation attributes, and examples of creating animations.

Animating GIF Images in SAS/GRAPH

Overview of GIF Animation

Using the GIF image format, you can create an animated image that displays a sequence of images automatically from a single GIF file. These animated sequences are commonly referred to as slide shows. The display sequence repeats until the web user selects **Stop** in the web browser, until the user displays another web page, or until the animation completes the number of iterations that it is configured to run. Animated images are useful for presentations that do not need to be controlled by the web user. Finite looping is appropriate for most cases such as demonstrating trends in data over time. Infinite looping is appropriate for unattended kiosk displays.

The information provided here is limited to creating animated GIF files using the SAS/GRAPH animation device in the HTML destination.For information about creating animated GIF files in the PRINTER destination using the GIF universal printers, see "Creating Animated GIF Images and SVG Documents" in *SAS 9.4 Universal Printing*.

Animating GIF Image Output

Use the GIF graphics output device with the ODS HTML destination to generate the animated GIF file and an HTML file that displays the animation in a web browser.

Note: The generated HTML file can also contain HTML tags for drill-down links and data tips. See Chapter 17, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 249.

Main Steps for Animating GIF Image Output

To create an animated GIF image, follow these general steps in your SAS program:

1. Use an OPTIONS statement to set the appropriate system options for the animated GIF image:

   OPTIONS <ANIMDURATION=*duration* | MIN>
   <ANIMLOOP=*number*>
   <ANIMOVERLAY | NOANIMOVERLAY>;

   For information about the animation options, see "Specifying System Options to Control Animation Attributes" on page 232.

2 At the point in your program where you want your animation output to begin, use the following OPTIONS statement to start the animation output:

OPTIONS ANIMATE=START;

3 Open the ODS HTML destination. After you have set the options and opened the ODS destination, proceed with your SAS code to create each frame in your file. The animation frames are created when you run SAS procedures.

> **TIP** You can use the ANIMDURATION= animation option in between procedures to change the duration that a frame is held in view.

4 To stop your animation output temporarily in your program, use the following OPTIONS statement at the point where you want to stop:

OPTIONS ANIMATE=STOP;

To resume your animation output later in your program, use the following OPTIONS statement:

OPTIONS ANIMATE=START;

5 When all of your animation output has been generated, close the ODS HTML destination:

6 Use the following OPTIONS statement to stop the animation output:

OPTIONS ANIMATE=STOP;

Animating Image Output with the SVG Device

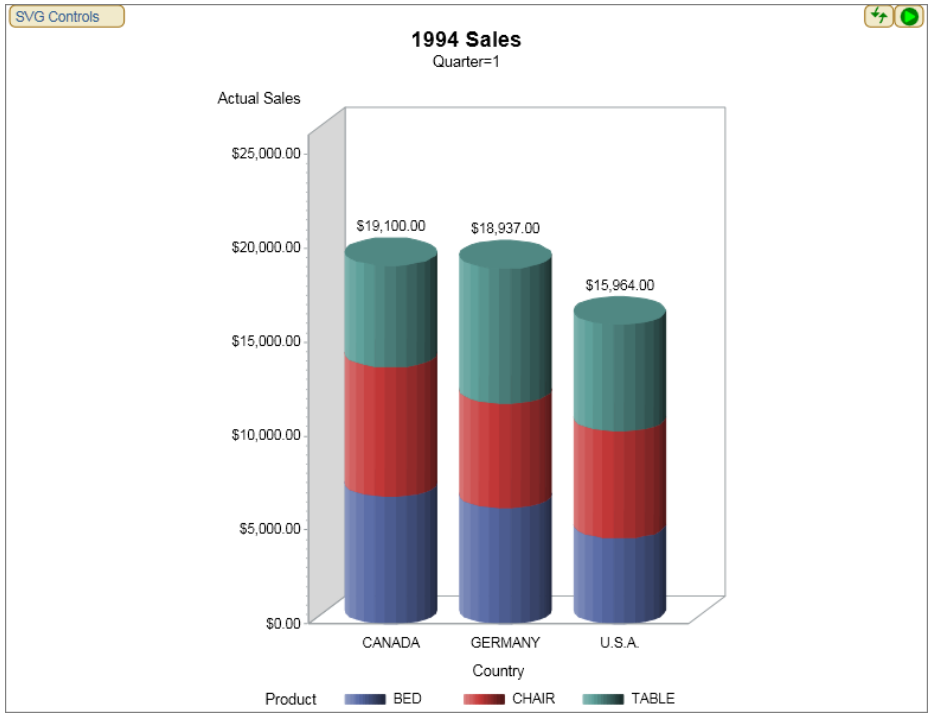
Overview of SVG Animation

You can use the SVG device to create a single SVG file that displays a sequence of images. The system options enable you to configure these animation attributes:

- the amount of time that a frame is in view
- whether frames are overlaid or are played sequentially
- the number of times an animation loop is repeated
- whether to immediately start the animation when the file is loaded in the web page
- whether a frame fades in and out of view and whether the frames are overlaid or played sequentially during the fade-in and fade-out time

The information provided here is limited to creating animated SVG files using the SAS/GRAPH animation device in the LISTING and HTML destinations. For information about creating animated GIF files in the PRINTER destination using the GIF universal printers, see "Creating Animated GIF Images and SVG Documents" in *SAS 9.4 Universal Printing*.

The following display shows one frame of an SVG animation. This animation was created by the example code shown in "Example: Creating an SVG Animation" on page 245.



By default, the SVG device adds three buttons to the top of the display.

Button	Description
SVG Controls	turns on and off the display of the buttons across the top of the SVG image.
	resets the animation to the initial image.
	pauses the animation. When the animation is paused, the button changes to .

By default, the display sequence repeats until you click or until you display another web page.

Main Steps for Animating SVG Output

To create an animated SVG image, follow these general steps in your SAS program:

1 Use the following GOPTIONS statement to specify the SVG device:

GOPTIONS DEVICE=SVG

2 Use an OPTIONS statement to set the graphics options that you want for your animation:

OPTIONS <ANIMDURATION=*duration* | MIN>
<ANIMLOOP=YES | NO | *number*>
<ANIMOVERLAY | NOANIMOVERLAY>
<SVGAUTOPLAY | NOSVGAUTOPLAY>
<SVGFADEIN=*number*>
<SVGFADEOUT=*number*>
<SVGFADEMODE=OVERLAP | SEQUENTIAL>;

Here is an example:

```
options animduration=5 animloop=yes noanimoverlay;
```

In this OPTIONS statement, the ANIMDURATION option specifies that each frame is held for 5 seconds. The ANIMLOOP option specifies to continuously repeat the animation loop. The NOANIMOVERLAY option specifies that each frame is played sequentially.

For information about the animation options, see "Specifying System Options to Control Animation Attributes" on page 232.

3 At the point in your program where you want your animation output to begin, use the following OPTIONS statement to begin generating the animation output:

OPTIONS ANIMATE=START;

4 Open the ODS HTML5, ODS HTML, or the ODS LISTING destination.

> **TIP** See "Browser Support for Viewing SVG Files" on page 169 to determine whether to use HTML5 or HTML.

5 After you have set the options and opened the ODS destination, proceed with your SAS code to create each frame in your file. The animation frames are created when you run SAS procedures.

> **TIP** You can use animation options in between procedures to change the duration that a frame is held in view and the fade-in and fade-out times. For example, you might want to hold a particular frame in view for a longer period of time. You would use the OPTIONS ANIMDURATION= statement before a procedure to increase the time that the frame is held in view.

6 To stop your animation output temporarily in your program, use the following OPTIONS statement at the point where you want to stop:

OPTIONS ANIMATE=STOP;

To resume your animation output later in your program, use the following OPTIONS statement:

OPTIONS ANIMATE=START;

7 When your animation output has been generated, close the ODS HTML5, ODS HTML, or the ODS LISTING destination.

8 Use the following OPTIONS statement to stop the animation output:

OPTIONS ANIMATION=STOP;

Adding Data Tips and Drill-down Links

Note: Animated GIF images and animated SVG images do not support data tips and drill-down links. However, using the ODS HTML output destination in combination with certain device drivers and any SAS/GRAPH procedure that supports the HTML= option enables you to use these functions. If you need to add data tips and drill-down links to your images, see Chapter 17, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 249.

For animated GIF images, the GIF graphics output device supports alpha transparency and anti-aliasing for lines.

Controlling Animations with System Options

Starting and Stopping Animation Output in Your SAS Program

The system option ANIMATE starts and stops the animation output in a SAS program. In your SAS program, to start your animation output, add an `OPTIONS ANIMATE=START` statement at the point where you want your animation to start. Add an `OPTIONS ANIMATE=STOP` statement at the point where you want your animation output to stop. You can start and stop your animation output multiple times in your SAS program, if necessary. This feature enables you to span procedure boundaries and include in your animation output selected graphics produced from multiple procedures in your SAS program.

For more information about the ANIMATE system options, see *SAS System Options: Reference*.

Specifying System Options to Control Animation Attributes

You can use system options to control various attributes of your animated image. Attributes include such things as the amount of time each image is displayed or the number of times the image sequence is repeated. You set the system options using

the OPTIONS statement before opening the ODS destination. The following table summarizes the system options that specify various attributes of animated images.

Image Output Supported	Option Name	Description
GIF, SVG	ANIMATION=	Starts or stops the creation of an animation file.
GIF, SVG	ANIMDURATION=	Specifies the amount of time that each frame in an animation is held in view.
GIF, SVG	ANIMLOOP=	Specifies whether the animation loop is played continuously or is played once. For SVG files, if ANIMLOOP=0 or YES, the animation loops continuously. All other values (NO and any number other than 0) display the animation only once.
GIF, SVG	ANIMOVERLAY | NOANIMOVERLAY	Specifies whether the animation frames are overlaid or are played sequentially. If you overlay frames, your frames require some level of transparency for the output not to be overwritten.
SVG	SVGAUTOPLAY | NOSVGAUTOPLAY	Specifies whether an SVG animation starts immediately in the web browser. If you specify NOSVGAUTOPLAY, the animation can be started by clicking .
SVG	SVGFADEIN=	Specifies the number of seconds for an SVG frame to fade into view.
SVG	SVGFADEMODE=	Specifies whether an SVG frame overlaps the previous frame or if each frame is played

Image Output Supported	Option Name	Description
		sequentially when a frame is fading in and out.
SVG	SVGFADEOUT=	Specifies the number of seconds for an SVG frame to fade out of view.

For more information about these system options, see "Dictionary of System Options" in *SAS System Options: Reference*.

To control the transparency of the background, use the CBACK= graphics option to specify an RGBA color code. See "RGBA Color Codes" on page 370.

Animating Annotation Text Labels

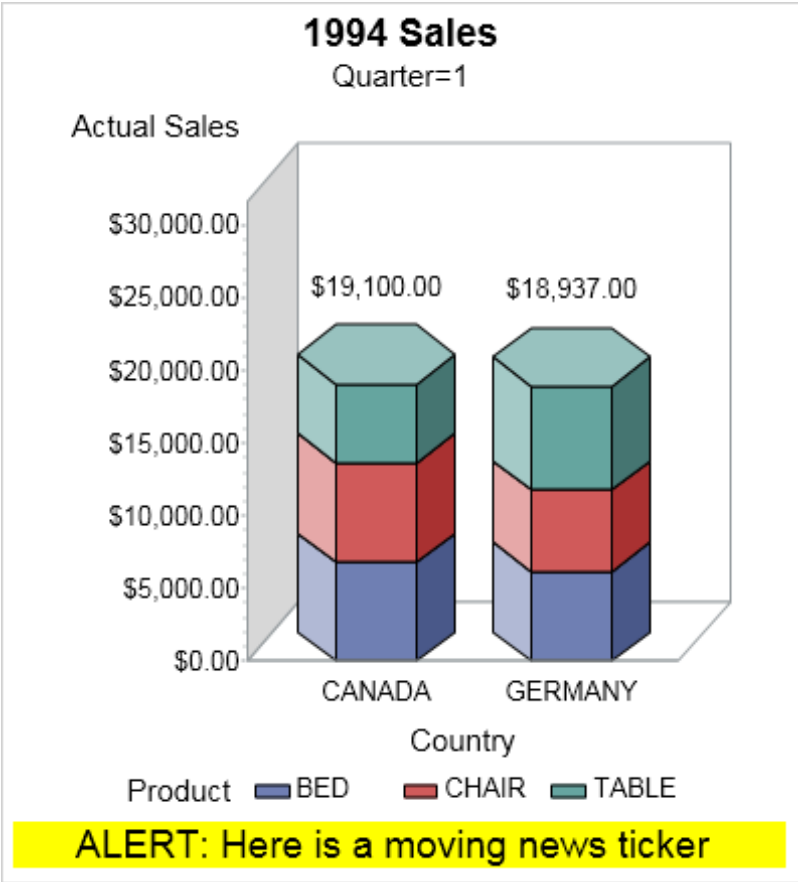
When you create text labels using the Annotate facility, you can animate those labels if you use one of the SAS/GRAPH SVG device drivers, except SVGANIM. Various animation actions are available, including moving text on a path, growing or shrinking text, moving text, and blinking text.

To animate labels, add the HTML variable to the LABEL function in the annotate data set. The HTML variable accepts attributes that specify the animation action and the parameters that control the animation.

In the following example, the HTML variable specifies that the text "ALERT: Here is a moving news ticker" moves continuously across the bottom of the screen. The code for this example can be found in Example Code 16.1.

The figure captures a still frame of the output.

Figure 16.1 *Example: Adding a Moving Text Annotation to Charts*



Here is the HTML variable statement that created the moving text.

```
html="animate=move('ALERT: Here is a moving news ticker',
  100, +0, 0, 5, repeat)";
```

The following items describe the parts of the statement:

- ANIMATE=MOVE(*parameters*) specifies the type of animation to perform. In this example, the MOVE animation is performed. The HTML variable specified must be enclosed in double quotation marks.
- *'ALERT: Here is a moving news ticker'* specifies the text that is moved. The text must be enclosed in single quotation marks. In addition, the text must match the text string that is provided for the TEXT variable in the LABEL function.
- *100* specifies the X coordinate that the text moves to in percentage units of the display or the SVG space.
- *+0* specifies the Y coordinate that the text moves to in percentage units of the display or the SVG space. The + indicates that the text moves the specified amount relative to the Y value that is specified for the label.
- *0, 5* specifies that the animation begins 0 seconds after the SVG page has been loaded and has a duration of 5 seconds.
- REPEAT specifies that the animation repeats continuously.

The attributes that determine the text color, background color, and so on, are specified separately in the data set.

Here are the main steps for producing the graph in the example.

1. In your annotation data set, after you specify FUNCTION='LABEL', add the HTML variable that specifies the animation action and associated parameters.

   For complete syntax, see "HTML Variable" on page 814.

2. Use the GOPTIONS statement to specify one of the SVG device drivers, except SVGANIM.

   ```
   goptions dev=svgview;
   ```

3. Specify the ODS HTML5, ODS HTML, or the ODS LISTING destination.

   > **TIP** If using one of the ODS HTML destinations, see "Browser Support for Viewing SVG Files" on page 169 to determine whether to use HTML5 or HTML.

4. After you have set the options and opened the ODS destination, proceed with your SAS code to create the graph. The graph code should be designed to display the annotation. For more information, see "Producing Graphics Output from Annotate Data Sets" on page 741.

SAS/GRAPH generates the output as an SVG graphic. To view the output, open the SVG file in a browser.

Note: If your browser does not render the graphic, see "Browser Support for Viewing SVG Files" on page 169.

The following code produces the output shown in the previous example.

Example Code 16.1 *Code for Adding a Moving Text Annotation to Charts*

```
/* Create a file reference for the ODS output */
filename odsout ".";

/* Specify the graphics options */
options nodate nonumber nofontembedding;
goptions reset=all device=svgview;
ods _all_ close;
ods html path=odsout file="animlabel.html";

/* Create the annotation data set */
/* that includes animation for the label */
data moving;
length color $ 8;
xsys='3'; ysys='3'; hsys='D'; position='6'; function='label';
x=0; y=5; color='black'; style='Arial'; size=18;
text='ALERT: Here is a moving news ticker';
html="animate=move('ALERT: Here is a moving news ticker',
  100, +0, 0, 5, repeat)";
output;
function='move'; x=0; y=3; hsys='3';
text=''; html='';  output;
function='bar'; x=100; y=5; hsys='3'; style='solid'; line=0;
  color='yellow';
output;
run;
```

```
/* Create the graph data by extracting information for */
/* Canada and Germany from sashelp.prdsale. */
data work.qsales;
set sashelp.prdsale(where=(country="CANADA" or country="GERMANY")
keep=Actual Country Product Quarter Year);
run;

/* Sort the data by quarter */
proc sort data=work.qsales;
by quarter;
run;

/* Generate the first set of graphs */
title1 "1993 Sales";
footnote h=2 ' ';
axis1 order=(0 to 30000 by 5000);
proc gchart data=work.qsales(where=(year=1993)) anno=moving;
vbar3d country / sumvar=actual subgroup=product sum raxis=axis1
shape=hexagon;
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;


/* Generate the second set of graphs */
title1 "1994 Sales";
proc gchart data=work.qsales(where=(year=1994)) anno=moving;
vbar3d country / sumvar=actual subgroup=product sum raxis=axis1
shape=hexagon;
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;

ods html close;
ods html; /* Not required in SAS Studio */
```

You can also use Base SAS Universal Printing to generate SVG output. See "Creating SVG Documents Using Universal Printing" in *SAS 9.4 Universal Printing*.

For another example of label text animation, see "Example: Moving Text on a Path" on page 781.

See Also

- Chapter 13, "Using SVG Graphics," on page 159
- Chapter 28, "Using Annotate Data Sets," on page 721
- "LABEL Function" on page 774

Examples: Generating Animated Images

About These Examples

This section provides examples of how to generate animated GIF images using the GIF graphics output device or animated SVG images using the SVG graphics output device. The examples include:

- "Example: Creating an Animated GIF Using BY-Group Processing" on page 238
- "Example: Creating an Animated GIF Using Run-Group Processing and Transparency" on page 240
- "Example: Using the GREPLAY Procedure to Control the Image Order in a GIF Animation" on page 243
- "Example: Creating an SVG Animation" on page 245

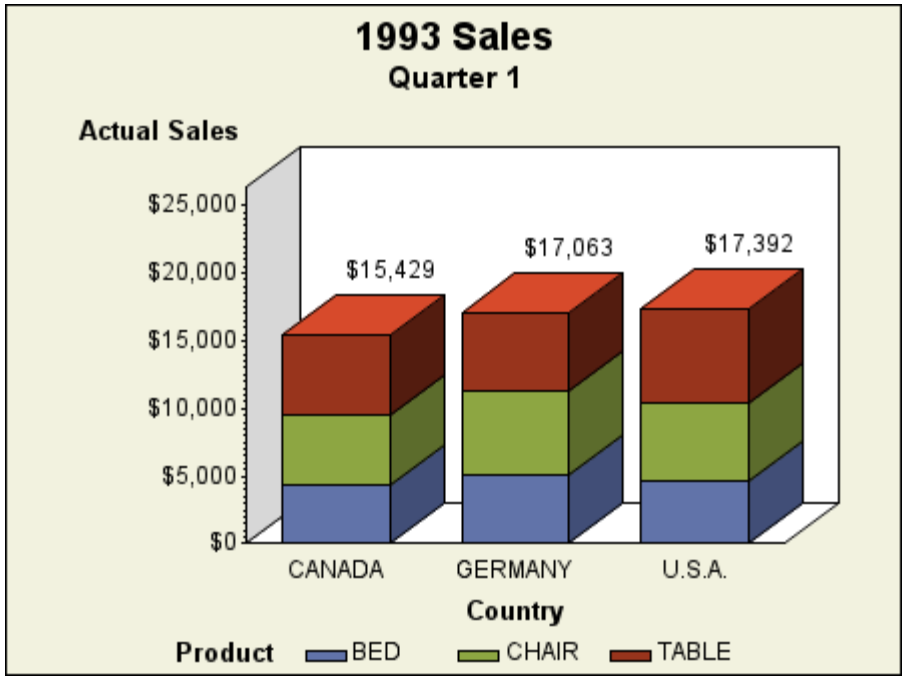
You can find an additional example, GWBANIMA, in the SAS Sample Library.

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

Example: Creating an Animated GIF Using BY-Group Processing

This example uses the GIF graphics output device to generate an animated GIF image from a SAS data set and two invocations of the GCHART procedure. Each invocation of the GCHART procedure uses BY-group processing to create a series of graphs. This example also generates an HTML file that displays the animation.

The following figure shows the first image of the animated GIF image sequence.



Here is the example program code.

```
/* Define a fileref for the ODS output */
filename odsout ".";

/* Specify a base name for the output */
%let name=gifa;

/* Specify a name for the HTML output file */
%let htmlout=gifanima.html;   /* HTML output */

/* Close all open destinations */
ods _all_ close;

/* Sort our sales data by year and quarter */
proc sort data=sashelp.prdsale out=work.sales;
    by year quarter;
run;

/* Set the graphics options */
goptions reset=all
    device=gif /* Specify the GIF shortcut device */
    border;    /* Specify a border around the graphs */

/* Set the system options for the animation */
options nobyline  /* Suppress the BY-line */
   animduration=2 /* Wait 2 seconds between graphs */
   animloop=yes;  /* Repeat the sequence continuously */

/* Create a consistent response axis */
axis1 order=(0 to 25000 by 5000);

/* Start the animation output */
options animation=start;
```

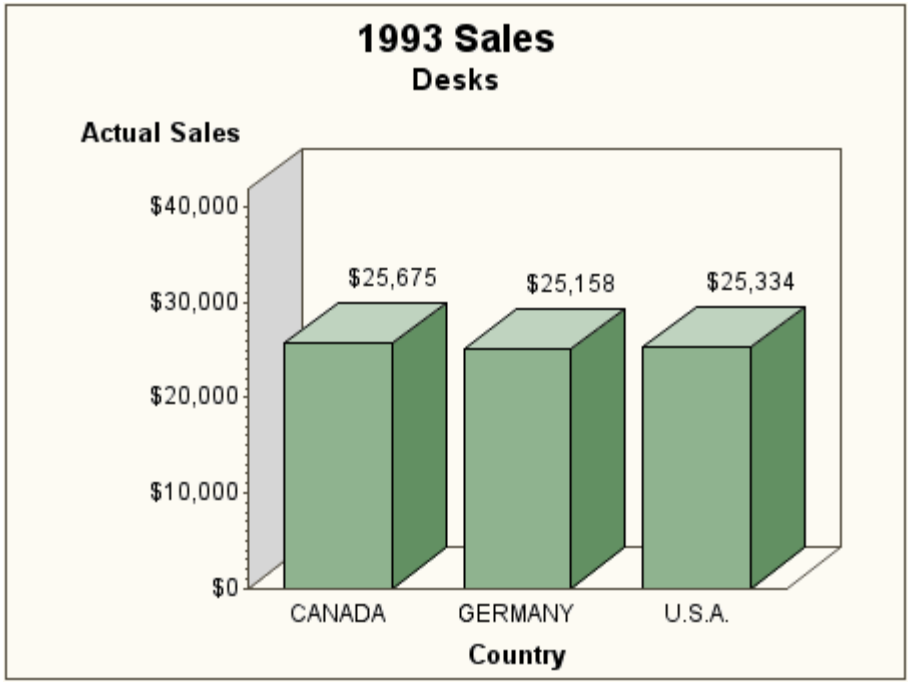
```
/* Open the ODS HTML destination */
ods html path=odsout file="&htmlout" style=harvest;

/* Generate the graphs by year and quarter */
title1 "#byval(year) Sales" lspace=1;
title2 "Quarter #byval(quarter)";
proc gchart data=work.sales;
    vbar3d country / name="&name" sumvar=actual subgroup=product
       sum raxis=axis1;
    where product in ("BED" "TABLE" "CHAIR");
    by year quarter;
    format actual dollar8.0;
run;
quit;

/* Close the ODS HTML destination */
ods html close;

/* Stop the animation output */
options animation=stop;

ods html; /* Not required in SAS Studio */
```

The animation displays the sales figures for each quarter of each of the two years. It waits 1.5 seconds between each graph. The sequence plays once initially, and then repeats two times for a total of three iterations. After the last iteration, it stops on the last graph in the sequence (1994 Q4). You can change the amount of time that each graph is displayed by changing the ANIMDURATION= system option value. You can change the number of times the sequence repeats by changing the ANIMLOOP= system option value.

Example: Creating an Animated GIF Using Run-Group Processing and Transparency

This example uses the GIF graphics output device to generate an animated GIF image file. RUN-group processing is used to show the 1993 sales data in a specific product order: desks, tables, chairs, sofas, and beds. This example also shows how to create an animated GIF image with a transparent background.

The following display shows the first image of the animated GIF image sequence.



Here is the example program code.

```
/* Define a fileref for the ODS output */
filename odsout ".";

/* Specify a name for the graphics output */
%let name=gifb;

/* Specify a name for the HTML output file */
%let htmlout=gifanimb.html; /* HTML output */

/* Close the ODS HTML destination */
ods _all_ close;

/* Specify a border and a transparent background */
goptions reset=all border cback=aFFFFFF00 gsfname=gout;

/* Set the system animation options */
options
   device=gif       /* Specify the GIF shortcut device */
   noanimoverlay    /* Display the images sequentially */
   animduration=2   /* Wait 2 seconds between graphs */
   animloop=4;      /* Repeat the sequence 4 times */

/* Delete the previously created graphs before creating new ones */
proc greplay igout=work.gseg nofs;
    delete _all_;
run; quit;

/* Create a consistent response axis */
axis1 order=(0 to 40000 by 10000);

/* Start the animation output */
options animate=start;
```

```
/* Open the ODS PRINTER destination */
ods html path=odsout file="&htmlout" style=analysis;

/* Generate the graphs using RUN-group processing */
title1 "1993 Sales" lspace=1;
proc gchart data=sashelp.prdsale(where=(year=1993));
    title2 "Desks";
    vbar3d country / name="&name" sumvar=actual sum
    raxis=axis1;
    where product="DESK";
    format actual dollar8.0;
run;
    title2 "Tables";
    vbar3d country / name="&name" sumvar=actual sum
    raxis=axis1;
    where product="TABLE";
run;
    title2 "Chairs";
    vbar3d country / name="&name" sumvar=actual sum
    raxis=axis1;
    where product="CHAIR";
run;
    title2 "Sofas";
    vbar3d country / name="&name" sumvar=actual sum
    raxis=axis1;
    where product="SOFA";
run;
    title2 "Beds";
    vbar3d country / name="&name" sumvar=actual sum
    raxis=axis1;
    where product="BED";
run;
quit;

/* Close the ODS HTML destination */
ods html close;

/* Stop the animation output */
options animate=stop;

ods html; /* Not required in SAS Studio */
```

The animation displays the sales data for Canada, Germany, and the U.S.A. It waits two seconds between each image and repeats the image sequence four times. The animation stops after the fourth iteration and displays the last graph in the sequence (beds). The images are generated using the GCHART procedure with RUN-group processing and WHERE clauses to select individual products. The CBACK= option specifies a transparent background for the graph, which allows the page background to show through. Because of the transparent graph background, the NOANIMOVERLAY option is used to display the frames sequentially. Otherwise, the previous frames can show through the current frame.

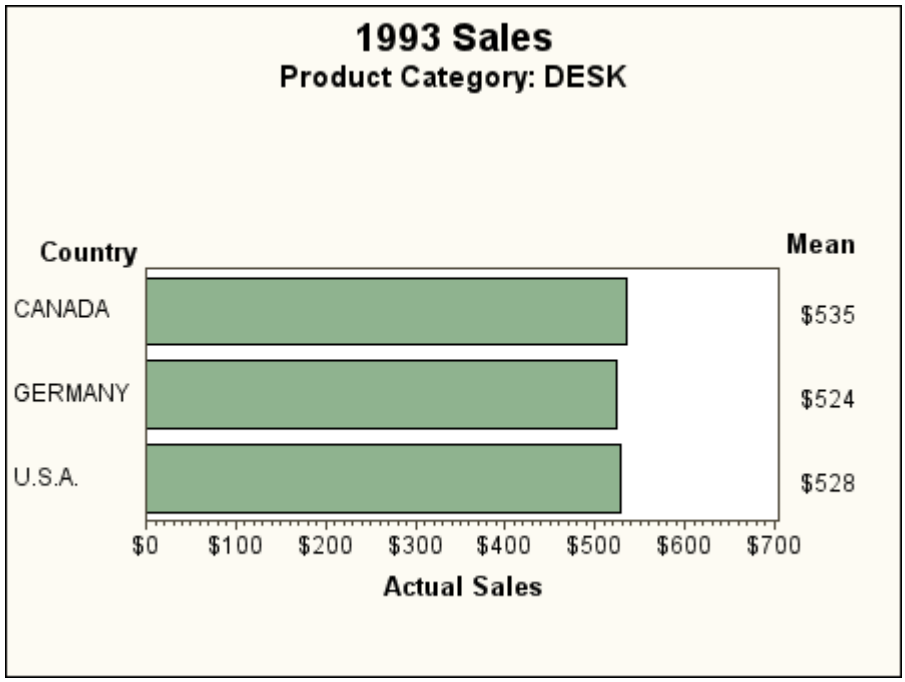
You can change the amount of time that each graph is displayed by changing the ANIMDURATION= system option value. You can change the number of iterations by changing the ANIMLOOP= system option value. You can make the background opaque by removing the CBACK= option from the GOPTIONS statement or you can

change the transparency value of the RGBA color code to vary the degree of transparency.

Example: Using the GREPLAY Procedure to Control the Image Order in a GIF Animation

This example uses the ODS HTML destination and the GIF graphics output device to create an animated GIF web presentation. The GCHART procedure, with BY-group processing, generates the graphs and stores them in a catalog. The GREPLAY procedure is then used with the GIF graphics output device to create an animated GIF image that plays the graphs in a specific product order: desks, tables, chairs, sofas, and beds.

The following display shows the first image of the animated GIF image sequence.



Note: When the GREPLAY procedure and the GIF graphics output device are used to create an animated GIF image, anti-aliasing for lines is not supported. In order to produce smoother lines in graphs that contain angled or curved lines, use RUN-group processing with WHERE statements to arrange the images instead of the GREPLAY procedure. See "Example: Creating an Animated GIF Using Run-Group Processing and Transparency" on page 240.

Here is the example program code.

```
/* Define a fileref for the ODS output */
filename odsout ".";

/* Specify a base name for the output */
%let name=gifc;
```

```
/* Specify a name for the HTML output file */
%let htmout=gifanimc.html;  /* HTML output */

/* Close the currently open ODS destinations */
ods _all_ close;

/* Create our data set by extracting the 1993 data from
   sashelp.prdsale */
data sales;
   set sashelp.prdsale;
   where year=1993;
run;

/* Sort the data by year and product */
proc sort data=sales;
    by product;
run;
quit;

/* Set graphics options */
goptions reset=all device=gif noborder nodisplay;

/* Delete the previously created graphs before creating new ones */
proc greplay igout=work.gseg nofs;
    delete _all_;
run; quit;

/* Create a consistent response axis */
axis1 order=(0 to 700 by 100);

/* Disable the BY-line */
options nobyline;

/* Open ODS LISTING and specify the graph style */
ods listing style=analysis;

/* Generate the graphs using BY-group processing */
title1 "1993 Sales" lspace=1;
title2 "Product Category: #BYVAL(product)";
proc gchart data=work.sales;
    hbar country / sumvar=actual type=mean name="&name" raxis=axis1
    mean meanlabel="Mean";
    by product;
    format actual dollar8.0;
run;
quit;

/* Close ODS LISTING */
ods listing close;

/* Reset the graphics options and set the device to GIF */
goptions reset=all device=gif;

/* Set the system animation options */
options animduration=2   /* Wait 2 seconds between graphs */
   animloop=yes;    /* Repeat the sequence continuously */
```

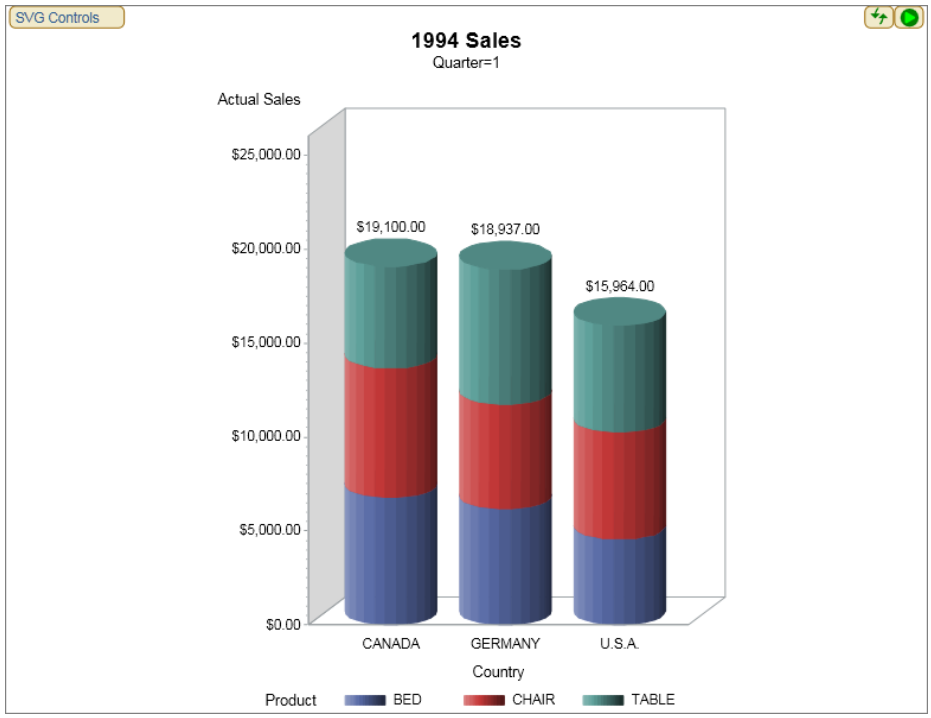
```
/* Start the animation output */
options animate=start;

/* Open the ODS HTML destination */
ods html path=odsout file="&htmout" style=analysis;

/* The graphs are to be replayed in the following product order: */
/* desks, tables, chairs, sofas, and beds */
/* This means that we have to replay the GRSEGs in the following
order:*/
/* Gifc2, Gifc4, Gifc1, Gifc3, and Gifc. */
proc greplay igout=work.gseg nofs;
    replay &name.2 &name.4 &name.1 &name.3 &name;
run;
quit;

/* Close the HTML destination */
ods html close;

/* Stop the animation output */
options ANIMATE=STOP;

/* Open ODS HTML (not required in SAS Studio) */
ods html;
```

The animation iterates through the 1993 product sales data for Canada, Germany, and the U.S.A. The animation waits two seconds between each image and repeats the animation sequence continuously.

Example: Creating an SVG Animation

This example animation shows sales figures for each of the four quarters in the years 1993 and 1994.

The following display shows one frame of the example SVG animation.



Example Code 16.2 *Creating an Animation Using the SVG Device*

```
/* Define a fileref for the ODS output */
filename odsout ".";

/* Specify a name for the HTML output file */
%let htmout=svgAnim.html;  /* HTML output */

/* Close the currently open ODS destinations */
ods _all_ close;

/* Set the graphic environment. Specify */
/*  the SVG device and start the animation output. */

goptions reset=all device=svg htext=1.5 htitle=2;
options animation=start animduration=5 noanimoverlay
        nodate nonumber;

/* Open the ODS HTML destination, specify  */
/* the filename. */

ods html path=odsout file="&htmout";

/* Create a data set by extracting data */
/* from sashelp.prdsale. Keep only the  */
/* columns needed for the graphs.       */

data work.qsales;
 set sashelp.prdsale (keep=Actual Country Product Quarter Year);
run;

/* Sort the data by quarter. When using */
/* BY groups, the data must be sorted   */
/* according to the BY variable.        */
```

```
proc sort data=work.qsales;
by quarter;
run;

/* Set the major tick mark interval */
/* on the vertical axis.            */

axis1 order=(0 to 25000 by 5000);

/* Generate graphs for 1993. */

title1 "1993 Sales";
proc gchart data=work.qsales(where=(year=1993));
vbar3d country / sumvar=actual subgroup=product sum
       raxis=axis1 shape=cylinder name='sales93';
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;

/* Generate graphs for 1994. */

title1 "1994 Sales";
proc gchart data=work.qsales(where=(year=1994));
vbar3d country / sumvar=actual subgroup=product sum
       raxis=axis1 shape=cylinder name='sales94';
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;

/* Close the ODS HTML destination */
ods html close;

/* Stop the animation output */
options animation=stop;

/* Reset the title to null  */
title1;

/* Open ODS HTML (not required in SAS Studio) */
ods html;
```

17

Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality

Overview of Enhancing Web Presentations

When you enhance a web presentation, you specify additional options, arguments, or parameters to enhance the web presentation that is generated by default. Enhancements include the following:

- Adding custom chart descriptions. See "Chart Descriptions for Web Presentations" on page 251.
- Displaying pop-up text when the mouse pointer is over a portion of the diagram. See "Data Tips for Web Presentations" on page 253.
- Adding drill-down functionality that enables you to link to other web pages. See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255 and "Generating Drill-Down Links with the Annotate Facility" on page 265.

Table 17.1 *Support for Chart Descriptions, Data Tips, and Drill-Down Functionality*

Device	Chart Descriptions		Data Tips		Drill-down Links	
	Generated by default	**Can be customized**	**Generated by default**	**Can be customized**	**Generated by default**	**Can be customized**
GIF	X	X		X		X
PNG	X	X		X		X
SVG family	X	X		X		X
ACTIVEX[1, 2, 3]			X	X		X
ACTXIMG[1, 2]	X	X	X	X		X
JAVAIMG[2]	X	X	X	X		X

1 Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

2 Starting with SAS 9.4M8, this device is no longer supported in the z/OS operating environment. If you attempt to use this device on z/OS, an error results.

3 For more information about the ACTIVEX device, see Chapter 15, "Generating Interactive ActiveX Graphics," on page 191.

Chart Descriptions for Web Presentations

What Is a Chart Description?

A chart description is the text that describes the entire chart. Default chart descriptions are generated when you use the HTML output destination, in combination with certain device driver entries. A description of your graphics output is created and stored in the HTML IMG tag ALT= attribute in your output file. You can suppress any chart description by specifying the NOALTDESC graphics option. You can display the chart description again by using the ALTDESC graphics option.

Chart descriptions are one way to meet Section 508 standards that require text equivalents for graphics elements. See "ACCESSIBLE, NOACCESSIBLE Graphics Options" on page 583 for an alternate technique.

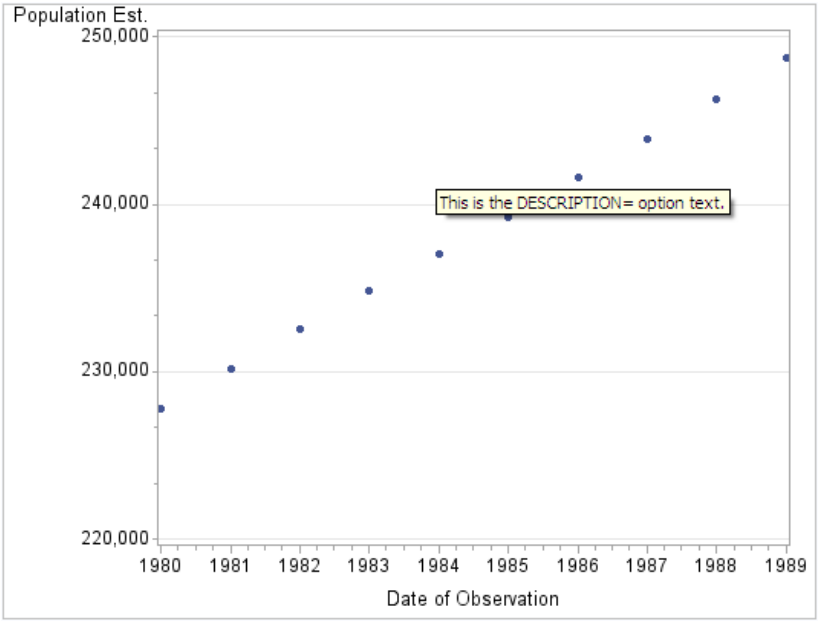
To supply your own text that describes the chart, use the DESCRIPTION= option with your procedure statement. The maximum length for a custom chart description is 256 characters. See individual procedure statements for DESCRIPTION= option details.

Example: Adding Custom Chart Descriptions

The following code generates a plot chart with a custom chart description. The custom chart description is created using the DESCRIPTION= option.

```
goptions reset=all border cback="white";
axis1 label=("Population Est.") minor=(n=2);
symbol v=dot;
proc gplot data=sashelp.citiyr;
   format pan comma7.;
   plot pan*date/ vaxis=axis1 autovref
      description="This is the DESCRIPTION= option text.";
run;
quit;
```

Figure 17.1 *Plot of Two Variables with a Custom Chart Description*



Note: Some web browsers do not display the ALT= attribute text as a data tip. Refer to the documentation for your preferred web browser to determine how the ALT= attribute text is displayed.

Chart Descriptions in PNG, GIF, ACTXIMG, and JAVAIMG Presentations

For output generated with the PNG, GIF, ACTXIMG, and JAVAIMG device drivers, using the ALTDESC graphics option displays the chart description, and is set by default.

The NOALTDESC graphics option suppresses the display of the chart description.

Specifying DESCRIPTION= provides content to the ALT attribute of your HTML file and replaces the default chart description content. The description value is limited to 256 characters. Chart descriptions are not supported in presentations generated by the Java and ActiveX device drivers.

Note: DESCRIPTION=" " can also be used to suppress the chart description.

Chart Descriptions in SVG, SVGT, SVGView, and SVGZ Presentations

For output generated with the SVG, SVGT, SVGView, and SVGZ device drivers, using the ALTDESC graphics option displays the chart description, and is set by default.

The NOALTDESC graphics option suppresses the display of the chart description.

Specifying DESCRIPTION= provides content to the feMerge element of your output file and replaces the default chart description content. The description value is limited to 256 characters.

Note: DESCRIPTION=" " can also be used to suppress the chart description.

Data Tips for Web Presentations

What Is a Data Tip?

A data tip is a data value or detailed information that is displayed as pop-up text when a user positions a mouse pointer over an element in a graph. A data tip typically displays the data value that is represented by a bar, a plot point, or some other data element. Data tips are created by default, and custom data tips are supported when using the HTML output destination in combination with certain device drivers.

Note: The information displayed in a data tip can often be read by screen reader software applications used by the visually impaired.

Adding Custom Data Tips with the HTML= Option

You can add custom data tips to the output of any SAS/GRAPH procedure that supports the HTML= option. The default device for the HTML destination is PNG, so the output of the following code is an HTML file that references a PNG image file.

To add custom data tips:

1. Add a data tip variable to the data set. In the example that follows, the data tip variable is named *rpt*.

2 Assign values to the data tip variable using the following form:

```
'title="data tip"'
```

Note: Some web browsers display text that is specified with the ALT= attribute as data tips. In that case, you can use the ALT= attribute instead, but be aware that the data tips might not be displayed in other browsers. Refer to the documentation for your preferred web browser to determine how the TITLE= and ALT= attribute text is displayed.

3 Add HTML=*data-tip-variable* to your procedure's statement.

When the mouse pointer is positioned over a graph data element, the browser displays the data tip. For an example, see "Example 1: Adding Custom Data Tips to a Graph in a Web Presentation" on page 267.

Data Tips in PNG, GIF, SVG, SVGT, SVGView, and SVGZ Presentations

The following devices do not generate data tips by default:

PNG	SVGT
GIF	SVGView
SVG	SVGZ

For SAS/GRAPH procedures that support the HTML= option, you can add custom data tips to the output as described in "Adding Custom Data Tips with the HTML= Option" on page 253.To determine whether a procedure supports the HTML= option, refer to the description for that procedure in Part 7, "SAS/GRAPH Procedures," on page 883

Data Tips in ActiveX and Java Presentations

For output generated with the ActiveX and Java device drivers, data tips are created by default. The data tips are displayed when the mouse pointer is positioned over a graph data element. Use the TIPS=NONE parameter to suppress data tips for ActiveX and Java. For example:

```
ODS HTML parameters=("Tips"="NONE")
```

For SAS/GRAPH procedures that support the HTML= option, you can add custom data tips to the output as described in "Adding Custom Data Tips with the HTML= Option" on page 253. To determine whether a procedure supports the HTML= option, refer to the description for that procedure in Part 6: SAS/GRAPH Procedures.

Note: The ACTIVEX and ACTXIMG devices do not display custom data tips when the number of observations exceeds 5000.

Note: Terminals set to use 16-bit colors or 32-bit colors are not supported when specifying data tips for output generated using DEVICE=ACTXIMG

Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options

About the URL=, HTML=, and HTML_LEGEND= Options

The URL=, HTML=, and HTML_LEGEND= options can be used in a number of statements that generate graphs. These options are use to add drill-down links, data tips, and enhancements to web presentations that are generated with the following device drivers:

SVG	JAVAIMG
PNG	ACTIVEX
GIF	ACTXIMG

The following table shows what each of these options can add to a graph.

Option	What It Can Add to a Graph
URL=	drill-down links to the data elements
HTML=	drill-down links, data tips, and enhancements to the data elements
HTML_LEGEND=	drill-down links, data tips, and enhancements to the graph legend values

As shown in the table, both the URL= option and the HTML= option can be used to add drill-down links to the data elements of a graph. You must use either the URL= option or the HTML= option, but not both. If you just want to link your graph data elements to a URL, use the URL= option or the HTML= option. However, if you want to add data tips or other enhancements to your graph data elements, you must use the HTML= option.

Note: If you specify both the URL= and HTML= option in your SAS/GRAPH procedure statement, the URL= option is ignored. A warning message appears in the SAS log.

Link Variables and Enhancement Variables

The URL=, HTML=, and HTML_LEGEND= options identify a variable in the data set that contains the link information. The variable can contain a link URL only or it can contain other functionality, such as a data tip or a link target. If the variable contains only a link URL, it is referred to as a *link variable*. If the variable contains a data tip, link target, or other functionality, because it does more than establish a link, it is referred to as an *enhancement variable*. Only link variables can be used with the URL= option. Both link variables and enhancement variables can be used with the HTML= and HTML_LEGEND= options.

The syntax of the link variable value depends on the link option with which it is used. For the URL= option, the link value syntax is as follows:

```
"URL<#anchor name>"
```

For the HTML= and HTML_LEGEND= options, the link value syntax is as follows:

```
"href='URL<#anchor name>'"
```

An enhancement variable is used when other attributes such as TITLE= or TARGET= are included in the link value in order to provide additional functionality. The enhancement value syntax is as follows:

```
"attribute='value' <...attribute='value'>"
```

where *attribute* can be an HTML <A> tag attribute such as HREF=, TITLE=, TARGET=, or ALT=, or an SVG device driver attribute such as ONMOUSEOVER. See “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 262.

The URL= option provides the simplest means of adding drill-down links to your presentation. You specify only the URL string, and the SAS/GRAPH software codes the URL into an HREF= attribute-value pair for you. The HTML= and HTML_LEGEND= options enable you to add one or more HTML <A> tag or SVG device driver attribute-value pairs to your link value. However, you must code the attribute-value pairs yourself, which can be complicated when multiple attribute-value pairs are included.

Note: Values created for the URL= option, and values created for the HTML= and HTML_LEGEND= options are not interchangeable.

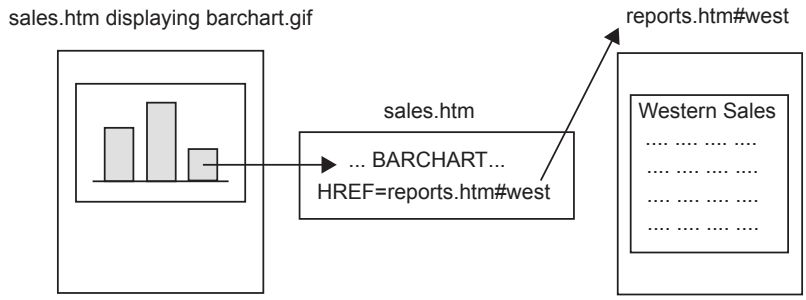
Working with Link and Enhancement Variables

Creating Link and Enhancement Variables in a Web Presentation

To create link and enhancement variables for your web presentation, you must do the following:

- define the link variable or enhancement variable, or both
- add data to the link variable or enhancement variable, or both
- for a link variable, in your SAS/GRAPH procedure statement, do one of the following:
  - identify the variable in the URL= option
  - identify the variable in the HTML= option, the HTML_LEGEND= option, or both
- for an enhancement variable, in your SAS/GRAPH procedure statement, identify the variable in the HTML= option, the HTML_LEGEND= option, or both

For example, assume that you have a web-based sales presentation that consists of a total sales bar chart in file sales.htm and a detailed sales chart for each region in file report.htm. Each bar in the total sales chart indicates the total sales for one of three corporate regions: central, southern, and western. You want to link each bar in your total sales chart to the detailed sales report for that region in file report.htm as shown in the following figure.



In addition, when a bar in the total sales chart is clicked, you want the drill-down chart in file reports.htm to be opened in a separate browser window named REGIONSALES.

To create the drill-down links for your presentation, perform the following steps:

1. Determine the link value that you need to implement your link behavior. For drill-down graphs to open in a separate browser window, your link value must include the URL and the TARGET= attribute as shown in the following example:

```
href="reports.html#west" target="regionsales"
```

See “Considerations When Using Anchored Links and Link Targets” on page 258.

2 Determine the type of variable that you need to create and the syntax for the link value. Because the value in this case includes a URL and the TARGET= attribute, you must use an enhancement variable. Therefore, the syntax for the link value is as follows:

```
"href='reports.html#west' target='regionsales'"
```

See “Link Variables and Enhancement Variables” on page 256.

3 Define your link variable in your data set. The following code fragment defines a link variable named RPT and assigns that variable a length of 80 characters.

```
data regsales;
   input Region State Sales;
   length RPT $80;
```

Note: When you create your link variable, be sure to define it with a length that is sufficient to contain your link value. There is no limit on the length of the variable.

4 In your DATA step, assign a value to your link variable as shown in the following example:

```
RPT="href='reports.html#west' target='regionsales'";
```

For information about assigning values to your link variable, see “Assigning Values to Link and Enhancement Variables” on page 259.

5 In your SAS/GRAPH procedure statement, use the HTML= option to identify the enhancement variable RPT as shown in the following code fragment.

```
title "Regional Sales";
proc gchart data=regsales;
   vbar region / sumvar=sales subgroup=region
     html=RPT;
run;
```

Considerations When Using Anchored Links and Link Targets

When you link to an anchor in the same HTML file, you can omit the HTML file name. Instead, include only the anchor name in the URL as shown in the following example:

```
RPT="href='#west' target='regionsales'"
```

However, if you want to use an SVG device to generate your web presentation, to link from the SVG document to an anchor in the parent HTML file, you must include the HTML file name and the anchor name in the link URL. If you omit the HTML file name in that case, the link will not work.

When you use the TARGET= attribute to open a drill-down link in a separate browser window, if a pop-up blocker is active in your web browser, it might prevent

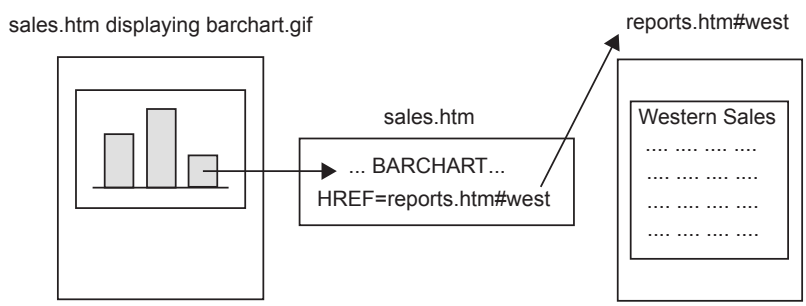
the new window from opening. In that case, you must disable or temporarily override the pop-up blocker, or you must modify the pop-up blocker settings to allow the window to open. Refer to your pop-up blocker documentation for more information.

Assigning Values to Link and Enhancement Variables

The most obvious method of adding these variables to your data set is to manually add them to the desired observations in your data set. This method is not practical or feasible in many cases. In those cases, you can use IF/THEN statements or variable substitution in your DATA step.

The following diagram shows how link variables are assigned to a bar chart. The three bars represent regional sales for a company's central, southern, and western regions.

Figure 17.2 *Links in Drill-Down Graphs*



Each bar in the chart links to an anchor tag in an HTML file named reports.htm. The anchor names in the linked file are the region names, which are Central, South, and West. A DATA step is used to create the sales data for each region. A second DATA step adds variable RPT, which contains the target report URLs. The value of Variable Region is used to provide the anchor for each report. Variable RPT is then specified in the HTML= option to provide the links. Here is the code.

```
/* Define a file reference for the ODS output */
filename odsout ".";

/* create data set REGSALES */
data regsales;
 length Region State $ 8;
 format Sales dollar8.;
 input Region State Sales;
 length rpt $80; /* the link dest. variable */
datalines;
West CA 13636
West OR 18988
West WA 14523
Central IL 18038
Central IN 13611
Central OH 11084
Central MI 19660
South FL 14541
```

```
South GA 19022
;

/* Assign HREF values to link dest. variable */
data regsales;
 set regsales;
 rpt="HREF='reports.htm#"||Region||"' target='regionsales'";
run;

goptions reset=all device=png xpixels=400 ypixels=360 border;

ods _all_ close;
ods html5 path=odsout file="sales.htm"
   options(bitmap_mode="inline");

/* Create chart that uses link targets */
title "Regional Sales";
proc gchart data=regsales;
   vbar region / sumvar=sales width=10
     html=rpt;
run;
quit;
ods html5 close;

/* Create the link targets */
goptions xpixels=320 ypixels=300;
ods html5 path=odsout file="reports.htm" anchor="south"
   options(bitmap_mode="inline");
title "Southern Sales";
proc gchart data=regsales;
 where region="South";
 vbar state /sumvar=sales width=10;
run;
quit;

ods html5 anchor="central";
title "Central Sales";
proc gchart data=regsales;
 where region="Central";
 vbar state /sumvar=sales width=10;
run;
quit;

ods html5 anchor="west";
title "Western Sales";
proc gchart data=regsales;
 where region="West";
 vbar state /sumvar=sales width=10;
run;
quit;
ods html5 close;
ods html; /* Not required in SAS Studio */
```

The following code prints data set REGSALES to the SAS log.

```
proc print data=work.regsales noobs;
run;
```

Here is the output.

Figure 17.3 REGSALES Data Set

Region	State	Sales	rpt
West	CA	$13,636	HREF='reports.htm#west' target='regionsales'
West	OR	$18,988	HREF='reports.htm#west' target='regionsales'
West	WA	$14,523	HREF='reports.htm#west' target='regionsales'
Central	IL	$18,038	HREF='reports.htm#central' target='regionsales'
Central	IN	$13,611	HREF='reports.htm#central' target='regionsales'
Central	OH	$11,084	HREF='reports.htm#central' target='regionsales'
Central	MI	$19,660	HREF='reports.htm#central' target='regionsales'
South	FL	$14,541	HREF='reports.htm#south' target='regionsales'
South	GA	$19,022	HREF='reports.htm#south' target='regionsales'

Links in SVG, PNG, and GIF Presentations

To add drill-down functionality to images generated with the SVG, SVGT, SVGZ, SVGVIEWPNG, and GIF drivers, do one of the following:

- Use the URL= option with a SAS/GRAPH procedure to add drill-down functionality to the graph data elements.
- Use the HTML= option with a SAS/GRAPH procedure to add drill-down functionality to the graph data elements.
- Use the HTML_LEGEND= option with a SAS/GRAPH procedure to add drill-down functionality to the legend entries.
- Use both the HTML= option and the HTML_LEGEND= option with a SAS/GRAPH procedure to add drill-down functionality to the legend entries.

Additional functionality is available through the use of HTML attributes with the SVG graphics devices. For more information, see “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 262.

Links in ACTXIMG and JAVAIMG Presentations

To add drill-down functionality to an image created with the ACTXIMG or JAVAIMG device drivers, use the HTML= option as described in “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 255.

Links in ACTIVEX Presentations

In SAS 9.4M7 and earlier releases, to add drill-down functionality to the ActiveX control created with the ACTIVEX device driver, use the URL= option or the HTML= option as described in "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255.

Links in Animated GIF Presentations

SAS/GRAPH does not directly support drill-down functionality for animated GIFs. To enable drill-down functionality from an animated GIF, use any third-party tools that are available to you. You can make the entire image a hotspot by including the <IMG> tag inside an <A HREF=> tag.

Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes

Additional Link Enhancements That Are Available with the SVG Graphics Devices

The SVG graphics devices provide additional functionality that you can use to enhance your drill-down links. These devices include SVG, SVGT, SVGView, and SVGZ. For information about these devices, see Chapter 13, "Using SVG Graphics," on page 159. The additional functionality that the SVG graphics devices provide is based on the mouse events that are shown in Table 17.2 on page 262.

Table 17.2 *Mouse Events and Available Functionality*

Mouse Event	Available Functionality
The mouse pointer is positioned over a linked shape.	You can add one or more of the following effects for the duration of the event: ■ display a pop-up image in a designated area of the graph ■ change the fill color of the linked shape ■ change the fill opacity of the linked shape

Mouse Event	Available Functionality
A linked shape is clicked.	In addition to opening a drill-down link, you can add one or more of the following effects: ■ change the fill color of the linked shape ■ change the fill opacity of the linked shape

The ONMOUSEOVER= and ONCLICK= HTML attributes control the mouse event effects with an SVG graphics device. These attributes are added to the HTML string with other attributes, such as HREF and ALT. The ONMOUSEOVER= HTML attribute controls the effects that occur when the mouse pointer is positioned over a linked shape. The ONCLICK= HTML attribute controls the effects that occur when a linked shape is clicked.

Note: The ONMOUSEOVER= and ONCLICK= HTML attributes are supported only by the SVG graphics devices. These attributes are ignored when other graphics devices are used.

Using the ONMOUSEOVER= HTML Attribute

Use the ONMOUSEOVER= HTML attribute in your HTML= option value to display a pop–up image, change the shape fill color, and change the shape fill opacity when the mouse pointer is positioned over a linked shape. The effects that are provided for a linked shape by the ONMOUSEOVER= HTML attribute persist only as long as the mouse pointer is positioned on that shape. The syntax of the ONMOUSEOVER= HTML attribute is as follows:

ONMOUSEOVER=*function(s)*

To achieve the effects that you want, you can specify one or more of the functions shown in Table 17.3 on page 263 for *functions()*.

Table 17.3 *ONMOUSEOVER= HTML Attribute Functions*

Function	Description
showImage	This function displays a pop–up image in a specified area of the graph. The pop–up image is stored in a separate file.
changeColor	This function changes the color of the linked shape to a specified color.
changeOpacity	This function changes the fill opacity of the linked shape to a specified value.

The syntax of the ShowImage function is as follows:

'showImage("*image-file*", *xorigin*, *yorigin*, *image-width*, *image-height*)'

The *image-file* parameter specifies the name of the file that contains the image to be displayed. The file name must be enclosed in quotation marks. The *xorigin* and *yorigin* parameters specify the X and Y coordinates of the origin of the image, which is the lower left corner of the image. The image can be placed anywhere in the graphics output area. However, any portion of the image that extends beyond the graphics output area is clipped. The *image-height* and *image-width* parameters specify the size of the image in pixels.

The syntax of the changeColor and changeOpacity functions are as follows:

changeColor(*color*)

changeOpacity(*factor*)

The `color` parameter can be a color name, such as red, green, or blue, or an RGB specification, such as #55CC22. The factor parameter can be a value between 0 and 1, which specifies the desired degree of opacity. A value of 0 makes the shape 100% opaque.

When you specify more than one function in your ONMOUSEOVER= HTML attribute, you must separate each function with a semicolon. Here is an example of an ONMOUSEOVER= HTML attribute that displays image myimage.gif and changes the color of the linked shape to blue when the mouse pointer is positioned over the shape.

```
'ONMOUSEOVER=ShowImage("./myimage.gif",300,550,200,250); changeColor(blue)'
```

In this example, the showImage function displays image myimage.gif at coordinates (300, 550), and scales the image to 200 pixels wide by 250 pixels high. The changeColor function changes the color of the linked shape to blue. When the mouse pointer is moved off of the linked shape, the image disappears, and the shape fill color reverts to its original color.

For an additional example, see “Example 3: Enhancing an SVG Drill-Down Presentation Using HTML Attributes” on page 273.

Using the ONCLICK= HTML Attribute

Use the ONCLICK= HTML attribute in your HTML= option value to change the shape fill color and the shape fill opacity when a linked shape is clicked. The syntax of the ONCLICK= HTML attribute is as follows:

'ONCLICK=*function(s)*'

To achieve the effects that you want, you can specify one or both of the changeColor and changeOpacity functions that are used with the ONMOUSEOVER= HTML attribute. (See Table 17.3 on page 263.) When you specify more than one function, separate each function with a semicolon. Here is an example of an ONCLICK= HTML attribute that changes the fill opacity of a linked shape to 50% when it is clicked.

```
'ONCLICK=changeOpacity(.5)'
```

Generating Drill-Down Links with the Annotate Facility

You can use the Annotate facility to generate web presentations with drill-down links in two ways: you can use PROC GANNO and an Annotate data set to generate an entire web presentation with drill-down links, or you can apply an Annotate data set to add drill-down functionality to a web presentation that is generated with the PNG, GIF, or ACTXIMG device driver.

When to Use PROC GANNO to Generate Drill–Down Links

You can use ODS, the GANNO procedure, an Annotate data set, and a PNG or GIF device driver to generate a web presentation with drill-down links. This method of generating a drill-down presentation is preferred if you do not need to use an image from another SAS/GRAPH procedure in your web presentation. For example, you could use PROC GANNO to generate an HTML output file that showed a GIF image, with accompanying text, and a selectable label containing the text “Click Here”. Larger presentations with multiple drill-down links are also entirely feasible.

Note: Although you can use the Annotate facility to enhance ACTXIMG device-generated output, you cannot use the GANNO procedure to create a web presentation with the ACTXIMG device driver.

For information about the GANNO procedure, see Chapter 33, “GANNO Procedure,” on page 887. To generate a drill-down graph with PROC GANNO, see “Generating Drill–Down Links with the Annotate Facility” on page 266.

When to Apply Annotate Data Sets to Web Presentations

You can use Annotate data sets to add drill-down links to web presentations generated by any procedure that uses the ANNOTATE= option. The web presentation must be generated with the PNG, GIF, or ACTXIMG device driver.

Using an Annotate data set to add drill-down links is preferable in the following circumstances:

- when you cannot add drill-down functionality by other means. Some SAS/GRAPH statements do not support the HTML= option, which SAS/GRAPH needs to generate an image map in the HTML output file. If the procedure does

support the ANNOTATE= option, then you can use that procedure as the basis of a web presentation with drill-down functionality.

- when you do not want web users to drill down by selecting graph elements. For example, if you did not want your web users to drill down by selecting the bars in a bar chart, you could define graphics elements with drill-down links using the Annotate facility.

To learn how to use the Annotate facility to add drill-down links to a web presentation, see “Generating Drill–Down Links with the Annotate Facility” on page 266.

Reference information about generating and applying Annotate data sets is provided in “The Annotate Function, Variable, and Macro Dictionaries” on page 724. Usage information is provided in Chapter 28, “Using Annotate Data Sets,” on page 721.

Generating Drill–Down Links with the Annotate Facility

Follow these steps if you are adding drill-down links to a web presentation or if you are generating an entire web presentations with PROC GANNO:

1. Plan your web presentation so that you know how and where you want to apply Annotate graphical elements with drill-down links. Also determine your drill-down URLs.

2. Generate an Annotate data set. Elements that can be defined as drill-down hot zones are generated by Annotate functions that use the HTML variable. To see which functions use the HTML variable, refer to Figure 28.3 on page 730. To generate the Annotate data set, see Chapter 28, “Using Annotate Data Sets,” on page 721.

3. Specify the PNG, GIF, or ACTXIMG device driver using the DEVICE= option in a GOPTIONS statement.

4. Close the currently open ODS destinations, and then open an HTML output file in ODS.

```
ods _all_ close;
ods html file="annodril.htm"
    style=analysis;
```

5. Generate a PNG or GIF image and identify the Annotate data set. Use the GANNO procedure or another SAS/GRAPH procedure that uses the ANNOTATE= option.

6. Close the HTML output file.

7. Generate any additional HTML files or images as needed to provide files that are named in drill-down URLs.

Examples of Using Annotated Data Sets

For examples of applying Annotate data sets to output, see "List of Annotate Examples" on page 746.

Examples

Example 1: Adding Custom Data Tips to a Graph in a Web Presentation

Features:

VBAR statement

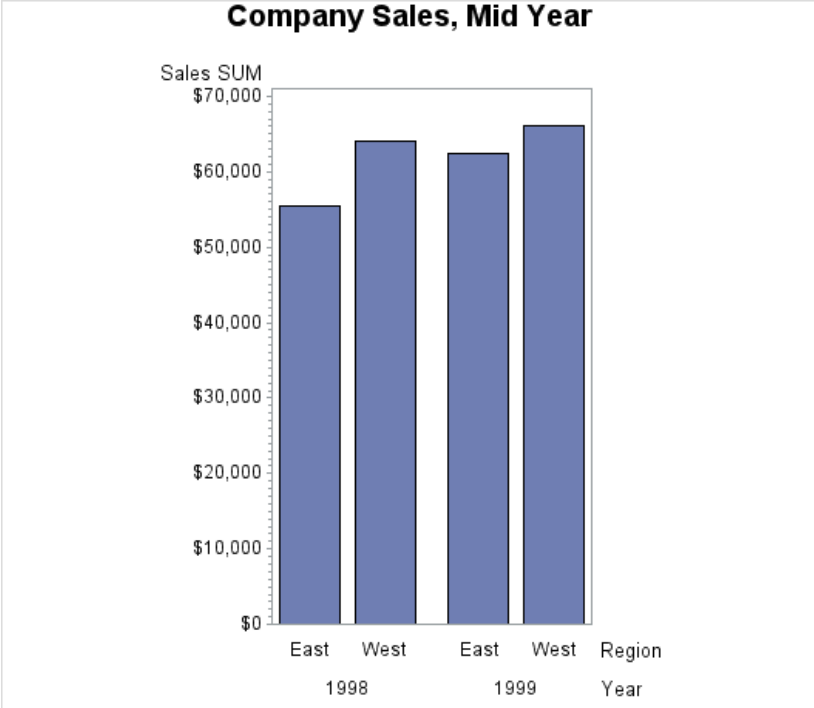
VBAR options
- GROUP=
- HTML=

ODS HTML statement options
- FILE=
- STYLE=

GOPTIONS statement option
- BORDER

This example uses the HTML= procedure option to add custom data tips to the vertical bar chart that is shown in the following figure. It generates the data tip `North Carolina and Massachusetts` for the East bars and `California and Oregon` for the West bars.

Output 17.1 *Bar Chart with Custom Data Tips*



Program

```
/* Create the temporary data set named sales. */
data sales;
   length Region $ 4 State $ 2;
   format Sales dollar8.;
   input Region State Sales Year Qtr;
   datalines;
West CA 13636 1999 1
West OR 18988 1999 1
West CA 14523 1999 2
West OR 18988 1999 2
East MA 18038 1999 1
East NC 13611 1999 1
East MA 11084 1999 2
East NC 19660 1999 2
West CA 12536 1998 1
West OR 17888 1998 1
West CA 15623 1998 2
West OR 17963 1998 2
East NC 17638 1998 1
East MA 12811 1998 1
East NC 12184 1998 2
East MA 12760 1998 2
;
/* Use an IF/THEN statement to assign values to the variable RPT. */
data sales;
   set sales;
```

```
      if state in ("NC" "MA") then
         RPT="title='North Carolina and Massachusetts'";
      if state in ("CA" "OR") then  RPT="title='California and
Oregon'";
run;

/* Reset the graphics options to their defaults and specify a border.
*/
goptions reset=all border;

/* Generate the bar chart. Add the HTML= option to associate */
/* the custom data tips with each graph element.             */
title "Company Sales, Mid Year";
proc gchart data=sales;
   vbar region / sumvar=sales
      group=year
      html=RPT;
run;
quit;
```

Example 2: Creating a Drill-Down HTML Graph

Features:

VBAR3D Statement

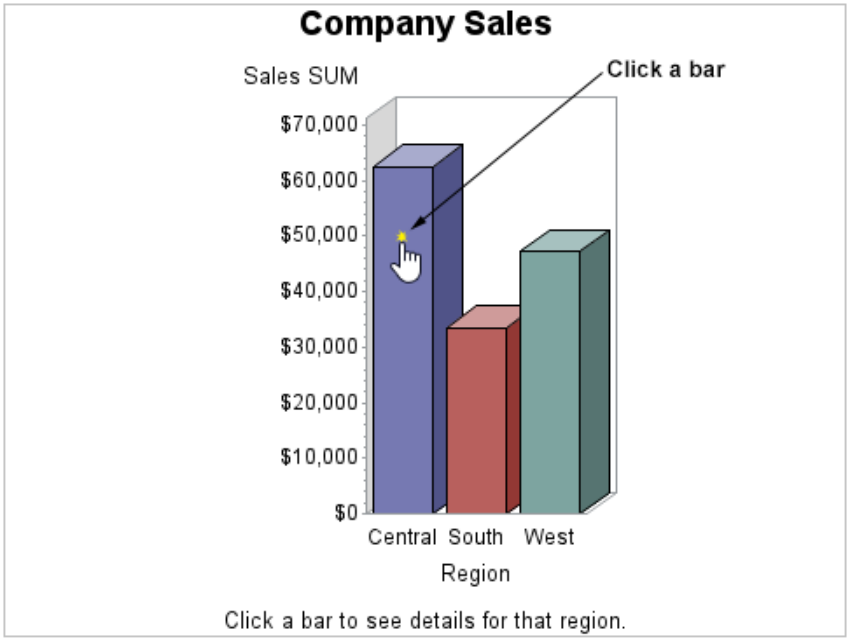
VBAR3D options
- SUMVAR=
- PATTERNID=
- WIDTH=
- HTML=

ODS HTML5 statement options
- BODY=
- PATH=
- OPTIONS

GOPTIONS statement options
- DEVICE=
- BORDER
- XPIXELS=
- YPIXELS=

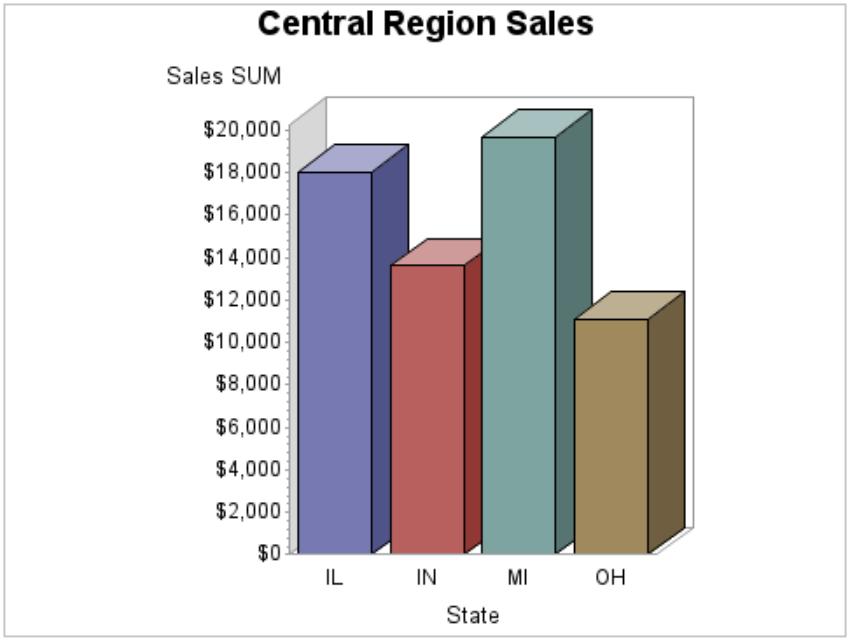
The initial graph displayed by this example is shown in the following figure. In this graph, REGION is the independent variable, and SALES is the dependent variable. Each bar is linked to a target graph for that region.

Output 17.2 *Initial Graph*



Clicking the Central bar as shown in the figure opens the Central sales graph, which is shown in the following figure.

Output 17.3 *Drill-Down Graph for the Central Region*



Program

```
filename odsout "file-path";
%let baseurl=base-url;
```

```
data regsales(drop=drilltarget);
   length Region State $8 drilltarget $30 link $80;
   format Sales dollar8.;
   input Region State Sales drilltarget;

   /* Build the link HTML string from each drill target */
   link="href='" || "&baseurl/" || trim(drilltarget) || "'
target='_self'";

/* Specify the data */
datalines;
Central IL 18038 central_sales.html
Central IN 13611 central_sales.html
Central OH 11084 central_sales.html
Central MI 19660 central_sales.html
South   FL 14541 south_sales.html
South   GA 19022 south_sales.html
West    CA 13636 west_sales.html
West    OR 18988 west_sales.html
West    WA 14523 west_sales.html
;

ods _all_ close;
goptions reset=all border device=png xpixels=480 ypixels=360;

ods html5 body="salesreport.html" path=odsout
options(bitmap_mode="inline");
title1 "Company Sales";
footnote1 j=c "Click a bar to see details for that region.";
proc gchart data=regsales imagemap=imap;
   vbar3d region / sumvar=sales width=8
   patternid=midpoint
   html=link; /* Set the HTML variable to link. */
run;
quit;
ods html5 close;

%macro createRegionGraph(region);
   goptions reset=all border device=png xpixels=480 ypixels=360;
   options nobyline;
   ods html5 body="%sysfunc(lowcase(&region))_sales.html" path=gout
      options(bitmap_mode="inline");
   title "&region Region Sales";
   proc gchart data=regsales;
      vbar3d state / sumvar=sales width=10 name="&region"
patternid=midpoint;
      where region eq "&region";
   run;
   quit;
   ods html5 close;
%mend createRegionGraph;

%createRegionGraph(Central);
%createRegionGraph(South);
%createRegionGraph(West);

goptions reset=all;
filename odsout clear;
```

Program Description

Specify the output path and the base URL for the drill-down links. Fileref ODSOUT specifies an aggregate output path for ODS output. Macro variable &BASEURL specifies the base URL for the drill-down targets.

```
filename odsout "file-path";
%let baseurl=base-url;
```

Create the data. The data includes the chart variables Region, State, and Sales, and variable Link, which contains the HREF string for the drill-down links. Variable Drilltarget is an interim variable that contains the name of the drill-down target HTML file. Variable Link incorporates Drilltarget in the Link value for each observation. Variable Drilltarget is dropped from the output data. The HREF string includes TARGET='_self', which opens the drill-down graph in the current window. You can specify a different target such as _blank for the drill-down graphs.

```
data regsales(drop=drilltarget);
   length Region State $8 drilltarget $30 link $80;
   format Sales dollar8.;
   input Region State Sales drilltarget;

   /* Build the link HTML string from each drill target */
   link="href='" || "&baseurl/" || trim(drilltarget) || "'
target='_self'";

/* Specify the data */
datalines;
Central IL 18038 central_sales.html
Central IN 13611 central_sales.html
Central OH 11084 central_sales.html
Central MI 19660 central_sales.html
South   FL 14541 south_sales.html
South   GA 19022 south_sales.html
West    CA 13636 west_sales.html
West    OR 18988 west_sales.html
West    WA 14523 west_sales.html
;
```

Close all open ODS destinations, and then set the graphics options.

```
ods _all_ close;
goptions reset=all border device=png xpixels=480 ypixels=360;
```

Create the initial drill-down chart. Option PATH= in the ODS HTML5 statement specifies fileref ODSOUT, which sends all of the ODS output to the specified path. The OPTIONS option specifies inline bitmap graphics, which embeds the images inline in the HTML file. This results in only one output file, the HTML file. The initial drill-down chart is a bar chart of Region and Sales. Variable Link provides the HTML for each bar's drill-down link.

```
ods html5 body="salesreport.html" path=odsout
options(bitmap_mode="inline");
title1 "Company Sales";
footnote1 j=c "Click a bar to see details for that region.";
proc gchart data=regsales imagemap=imap;
   vbar3d region / sumvar=sales width=8
   patternid=midpoint
   html=link; /* Set the HTML variable to link. */
```

```
run;
quit;
ods html5 close;
```

Create a macro that creates a graph for a region. A macro provides an efficient way to create multiple graphs that are similar. This macro generates a bar chart of State and Sales for a specified region. Only one output file, the HTML file, is generated for each region graph.

```
%macro createRegionGraph(region);
   goptions reset=all border device=png xpixels=480 ypixels=360;
   options nobyline;
   ods html5 body="%sysfunc(lowcase(&region))_sales.html" path=gout
      options(bitmap_mode="inline");
   title "&region Region Sales";
   proc gchart data=regsales;
      vbar3d state / sumvar=sales width=10 name="&region"
patternid=midpoint;
      where region eq "&region";
   run;
   quit;
   ods html5 close;
%mend createRegionGraph;
```

Use the macro to generate a drill-down graph for each region.

```
%createRegionGraph(Central);
%createRegionGraph(South);
%createRegionGraph(West);
```

Reset the graphics options, and clear fileref ODSOUT.

```
goptions reset=all;
filename odsout clear;
```

Example 3: Enhancing an SVG Drill-Down Presentation Using HTML Attributes

Features:

VBAR statement

VBAR options:
- HTML=
- NAME=
- PATTERNID=
- WIDTH=

GREPLAY DELETE statement

BY-group processing

GOPTIONS statement options:
- BORDER
- DEVICE=
- XPIXELS=
- YPIXELS=

ODS HTML statement options:
- ANCHOR=
- BODY=

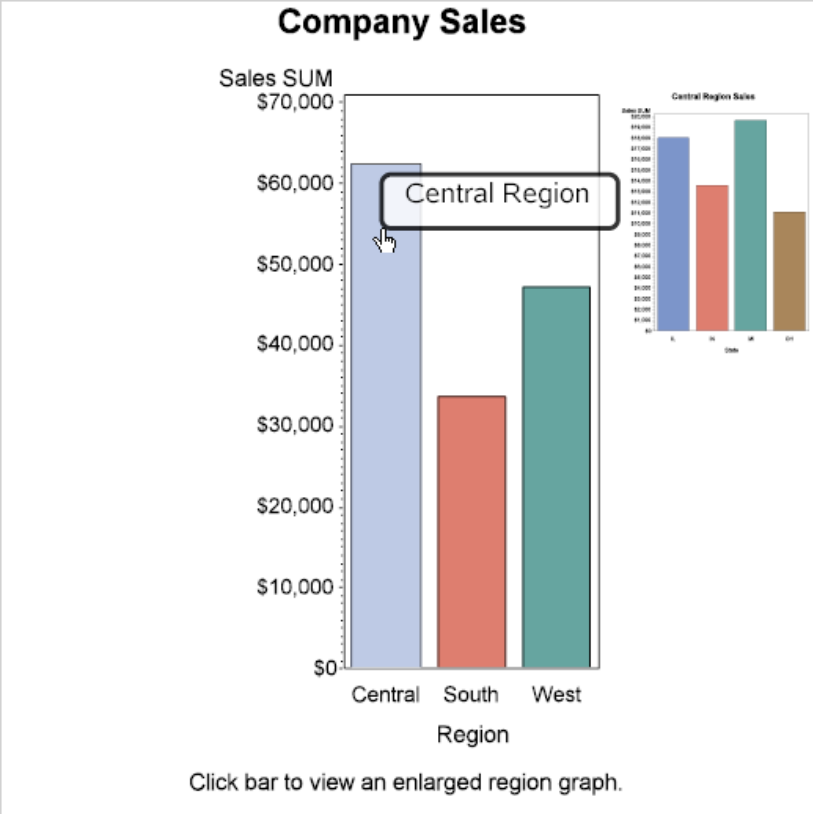
PARAMETERS=
PATH=
STYLE=

OPTIONS statement NOBYLINE option

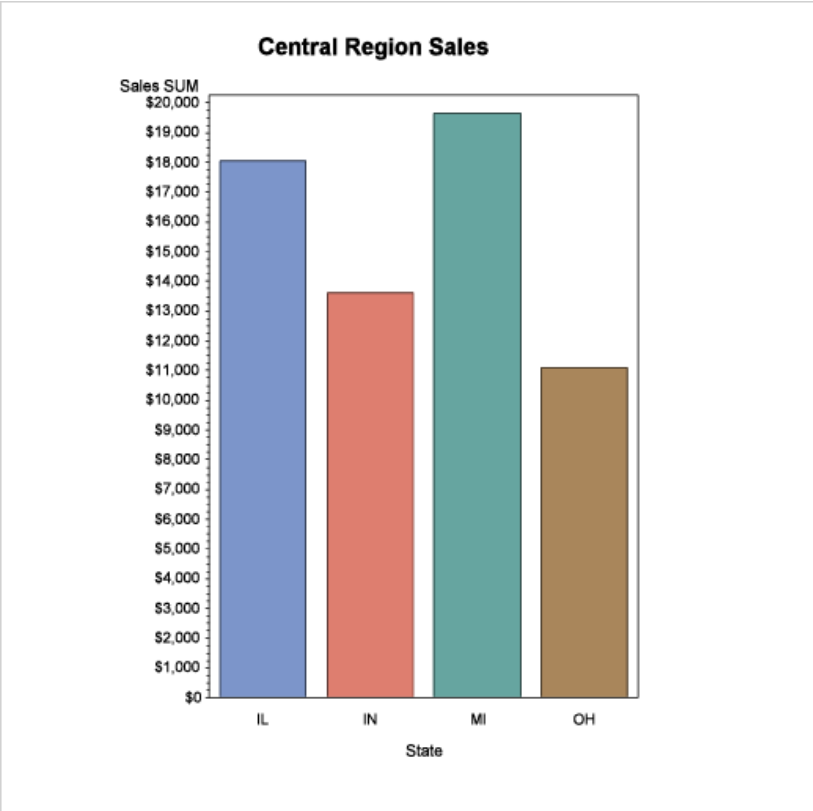
This example uses the SVG graphics device with the ONMOUSEOVER= HTML attribute in the HTML= string to add special effects to a drill-down graph. The drill-down graph is a simple bar chart of regional sales data. Each bar in the main chart is linked to a drill-down graph that displays detailed sales data for that region. The special effects are activated when the mouse pointer is positioned on a bar. The drill-down link for each bar contains the ONMOUSEOVER= HTML attribute. The ONMOUSEOVER= HTML attribute value includes the showImage and changeOpacity functions. The showImage function displays a pop–up preview image of a bar's drill-down chart in the upper right corner of the graph when the mouse pointer is positioned on that bar. The changeOpacity function changes the fill opacity of a bar to 50% when the mouse pointer is positioned on that bar. When a bar is clicked, an enlarged version of the drill-down graph for that bar opens in the browser window.

The main graph is stored in file SALESREPORT.HTML. Here is the graph as it is initially displayed when this file is opened.

For example, when the mouse pointer is positioned on the Central bar, the fill opacity of the bar changes to 50% as shown in the following figure. A pop-up image of the Central Region Sales drill-down chart is displayed in the top right corner of the graphics output area.



If the Central bar is clicked, an enlarged version of the Central Region Sales drill-down chart is opened in the browser as shown in the following figure.



Program

```
filename gout ".";

data regsales(drop=drillurl imgurl);
   length Region State $8 drillurl imgurl $50 htmlstr $256;
   format Sales dollar8.;
   input Region State Sales drillurl imgurl;

   /* Create and add the HTML variable for each observation. */
   htmlstr="href="||quote(trim(drillurl))||
      " title="||quote(trim(Region)||" Region") ||
      " ONMOUSEOVER=ShowImage("||quote(trim(imgurl))||
      ",400,25,200,200); changeOpacity(.5)";

datalines;
Central IL 18038 ./regsales.html#rpt  ./sales.png
Central IN 13611 ./regsales.html#rpt  ./sales.png
Central OH 11084 ./regsales.html#rpt  ./sales.png
Central MI 19660 ./regsales.html#rpt  ./sales.png
South   FL 14541 ./regsales.html#rpt1 ./sales1.png
South   GA 19022 ./regsales.html#rpt1 ./sales1.png
West    CA 13636 ./regsales.html#rpt2 ./sales2.png
West    OR 18988 ./regsales.html#rpt2 ./sales2.png
West    WA 14523 ./regsales.html#rpt2 ./sales2.png
;

goptions reset=all border device=svg xpixels=600 ypixels=450;

ods _all_ close;

ods html body="salesreport.html" path=gout style=listing
    parameters=("drilldownmode"="html");

title1 "Company Sales";
footnote1 j=c "Click bar to view an enlarged region graph.";
proc gchart data=regsales;
   vbar region / sumvar=sales width=8
   patternid=midpoint
   html=htmlstr; /* Set the HTML variable to htmlstr. */
run;
quit;

ods html close;

proc greplay nofs igout=work.gseg;
   delete _all_;
run;
quit;

goptions reset=all border device=pngt;

options nobyline;

ods html body="regsales.html" path=gout anchor="rpt"
   style=listing;

title "#byval(region) Region Sales";
proc gchart data=regsales;
   vbar state / sumvar=sales width=10 name="sales" patternid=midpoint;
   by region;
```

```
run;
quit;

ods html close;
ods html; /* Not required in SAS Studio */
```

Program Description

Here is a detailed description of the SAS program for this example.

Set the file output path. By default, ODS sends its output to the SAS Work directory. You can create a fileref to specify a different location for your ODS output, which is done later in this program.

```
filename gout ".";
```

Create the sales data set. The data set includes the region name, state codes, and the regional sales figures. It also includes the drill-down URL for each of the drill-down graphs (DRILLURL) and the path to the pop-up image for each drill-down graph (IMGURL). Notice that the base anchor RPT is used in the drill-down URLs. This base anchor is specified later in this program. The HTMLSTR variable is added to the data for each observation to store the HTML drill-down string. It is formed from the DRILLURL and IMGURL variables, and includes the TITLE= and ONMOUSEOVER= HTML attributes. The ONMOUSEOVER= HTML attribute value contains the ShowImage and changeOpacity functions to achieve the desired effects. After the HTMLSTR variable is formed, the DRILLURL and IMGURL variables are no longer needed and are dropped from the data set.

```
data regsales(drop=drillurl imgurl);
   length Region State $8 drillurl imgurl $50 htmlstr $256;
   format Sales dollar8.;
   input Region State Sales drillurl imgurl;

   /* Create and add the HTML variable for each observation. */
   htmlstr="href="||quote(trim(drillurl))||
      " title="||quote(trim(Region)||" Region") ||
      " ONMOUSEOVER=ShowImage("||quote(trim(imgurl))||
      ",400,25,200,200); changeOpacity(.5)";

datalines;
Central IL 18038 ./regsales.html#rpt  ./sales.png
Central IN 13611 ./regsales.html#rpt  ./sales.png
Central OH 11084 ./regsales.html#rpt  ./sales.png
Central MI 19660 ./regsales.html#rpt  ./sales.png
South   FL 14541 ./regsales.html#rpt1 ./sales1.png
South   GA 19022 ./regsales.html#rpt1 ./sales1.png
West    CA 13636 ./regsales.html#rpt2 ./sales2.png
West    OR 18988 ./regsales.html#rpt2 ./sales2.png
West    WA 14523 ./regsales.html#rpt2 ./sales2.png
;
```

Set the graphics options for the top-level graph. The SVG graphics device is used for the top-level graph.

```
goptions reset=all border device=svg xpixels=600 ypixels=450;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Specify the ODS HTML settings for the top-level graph. The top-level graph output is sent to file SALESREPORT.HTML in the directory that was defined earlier by fileref GOUT (the current directory in this example). The LISTING style is used for this graph. The drill-down mode is set to HTML.

```
ods html body="salesreport.html" path=gout style=listing
    parameters=("drilldownmode"="html");
```

Generate the top-level drill-down graph.

```
title1 "Company Sales";
footnote1 j=c "Click bar to view an enlarged region graph.";
proc gchart data=regsales;
   vbar region / sumvar=sales width=8
   patternid=midpoint
   html=htmlstr; /* Set the HTML variable to htmlstr. */
run;
quit;
```

Close the ODS HTML destination.

```
ods html close;
```

Clear the GRSEG catalog. Because the image files must be SALES.SVG, SALES1.SVG, and SALES2.SVG, as defined in variable IMGURL in the data set, the existing SALES entries must be cleared from the GRSEG catalog. Otherwise, different file names might be used. In that case, the pop-up images will not be displayed.

```
proc greplay nofs igout=work.gseg;
   delete _all_;
run;
quit;
```

Specify the graphics options for the drill-down graphs. Because the drill-down graph images are overlaid onto the main graph, the PNGT device is used for the drill-down graphs. As a result, the background of the main graph shows through the pop-up graph images.

```
goptions reset=all border device=pngt;
```

Suppress the BY line. BY-group processing is used to generate the drill-down graphs. To suppress the BY line in the drill-down graphs, set the NOBYLINE system option.

```
options nobyline;
```

Specify the ODS HTML settings for the drill-down graphs. The drill-down graphs are stored in file REGSALES.HTML in the directory that is defined by fileref GOUT. The ANCHOR=RPT sets the base anchor to RPT, which is specified in the DRILLURL variable in the data set.

```
ods html body="regsales.html" path=gout anchor="rpt"
   style=listing;
```

Generate the drill-down graphs.

```
title "#byval(region) Region Sales";
proc gchart data=regsales;
   vbar state / sumvar=sales width=10 name="sales" patternid=midpoint;
   by region;
run;
```

```
quit;
```

Close the ODS HTML destination to close the output file, and then reopen ODS HTML (not required in SAS Studio).

```
ods html close;
ods html; /* Not required in SAS Studio */
```

18

Using SAS/GRAPH Output with Microsoft Office Products

What to Consider When Choosing an Output Format

Considerations

When choosing a format for your SAS/GRAPH output to use with Microsoft products, you must consider the following:

- whether you need output in a graphics format or a document format
- the resolution and size of your graphs
- the color depth required for your graphs

- whether you need multiple graphs per page
- whether you need to edit your graphs using Microsoft products or using other third-party software

Graphics Formats versus Document Formats

The SAS/GRAPH software supports output in both graphics format and document format. The graphics format includes graphics information and some text, such as titles, footnotes, and legends.

The document format can include both text and graphics in a single document. These documents store graphics in one of the following ways:

- in the format of the document
- in a graphics format embedded in the document
- in an external file that the document links to

To include images in a document, the images must be compatible with the document. Here is a summary of the compatibility between the SAS/GRAPH document and graphics formats:

Table 18.1 *Document Format and Graphics Format Compatibility*

Document Format	Compatible Graphics Formats
HTML	SVG, PNG, GIF, and ACTIVEX[1]
RTF	EMF, PNG, and ACTIVEX[1]

1 Includes PNG output from the ACTXIMG and JAVAIMG devices.

Image Resolution and Size

Each of the SAS/GRAPH graphics output devices has a default size and resolution setting for the graphics that they generate. For information about the default settings for each device, see “Overview: Using Graphics Devices” on page 90. If you are using a raster format for your graphs, resizing the graph after it is imported into a Microsoft application might degrade the quality of the graph. To preserve the qualify of your raster image, when you create your graph in SAS, set the size to the size that you need in the Microsoft application. Doing so avoids the need to resize after it is imported. See “Setting the Size of Your Graph” on page 129. You can also change to one of the vector formats, which can be resized with no loss of quality.

If you need a high-resolution image, many of the graphics output devices enable you to use the graphics options to change their default resolution. Some of the devices have device variants that you can use to generate high-resolution images. See “Setting the Resolution of Your Graph” on page 130.

Color Depth

Another consideration when choosing a graphics format is color depth. This is the number of bits that are used to represent each color in an image. Color depth can affect the smoothness, clarity, and color trueness of the elements in a rasterized image. A greater color depth means that more distinct colors are available to represent elements such as gradient shading and antialiasing in text.

Most of the graphics file formats support Truecolor, which provides a 24-bit color depth. The GIF format provides only an 8-bit color depth, which can represent up to 256 distinct colors in a single image. For many graphics, 8-bit color depth is sufficient. However, if your output includes background images, color gradients, or other color-intensive elements, consider using a format that supports Truecolor. The formats that support Truecolor include the following:

BMP EPS SVG
EMF PNG

See "Overview: Using Graphics Devices" on page 90 for information about the color depth supported by each of the graphics output devices. See alsoChapter 23, "Using Colors in SAS/GRAPH Programs," on page 363.

Multiple-Image Graphics Files

If you need to store more than one graph in a file, you can use one of the following methods:

- Use the GREPLAY procedure to replay multiple graphs to a file of the same format that was used to generate the original graphs.
- Use the ODS DOCUMENT destination and the DOCUMENT procedure to replay multiple graphs to a file of any supported format
- Use the ODS PRINTER destination with a Universal Printer device that supports multiple-page documents.
- Use the GIF graphics device to insert multiple graphs into an animated GIF image. See Chapter 16, "Generating Animations," on page 227.

If you intend to use animated GIFs, be aware that animated GIFs in Microsoft Word documents, PowerPoint presentations, and Excel worksheets do not play by default. Only the first image in the animation sequence is displayed. To play animated GIFs:

- in a Microsoft Word document, you can save the document as an HTML file, and then open the saved HTML file in a web browser.
- in a Microsoft PowerPoint presentation, you can view the presentation in the Slide Show mode.
- in a Microsoft Excel workbook, you can add ActiveX controls to your workbook forms to play each animated GIF.

For more information, refer to the Microsoft Office documentation.

See "Using the GREPLAY Procedure to Store Multiple Graphs in One Graphics Output File" on page 145 and "Exporting Your Output" on page 152 for information about replaying your graphs. See "Animating GIF Images in SAS/GRAPH" on page 228.

Ability to Edit: Vector versus Raster Formats

You might need the ability to edit your graphs using Microsoft or other third-party software. Make sure that you choose a graphics format that enables you to perform the type of editing that you need to do. For vector formats, such as EMF and SVG, you can edit individual text and graphics elements using graphics editing software. For raster images, some programs such as Microsoft Paint enable you to edit the image. However, in Microsoft Office products, editing is limited to changing only the global attributes of the image, such the size, contrast, brightness, and so on.

Comparison of the Graphics Output

SAS/GRAPH Images That You Can Import into Microsoft Products

The SAS/GRAPH software can generate the following types of graphics output that can be imported into Microsoft products:

EMF	GIF and BMP	ACTIVEX (RTF)
PNG	HTML (PNG)	ACTXIMG (PNG)
TIFF	RTF	JAVAIMG (PNG)

Note the following:

- The ODS HTML destination generates two files: a PNG file (by default) that contains the graph and an HTML file that enables you to view the graph file.
- Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.
- The ODS RTF destination, when used with a device such as PNG or GIF, generates a single RTF file with an embedded image. When used with the ACTIVEX device, the ODS RTF destination generates a single page file with an embedded SAS/GRAPH ActiveX Control. (See Problem Note 65142: Creating an RTF file by using the ACTIVEX device driver results in an RTF file that Microsoft Word cannot open.)
- The ACTIVEX device is not supported in the Microsoft PowerPoint destination. If you specify this device, then SAS/GRAPH substitutes the ACTXIMG device for ACTIVEX.

- The ACTXIMG and JAVAIMG devices generate a PNG file that contains a graph. This PNG image file is identical to the image generated by the ACTIVEX device.
- When you specify the ACTIVEX device in a procedure that does not support the ACTIVEX device, the ACTXIMG device is used instead.

Table 18.2 on page 285 provides a brief comparison of these graphics output formats and lists some of the graphics output devices that generate each output type. For detailed information about all of the graphics output devices, see "Overview: Using Graphics Devices" on page 90.

Table 18.2 *Comparison of the Graphics and Document Types*

Type	Advantages and Limitations	Devices
EMF	Advantages: ■ Most Windows applications recognize the EMF format. ■ Graphs stored in EMF can usually be edited after they are imported. ■ Graphs are imported at full size into Office, and can be resized without a loss of quality. Limitations: ■ The EMF format does not support transparency.See "Working around the EMF Transparency Limitation" on page 287. ■ Only one graph per file is supported.	SASEMF EMF
PNG	Advantages: ■ Designed to display images on the web. ■ Uses lossless data compression. ■ Supports transparency (with the PNGT device). ■ Can store high-resolution images. ■ Supports truecolor images. Limitation: cannot be resized without a loss of quality.	PNG (no transparency) PNG300 (no transparency) PNGT (transparency)
TIFF	Advantages: ■ TIFF can store high-resolution graphics. Limitations: ■ TIFF is not a web graphics format. ■ TIFF images cannot be resized without a loss of quality.	TIFF (RGBA color) TIFFB (monochrome) TIFFK (CMYK color) TIFFP (256 colors)

Type	Advantages and Limitations	Devices
GIF and BMP	Advantages: ■ GIF supports transparent backgrounds. ■ GIF supports alpha-transparency (RGBA), which enables the blending of overlapping colors. ■ GIF supports anti-aliasing, which improves the appearance of text and angled lines. ■ GIF can store multiple images per file when it is formatted as an animated GIF. ■ Both formats support the IBACK option and the IMAGE annotation function for including logos and other images in the background of the graph. Limitations: ■ Both formats have a fixed resolution of 96 DPI. ■ The GIF standard is limited to 256 colors. ■ Both formats lose quality when resized.	BMP GIF
HTML	Advantage: Can store text and graphics. Limitation: The images are not loaded into a PowerPoint or Excel document when the HTML is imported. Only the text and tables are imported.	GIF ACTIVEX [1, 2, 3] ACTXIMG and JAVAIMG, which create PNG files[1] PNG and PNGT
RTF	Advantages: ■ Designed specifically for sharing documents between word processors. ■ Can store both text and graphics.	ACTIVEX [1, 3, 4, 5] ACTXIMG and JAVAIMG, which create PNG files[1] PNG and PNGT SASEMF and EMF

1 Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

2 The default output environment in SAS Studio does not support the ACTIVEX device in the HTML format. To use the ACTIVEX device in HTML output in SAS Studio, you must open your own ODS HTML destination. See "About Using the SAS/GRAPH ACTIVEX Device in SAS Studio" on page 195.

3 See Usage Note 57554: Displaying an ActiveX graph in Microsoft Internet Explorer might result in a BEX error after the application of Microsoft patch KB3000850 on support.sas.com.

4 See Problem Note 65142: Creating an RTF file by using the ACTIVEX device driver results in an RTF file that Microsoft Word cannot open.

5 The SAS/GRAPH ActiveX Control is not compatible with the RTF output format when UTF-8 character encoding is used. See "Generating ActiveX RTF Output in SAS Studio" on page 196.

Working around the EMF Transparency Limitation

To work around the EMF transparency limitation, use the CBACK= or IBACK= graphics options to assign the matching color or image for the graph background. You could instead edit the EMF file after it is imported to remove the default background.

Enhancing Your Graphs

You can use various features in SAS/GRAPH that enable you to enhance your graphs. The following table lists some of these features.

Table 18.3 *Features That Can Enhance Your Graph*

Feature in SAS/GRAPH	Reference
Changing the style of the graphic	Chapter 21, "Using ODS Styles, Device Parameters, and Options," (p. 321)
Adding annotations to the graph	Chapter 28, "Using Annotate Data Sets," (p. 721)
Making the graph interactive	Chapter 15, "Generating Interactive ActiveX Graphics," (p. 191)
Adding drill-down links and data tips to the graph	Chapter 17, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," (p. 249)
Animating the graph	Chapter 16, "Generating Animations," (p. 227)

Importing Your Graphs into Microsoft Office

You can import SAS/GRAPH output that you generated in the HTML or RTF format into Microsoft Office documents such as Microsoft Word, Microsoft Excel, and Microsoft PowerPoint. For information about how to import HTML or RTF files into Microsoft Office documents, refer to the Microsoft web site. You can also use the

ODS destination for PowerPoint to generate SAS/GRAPH output in a Microsoft PowerPoint format. If your SAS/GRAPH output uses the SAS/GRAPH ActiveX Control, ActiveX controls must be enabled in the Microsoft product in order for the import to be successful. Refer to the Microsoft product documentation for information about how to enable ActiveX controls. See Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191 for information about how to generate interactive output using the SAS/GRAPH ActiveX Control.

19

Writing Your Graphs to a PDF File

Overview of Writing Graphs to a PDF File

You can use one of the following methods to write your graph output to a PDF file:

- use the ODS PDF destination to write to a standard PDF file or use the ODS PDF destination with the PRINTER=PDFA ODS option to write to a PDF file that you can archive
- use the ODS PRINTER destination with the PRINTER=PDF ODS option to write to a standard PDF file or with the PRINTER=PDFA ODS option to write to a PDF file that you can archive
- use the ODS LISTING destination with the DEVICE=PDF graphics option to write to a standard PDF file

- use the ODS LISTING destination with the DEVICE=PDFA graphics option to write to a PDF file that you can archive

Using any of these methods, you can do the following:

- create a PDF document that contains one or more pages of graphs with one or more graphs per page.
- use the system fonts or the resident PDF fonts in your document. See "Specifying Fonts" on page 290.
- use system options to control the page layout in your document. See "Changing the Page Layout" on page 292.

The ODS PDF destination and the PDF devices provide additional features and capabilities of their own. To help you determine which method to use for generating your PDF file, Table 19.1 on page 290 lists the additional features and capabilities that the ODS PDF destination and the PDF devices provide.

Table 19.1 *Features and Capabilities Supported by ODS PDF and the PDF Devices*

Method Used	PDF File Types	Document Metadata	Bookmarks	Compression	Drill-Down Links
ODS PDF	PDF Version 1.4	Yes	Yes	Yes	Yes
ODS PDF PRINTER=PDFA	PDF/A-1b	Yes	Yes	Yes	Yes
ODS PRINTER PRINTER=PDF	PDF Version 1.4	Yes	Yes	Yes	Yes
ODS PRINTER PRINTER=PDFA	PDF/A-1b	Yes	Yes	Yes	Yes
ODS LISTING with DEVICE=PDF graphics option	PDF Version 1.4	No	No	Yes[1]	Yes
ODS LISTING with DEVICE=PDFA graphics option	PDF/A-1b	No	No	Yes[1]	Yes

1 The DEFLATION= system option must be specified in the OPTIONS statement.

See also "Creating an Archivable PDF File" on page 291 and "Adding Drill-Down Graphs in Your PDF File" on page 293.

Specifying Fonts

The ODS PDF destination and the PDF devices support the SAS/GRAPH fonts, the TrueType fonts that are installed with the Base SAS product, the TrueType fonts that are installed on the system and are registered in the SAS registry, and the resident

PDF fonts. The resident PDF fonts are the Base 14 fonts that are installed by default with the Adobe Acrobat Reader. These fonts include:

Courier	Helvetica/oblique	Times/bold
Courier/oblique	Helvetica/bold	Times/bold/italic
Courier/bold	Helvetica/bold/oblique	Symbol
Courier/bold/oblique	Times	ITC Zapf Dingbats
Helvetica	Times/italic	

For more information about fonts, see Chapter 22, “Specifying Fonts in SAS/GRAPH Programs,” on page 349.

Creating an Archivable PDF File

You can use the ODS PDF destination, or the ODS LISTING destination with the DEVICE=PDFA graphics option, to write your graph output to an archivable PDF file. By default, the ODS PDF destination writes your output to a PDF Version 1.4 file. To write your graphs to a PDF file that can be archived, add the PRINTER=PDFA option to your ODS statement. The PDFA Universal Printer shortcut device creates a PDF file that is compliant with PDF/A-1b standards and can be archived. See “Using Graphics Devices” on page 89 for information about the PDFA Universal Printer shortcut device.

See “Example: Writing Multiple Graphs to a One-Page, PDF/A-1b-Compliant File” on page 299 for an example of how to create an archivable PDF file using the ODS PDF destination.

You can also use the ODS LISTING destination with the DEVICE=PDFA graphics option to create an archivable PDF file. The following example writes a simple bar chart of electrical power revenue and generation sources to an archivable PDF file named useprev.pdf.

```
/* Create the output file reference */
filename pdfout "useprev.pdf";

/* Specify the PDFA device and the output filename */
goptions reset=all device=pdfa border gsfname=pdfout;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Open the LISTING destination and specify the style */
ods listing style=statistical;

/* Generate the graph */
title  "US Electric Power - Revenue and Generation Sources";

proc gchart data=sashelp.electric (where=(year >= 2000)) ;
   vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;

/* Close ODS LISTING and open ODS HTML */
```

```
ods listing close;
ods html; /* Not required in SAS Studio */
```

Changing the Page Layout

The following system options enable you to change the page layout for your PDF document when you use the ODS PDF destination or one of the PDF devices:

- ORIENTATION=PORTRAIT | LANDSCAPE | REVERSEPORTRAIT | REVERSELANDSCAPE
- PAPERSIZE="*paper-size*"
- LEFTMARGIN=*value*
- RIGHTMARGIN= *value*
- TOPMARGIN= *value*
- BOTTOMMARGIN=*value*

The PAPERSIZE= and margin options must be set before the ODS PDF statement is executed. These settings apply to the entire file and cannot be changed as the output is being generated. In contrast, the ORIENTATION= system option can be used to set the initial page orientation and to change the page orientation at any point as the output is being generated. See *SAS System Options: Reference* for information about these system options. See "Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata" on page 295 for an example of how to use these system options to change the page layout of a PDF file.

Adding Metadata to Your PDF File

The following ODS PDF options enable you to add document metadata to your PDF file:

- AUTHOR="*author-name*"
- KEYWORDS="*word1 word2 ...* "
- SUBJECT="*document-subject*"
- TITLE="*document-title*"

See "Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata" on page 295 for an example of how to add metadata to a PDF file.

Adding Bookmarks for Your Graphs

The ODS PROCLABEL=*label* statement enables you to add bookmarks for your graphs when you use the ODS PDF destination. The PROCLABEL= ODS option specifies the name of the top-level bookmark. The description for each procedure that you run after your ODS PROCLABEL= statement is added as a subtopic under the top-level bookmark that the PROCLABEL= option defines. You can use the DESCRIPTION= option to set the text of the subtopic bookmark for each graph procedure. If you do not specify a description, the default graph description is used. See "Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata" on page 295 for an example.

Adding Drill-Down Graphs in Your PDF File

The SAS/GRAPH PDF device enables you to add drill-down graphs to your PDF file when you generate your PDF file using the ODS PDF or ODS PRINTER destination. The drill-down links include links that are created with the URL=, HTML=, and HTML_LEGEND= options for graph statements and annotation data, and the LINK= option for TITLE, FOOTNOTE, and NOTE statements. The links can be to an external file or URL. Creating drill-down graphs for a PDF document is similar to creating drill-down graphs for a web presentation. See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255.

Note: In the PDF format, when the mouse pointer is positioned on a drill-down link, a data tip displays the destination URL for that link. Custom data tips are not supported in the PDF format.

In order to write a drill-down graph to a PDF file, you must do all of the following:

- use one of the following ODS statements to open the ODS PDF or ODS PRINTER destination:

  ```
  ODS PDF;
  ```

  ```
  ODS PRINTER PRINTER=PDF;
  ```

  If you use the ODS PRINTER destination, you must specify the PDF printer. For more information about the ODS PDF and ODS PRINTER destinations, see *SAS Output Delivery System: User's Guide*.

Note: Because the drill-down graphs are external to the PDF file, do not use the PDFA printer to generate an archivable PDF file. An archivable PDF file must be a totally self-contained file.

- use one or more of the following options to specify the URL for your drill-down links:
  - the URL= or HTML= option in your drill-down chart statement
  - the HTML_LEGEND= option in your drill-down chart statement
  - the HTML= option in your annotate data
  - the LINK= option on your TITLE, NOTE, or FOOTNOTE statement

If all of these conditions are not met, your graph is written to the PDF file, but the drill-down links do not work as expected.

See the following examples:

- "Example: Writing a Drill-Down Graph to a PDF File" on page 300
- "Example: Writing Multiple Drill-Down Graphs to a PDF File Using BY-Group Processing" on page 304
- "Example: Writing Multiple Graphs to a PDF File Using the GREPLAY Procedure" on page 307

Changing the Default Compression Level for Your PDF File

The COMPRESS= ODS PDF option enables you to change the default compression level for your PDF file. The COMPRESS= option can be set to an integer value between 0 and 9, which specifies the level of compression. A value of 0 means no compression. The default level is 6. For information about the ODS PDF statement COMPRESS= option, see "ODS HTML Statement" in *SAS Output Delivery System: User's Guide*.

You can also use the DEFLATION= system option to compress your PDF files. Like the COMPRESS= ODS PDF option, the COMPRESS= system option can be set to an integer value between 0 and 9, where 0 indicates no compression and 9 indicates maximum compression. For information about the DEFLATION= system option, see "DEFLATION= System Option" in *SAS System Options: Reference*.

Examples: Writing Graphs to a PDF File

About These Examples

The examples in this section show you how to exports graphs to a PDF file. The examples include:

Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata

Here is an example that writes a multipage PDF document to file EuropeanCars.pdf. The document contains two sets of graphs that show information about European cars and car makers. The first set of graphs shows the number of models that each car maker produces that fall within the following ranges of fuel efficiency in highway miles per gallon (MPG):

- 19 MPG or less
- between 20 and 29 MPG (inclusive)
- 30 MPG or higher

The second set of graphs shows the total number of models of the following vehicle types: sedans, SUVs, wagons, and sports cars. RUN-GROUP processing is used to generate all of the graphs. Each page displays a single bar chart and is set up for A4 paper with a 1 cm right, left, and bottom margin, and a 2 cm top margin. For the first set of graphs, the page orientation for each graph is set as shown in the following table.

Graph	Page Orientation
19 MPG or less	Portrait

Graph	Page Orientation
between 20 and 29 MPG	Landscape
30 MPG or higher	Portrait

For the second set, the page orientation for each graph is set as shown in the following table:

Graph(s)	Page Orientation
Sedans	Landscape
SUVs, wagons, and sports cars	Portrait

The document also contains bookmarks and metadata. The PROCLABEL= ODS option is used to set the text of the top-level bookmark heading for each graph set. The DESCRIPTION= option is used with each procedure to set the text of each subheading bookmark. The document metadata includes the following:

- the document's title: European Cars by MPG and Type
- the document's author: J. L. Cho
- the subject of the document: European Cars
- a list of search keywords: automobiles, European, cars, MPG, sedans, wagons, and SUVs

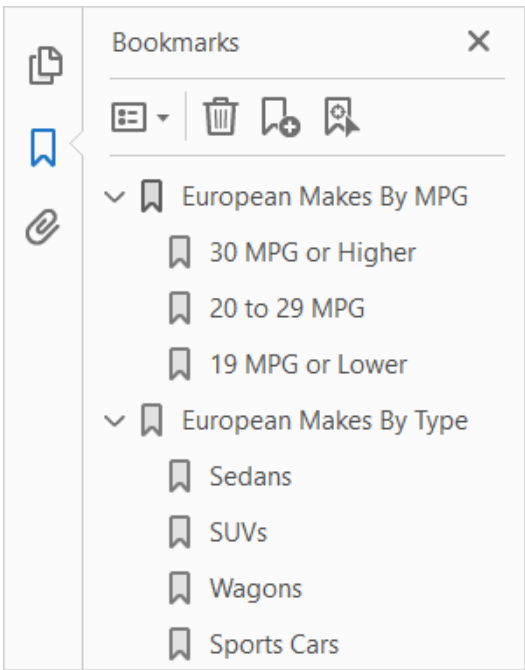
Here is the SAS program for this example.

```
/* Close the currently open ODS destinations */
ods _all_  close;

/* Reset the graphics options  */
goptions reset=all;

/* Modify the PDF page properties. These options must
   be set before the ODS PDF statement is executed. */
options
   papersize=A4
   leftmargin=1cm
   rightmargin=1cm
   bottommargin=1cm
   topmargin=2cm;

/* Open the PDF destination */
ods pdf style=seaside
   file="EuropeanCars.pdf"  /* Output filename */
   compress=0          /* No compression */
   /* Add metadata */
   author="J. L. Cho"
   subject="European Cars"
   title="European Cars by MPG and Type"
   keywords="automobiles European cars MPG sedans wagons SUVs";
```

```
/* Create a data set of European cars from SASHELP.CARS */
proc sort data=sashelp.cars(where=(origin="Europe")) out=europeancars;
   by make type mpg_highway;
run;

/* Create the axes */
axis1 label=(angle=90 "Number of Models Rated at Specified MPG
Highway");
axis2 label=(angle=90 "Number of Models of Specified Type");
run;

/* Set the top-level bookmark for the first set of graphs */
ods proclabel="European Makes By MPG";

/* Create the first set of graphs */
proc gchart data=europeancars;
   vbar Make / name="HighMPG"
      raxis=axis1 autoref clipref cref=lightgray lautoref=33
      description="30 MPG or Higher"; /* Set subheading text */
      title1 "30 MPG or Higher";
      where MPG_Highway >= 30;
   run;
   options orientation=LANDSCAPE; /* Display LANDSCAPE */
   vbar Make / name="MedMPG"
      raxis=axis1 autoref clipref cref=lightgray lautoref=33
      description="20 to 29 MPG"; /* Set subheading text */
      title1 "20 to 29 MPG";
      where MPG_Highway < 30 and MPG_Highway >= 20;
   run;
   options orientation=PORTRAIT; /* Display PORTRAIT */
   vbar Make / name="LowMPG"
      raxis=axis1 autoref clipref cref=lightgray lautoref=33
      description="19 MPG or Lower"; /* Set subheading text */
      title1 "19 MPG or Lower";
      where MPG_Highway < 20;
   run;
quit;

/* Set the top-level bookmark for the second set of graphs */
ods proclabel="European Makes By Type";

/* Create the second set of graphs */
proc gchart data=europeancars;
   options orientation=LANDSCAPE; /* Display LANDSCAPE */
   vbar Make / name="Sedans"
      raxis=axis2 autoref clipref cref=lightgray lautoref=33
      description="Sedans"; /* Set subheading text */
      title1 "Sedans";
      where Type = "Sedan";
   run;
   options orientation=PORTRAIT; /* Display PORTRAIT */
   vbar Make / name="SUVs"
      raxis=axis2 autoref clipref cref=lightgray lautoref=33
      description="SUVs"; /* Set subheading text */
      title1 "SUVs";
      where Type="SUV";
```

```
      run;
      vbar Make / name="Wagons"
         raxis=axis2 autoref clipref cref=lightgray lautoref=33
         description="Wagons"; /* Set subheading text */
         title1 "Wagons";
         where type="Wagon";
      run;
      vbar Make / name="Sports"
         raxis=axis2 autoref clipref cref=lightgray lautoref=33
         description="Sports Cars"; /* Set subheading text */
         title1 "Sports Cars";
         where type="Sports";
      run;
   quit;

   /* Close the PDF destination, and then open the HTML destination
      (not required in SAS Studio) */
   ods pdf close;
   ods html;  /* Not required in SAS Studio */

   /* Reset options, titles, and footnotes */
   goptions reset=all;
   title;
   footnote;
```

The bookmarks in the PDF document are shown in the following figure:



The document metadata is displayed on the **Description** tab of the Document Properties dialog box. To open the Document Properties dialog box, type CTRL-D anywhere in the PDF viewer window or right-click in the PDF viewer window, and then select **Document Properties** from the pop-up menu. The following figure shows the document metadata that is displayed for this example.

Example: Writing Multiple Graphs to a One-Page, PDF/A-1b-Compliant File

Here is an example that creates the PDF/A-1b-compliant file FourVbars.pdf, which contains four vertical bar charts on one page. The graphs show the average sales of sofas, chairs, desks, and beds in Canada by province. The PRINTER=PDFA ODS option is used to create a PDF file that is compliant with PDF/A-1b standards. To create a standard Version 1.4 PDF file, remove the PRINTER=PDFA option from the ODS statement. The STARTPAGE=NEVER ODS option is used to prevent a page break from being inserted between each graph. The HORIGIN= and VORIGIN= graphics options are used to position each graph on the page. Here is the SAS code for this example.

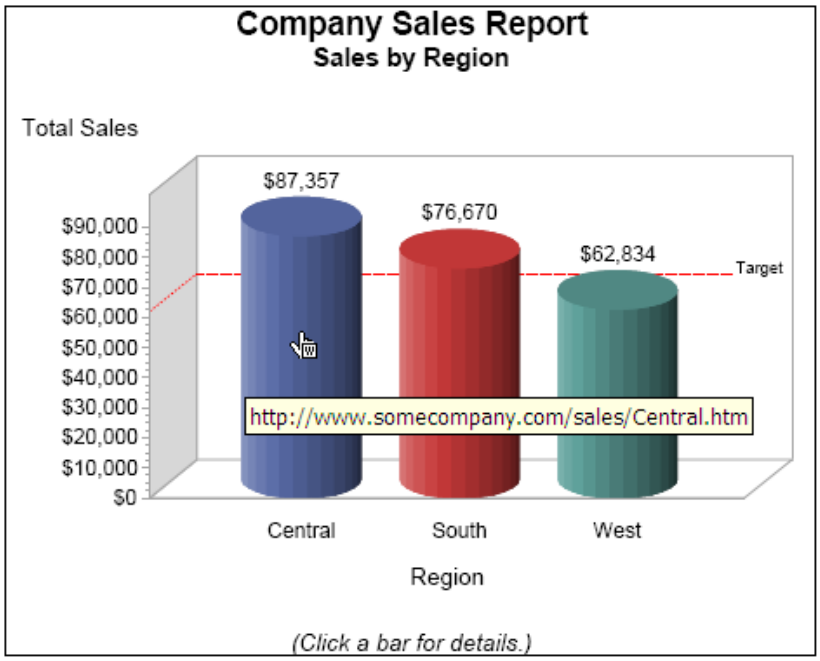
```
/* Close the currently open ODS destinations */
ods _all_ close;

/* Set page options */
options orientation=portrait rightmargin=0.1in leftmargin=0.1in;
goptions reset=all ftext="Helvetica/bold";

/* Create the axes */
axis1 label=(angle=90 "Average Sales"); /* Left-side graph Y axis */
axis2 label=none;         /* Right-side graph Y axis */
axis3 label=("Province"); /* Midpoint axis */

/* Open PDF */
ods pdf style=printer
    notoc
    printer=pdfa         /* Create an archivable PDF */
    file="FourVbars.pdf" /* Output filename */
    startpage=never;     /* Do not insert a pagebreak after each graph
*/

/* Size each graph 4in x 4in */
goptions hsize=4in vsize=4in;

/* Generate the graphs */
```

```
proc gchart data=sashelp.prdsal3;
   /* Format the sales values */
   format Actual comma5.0;

   /* Create the Sofas graph in the top-left quadrant */
   title1 "Average Sales of Sofas";
   goptions horigin=0in vorigin=5in;
   vbar State / sumvar=Actual type=mean
      autoref clipref raxis=axis1 maxis=axis3;
      where product="SOFA" and country="Canada";
   run;

   /* Create the Chairs graph in the top-right quadrant */
   title1 "Average Sales of Chairs";
   goptions horigin=4.2in vorigin=5in;
   vbar State / sumvar=Actual type=mean
      autoref clipref raxis=axis2 maxis=axis3;
      where product="CHAIR" and country="Canada";
   run;

   /* Create the Desks graph in the bottom-left quadrant */
   title1 "Average Sales of Desks";
   goptions horigin=0in vorigin=0in;
   vbar State / sumvar=Actual type=mean
      autoref clipref raxis=axis1 maxis=axis3;
      where product="DESK" and country="Canada";
   run;

   /* Create the Beds graph in the bottom-right quadrant */
   title1 "Average Sales of Beds";
   goptions horigin=4.2in vorigin=0in;
   vbar State / sumvar=Actual type=mean
      autoref clipref raxis=axis2 maxis=axis3;
      where product="BED" and country="Canada";
   run;
quit;

/* Close PDF, and then reopen ODS HTML (not required
   in SAS Studio) */
ods pdf close;
ods html; /* Not required in SAS Studio */

/* Reset graphics options and titles */
goptions reset=all;
title;
```

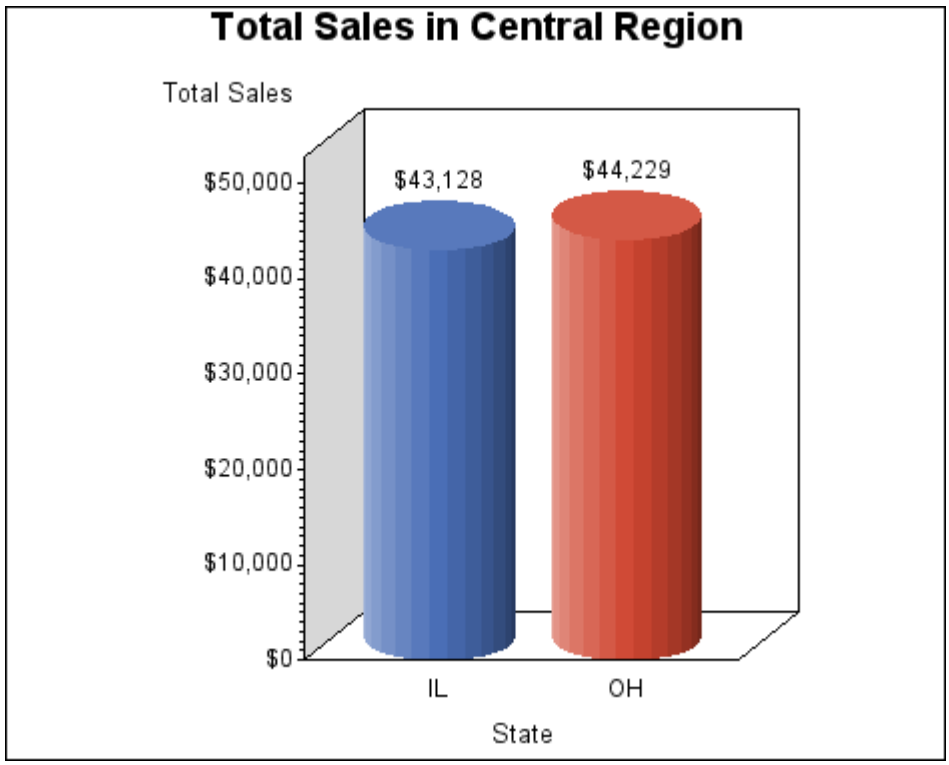
Example: Writing a Drill-Down Graph to a PDF File

Here is an example that exports a drill-down graph to a PDF file. This example writes a top-level drill-down graph to a PDF file, and then writes the drill-down target graphs to HTML files. The top-level drill-down graph in the PDF file shows the total sales in three regions in which a company operates. A reference line indicates a sales target of $62,000 for each region. Each region bar in the graph is linked to a

graph that breaks down the sales data for that region. When the mouse pointer is positioned on a bar, a data tip shows the URL to the drill-down graph. An example is shown in the following figure.



The drill-down graphs consist of one HTML file for each region: Central.htm, South.htm, and West.htm. Each HTML file contains a graph that breaks down the total sales data by state for that region. When you click a bar in the top-level graph, the HTML file to which that bar is linked appears in your web browser. In this example, if you click the Central bar, the Central.htm file is opened, which displays the following graph in your web browser.



Here is the SAS code for this example.

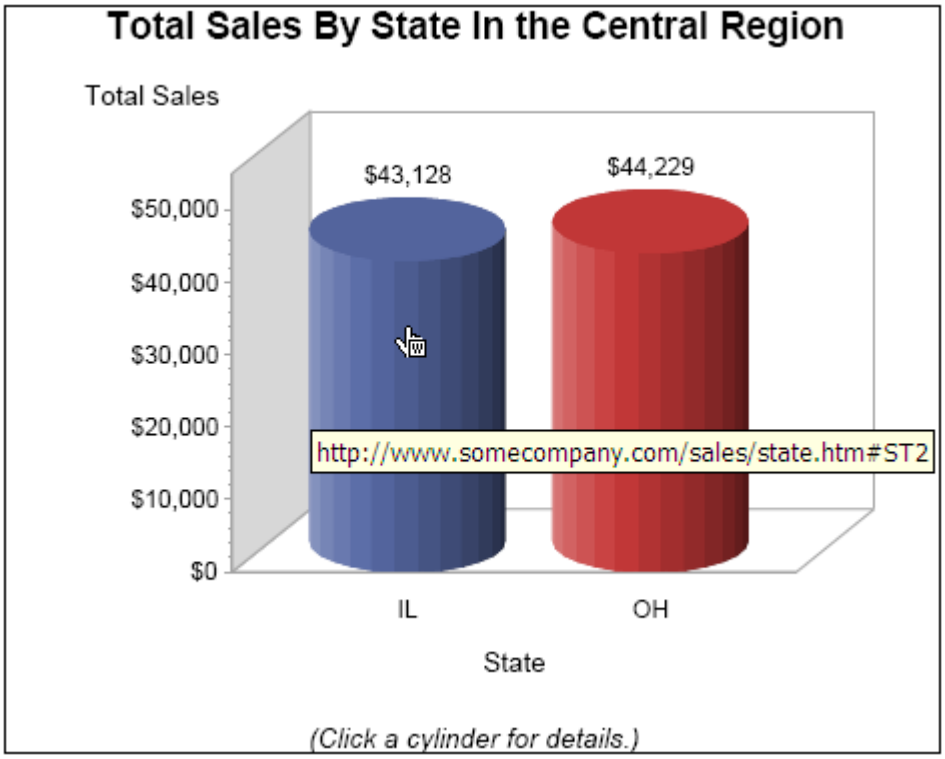
```
/* Define the ODS output path. */
%let outpath=.;
filename odsout "&outpath";

/* Define the PDF output filename. */
filename pdfout "&outpath\salesrpt_single_page.pdf";

/* Define the base URL for the links. */
%let baseurl=.;

 /* Create the data set REGSALES. */
data regsales;
   length Region State $ 8 Location $ 15;
   format Sales dollar8.;
   input State Region Location Sales;
datalines;
IL Central EVANSTON   18038
IL Central CHICAGO    14322
IL Central AURORA     10768
OH Central COLUMBUS   13611
OH Central DAYTON     11084
OH Central CINCINNATI 19534
FL South   MIAMI      14541
FL South   TAMPA      16733
NC South   RALEIGH    19022
NC South   WILMINGTON 12876
NC South   CHARLOTTE  13498
CA West    SANTA-CRUZ 13636
CA West    LONG-BEACH 15687
WA West    SEATTLE    18988
WA West    TACOMA     14523
;

/* Add the link information to the data. */
data regsales;
   set regsales;
   length RPTR $ 80;
   if (Region="Central") then RPTR="&baseurl./Central.htm";
   else if (Region="South") then RPTR="&baseurl./South.htm";
   else if (Region="West") then RPTR="&baseurl./West.htm";
   else RPTR=.;
run;

/* Close the currently open ODS destinations, and then set the
   graphics options. */
ods _all_ close;
goptions reset=all device=png border xpixels=520 ypixels=450;

/* Clear titles and footnotes */
title;
footnote;

/* Create a macro to use for generating the region charts. */
%macro do_region(region);
   /* Open ODS HTML destination. Use the region name as the filename */
```

```
   ods html file="&region..htm" path=odsout style=listing;

   /* Set the axis label and title */
   axis1 label=("Total Sales");
   title1 "Total Sales in &region Region";

   /* Create the chart */
   proc gchart data=regsales;
      vbar3d state / sumvar=sales outside=sum name="region"
         raxis=axis1 patternid=midpoint
         description="Total sales for &region region"
         shape=cylinder width=15;
      where region="&region";
   run; quit;

   /* Close the ODS HTML destination */
   ods html close;
%mend do_region;

/* Call the %DO_REGION macro for Central, South, and West. */
%do_region(Central);
%do_region(South);
%do_region(West);

/* Open the PDF destination and set the style. */
ods pdf file=pdfout style=statistical;

/* Set the graphics options. You must use the PDF or PDFA device. */
goptions reset=all gsfname=outp device=pdf border
   vorigin=2.5in horigin=0.6in vsize=550pt hsize=520pt;

/* Generate the Company Sales report PDF document. */
title1 "Company Sales Report";
title2 "Sales by Region";
footnote1 "(Click a bar for details.)";
axis1 label=("Total Sales") reflabel=(j=r "Target  ");
proc gchart data=regsales;
      vbar3d region /
      cref=red lref=3 ref=62000 /* Draw a reference line at the */
                                /* sales target. */
      sumvar=sales outside=sum raxis=axis1
      shape=cylinder width=15 patternid=midpoint
      description="Company sales report"
      url=RPTR;  /* Specify RPT as the URL variable in the data. */
run; quit;

/* Close the PDF destination, and then open HTML (not required
   in SAS Studio)*/
ods pdf close;
ods html; /* Not required in SAS Studio */

/* Reset graphics options, titles, and footnotes */
goptions reset=all;
title;
footnote;
```

In the SAS program, notice that column RPTR is added to the data. For each observation, RTPR is set to the URL of the appropriate report based on the value of column REGION. Because column RPTR contains a URL string only, it is specified in the URL= option on the drill-down chart GCHART statement. The HTML= option cannot be used in this case.

Example: Writing Multiple Drill-Down Graphs to a PDF File Using BY-Group Processing

Here is an example that creates a PDF file that contains three drill-down graphs of sales data. This example uses the ODS LISTING destination with the PDF device and BY-group processing to write the top-level graphs to a PDF file. It uses the ODS HTML destination with BY-group processing to generate the drill-down graphs. In the PDF file, each graph shows the total sales data by state in each of three regions: Central, South, and West. In each of the graphs, each bar is linked to a drill-down graph that breaks down the total sales data for that state by store location. A data tip displays the URL to the drill-down graph as shown in the following figure.

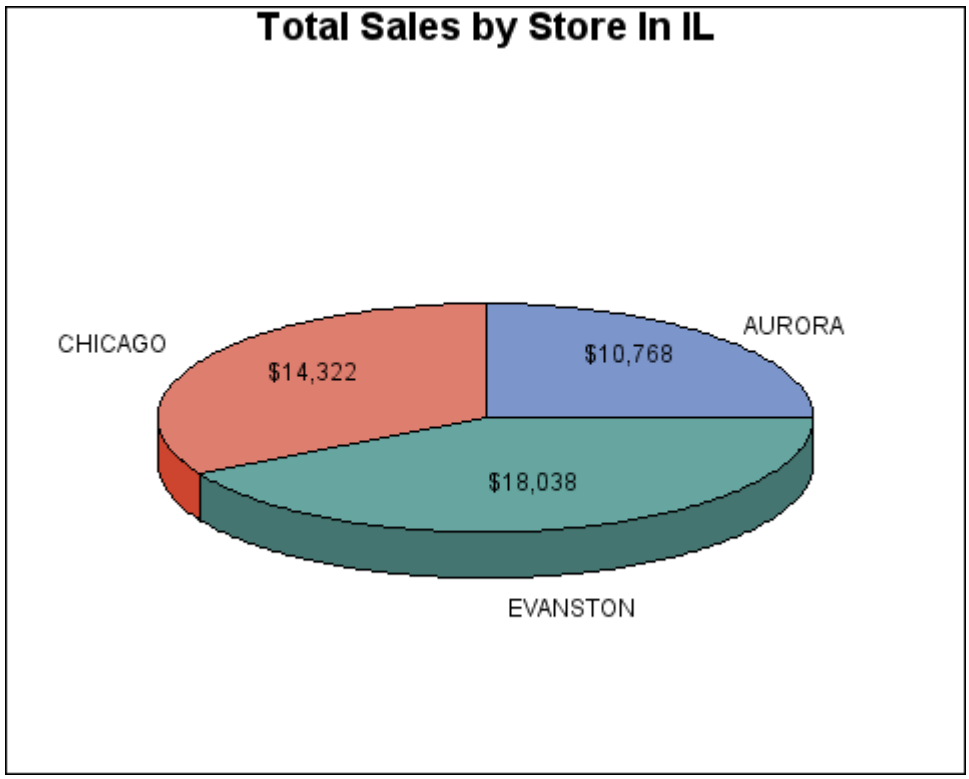


The drill-down graphs consist of one graph for each of the following states: IL, OH, FL, NC, CA, and WA. Each drill-down graph shows the total sales data for a state by store location. All six of the drill-down graphs are written to the HTML file state.htm. In the state.htm file, a named anchor is placed at the top of each graph. The ANCHOR= option is used in the ODS HTML statement to set the base anchor name to ST. When BY-group processing occurs, the anchor names are assigned to each STATE value as they occur in the data. The first name assigned is the base name, which is ST in this example. For the subsequent names, a number is appended to

the base name to create a unique name for each STATE value. Because the data is sorted by STATE, the anchor names are assigned as shown in the following table.

STATE Value	Anchor Name
CA	ST
FL	ST1
IL	ST2
NC	ST3
OH	ST4
WA	ST5

The named anchors are included in the URLs that link the bars in the top-level graph to the drill-down graph in the state.htm file. The data tip shown in the previous figure provides an example. Clicking the IL bar in this example opens the state.htm file and positions the cursor at the IL graph anchor (ST2) as shown in the following figure.



Here is the SAS code for this example.

```
/* Define the ODS output path. */
%let outpath=.;
filename odsout "&outpath";
```

```
/* Define the PDF output filename. */
filename pdfout "&outpath\salesrpt_single_page.pdf";

/* Define the base URL for the links. */
%let baseurl=./state.htm;

 /* Create the data set REGSALES. */
data sales;
   length Region State $ 8 Location $ 15;
   format Sales dollar8.;
   input State Region Location Sales;
datalines;
CA West    SANTA-CRUZ 13636
CA West    LONG-BEACH 15687
FL South   MIAMI      14541
FL South   TAMPA      16733
IL Central EVANSTON   18038
IL Central CHICAGO    14322
IL Central AURORA     10768
NC South   RALEIGH    19022
NC South   WILMINGTON 12876
NC South   CHARLOTTE  13498
OH Central COLUMBUS   13611
OH Central DAYTON     11084
OH Central CINCINNATI 19534
WA West    SEATTLE    18988
WA West    TACOMA     14523
;
run;

/* Add the link information to the data */
data sales;
   set sales;
   length RPTS $ 80;
   if (State="CA") then RPTS="&baseurl.#ST";
   else if (State="FL") then RPTS="&baseurl.#ST1";
   else if (State="IL") then RPTS="&baseurl.#ST2";
   else if (State="NC") then RPTS="&baseurl.#ST3";
   else if (State="OH") then RPTS="&baseurl.#ST4";
   else if (State="WA") then RPTS="&baseurl.#ST5";
   else RPTS=.;
run;

/* Close the currently open ODS destinations, and then set the
   graphics options. */
ods _all_ close;
goptions reset=all device=png border hsize=5in vsize=4in;

/* Generate the state sales charts. Use ST as the base HTML anchor
   and state.htm as the output filename. The drill-down chart links
   to this file. */
ods html5 file="state.htm" path=odsout anchor="ST" style=listing
   options(bitmap_mode="inline");
title1 "Total Sales by Store In #byval(State)";
title2;
proc gchart data=sales;
```

```
   pie3d Location / sumvar=Sales
      noheader
      slice=outside value=inside
      description="Total Sales by Store In #byval(State)";
   by State;
run; quit;
ods html5 close; /* Close ODS HTML. */

/* Sort the data by region. */
proc sort data=sales out=regsort;
   by Region;
run; quit;

/* Set the graphics options. You must use the PDF or PDFA device. */
goptions reset=all gsfname=outp device=pdf border
   vorigin=2.5in horigin=0.5in hsize=450pt vsize=380pt;

/* Open the PDF destination and specify the STATISTICAL style */
ods pdf file=pdfout style=statistical;

/* Generate the drill-down chart. */
axis1 label=("Total Sales");
title1 "Total Sales By State In the #byval(Region) Region";
footnote "(Click a bar for details.)";
proc gchart data=regsort;
   vbar3d state / sumvar=sales outside=sum name="region"
      raxis=axis1 patternid=midpoint
      shape=cylinder width=15
      url=RPTS; /* Specify RTPS as the URL variable. */
   by Region;
run; quit;

/* Close the PDF destination, and then open ODS HTML (not required
   in SAS Studio) */
ods pdf close;
ods html; /* Not required in SAS Studio */

/* Reset graphics options, titles, and footnotes */
goptions reset=all;
title;
footnote;
```

Example: Writing Multiple Graphs to a PDF File Using the GREPLAY Procedure

Here is an example that creates a sales report in both the HTML and PDF format. It first generates the sales report in the HTML format. It then uses the GREPLAY procedure to replay the graphs to a PDF file. The second graph in the report is a drill-down graph in both the HTML report and the PDF report.

```
/* Define the ODS output path. */
%let outpath=.;
filename odsout "&outpath";
```

```
/* Define the output filename. */
%let outfilename=salesrpt_single_page;
filename pdfout "&outpath\&outfilename..pdf";

/* Define the base URL for the links. */
%let baseurl=./state.htm;

 /* Create the data set REGSALES. */
data sales;
   length Region State $ 20 Location $ 20;
   format Sales dollar8.;
   input State Region Location Sales;
datalines;
CA West    SANTA-CRUZ 13636
CA West    LONG-BEACH 15687
FL South   MIAMI      14541
FL South   TAMPA      16733
IL Central EVANSTON   18038
IL Central CHICAGO    14322
IL Central AURORA     10768
NC South   RALEIGH    19022
NC South   WILMINGTON 12876
NC South   CHARLOTTE  13498
OH Central COLUMBUS   13611
OH Central DAYTON     11084
OH Central CINCINNATI 19534
WA West    SEATTLE    18988
WA West    TACOMA     14523
;

/* Add the drill-down link information to the data. */
data sales;
   set sales;
   length RPTS $ 80;
   select(State);
     when("CA") RPTS="&baseurl.#ST";
     when("FL") RPTS="&baseurl.#ST1";
     when("IL") RPTS="&baseurl.#ST2";
     when("NC") RPTS="&baseurl.#ST3";
     when("OH") RPTS="&baseurl.#ST4";
     when("WA") RPTS="&baseurl.#ST5";
     otherwise RPTS=.;
   end;
run;

/* Close the currently open ODS destinations, and then set
   the graphics options. */
ods _all_ close;
goptions reset=all device=png border xpixels=450 ypixels=450;

/* Generate the state sales charts. Use ST as the base HTML anchor, and
   state.htm as the output filename. The drill-down graph will link to
   this file. */
ods html5 path=odsout file="state.htm" anchor="ST" style=listing
   options(bitmap_mode="inline");
```

```
title1 "Total Sales by Store Location";
title2;
proc gchart data=sales;
   pie3d Location / sumvar=Sales noheader;
   by State;
run; quit;
ods html5 close; /* Close ODS HTML5. */

/* Delete the old chart GRSEGs. */
proc greplay igout=work.gseg nofs;
   delete _all_;
run; quit;

/* Set the graph size. */
goptions reset=all xpixels=550 ypixels=550 gunit=pt;

/* Open the HTML5 destination to generate the HTML report. */
ods html5 path=odsout file="&outfilename..html" style=statistical
   options(bitmap_mode="inline");

/* Generate the region, state, and local charts. Use Region, State,
   and Location as the GRSEG names. */
axis1 label=("Total Sales");
proc gchart data=sales;
   /* Generate the Regional sales chart. */
   title1 "Total Sales by Region";
   vbar3d region / sumvar=sales outside=sum name="Region"
      raxis=axis1 patternid=midpoint shape=cylinder width=15
      description="Total Sales by Region";
run;
   /* Generate the drill-down State sales chart. */
   title1 "Total Sales by State";
   footnote "(Click a bar for details.)";
   hbar3d state / sumvar=sales sum sumlabel="Total Sales" name="State"
      patternid=midpoint raxis=axis1 shape=cylinder width=5
      description="Total Sales by State"
      url=RPTS; /* Specify the URL variable in the data. */
run;
   /* Generate the Location sales chart. */
   title1 "Total Sales by Location";
   footnote; /* Clear the footnote */
   pie location / name="Location" sumvar=sales noheading
      plabel=(height=9pt)
      description="Total Sales by Location";
run; quit;

/* Close the HTML5 destination. */
ods html5 close;

/* Set the graphics options for write to a PDF file. */
goptions device=pdf noborder;

/* Suppress the creation date and time in the PDF output. */
option nodate;

/* Create a replay template that centers that graph on the page
```

```
   and draws a navy border around each graph. */
proc greplay tc=work.tempcat nofs;
tdef newtemp
     1/llx=10 lly=20
       ulx=10 uly=80
       urx=90 ury=80
       lrx=90 lry=20
       color=navy;
     template newtemp;
quit;

/* Set the report title and footnote. */
title1 "Sales Report";
footnote1 "Sales data is based on first-quarter results.";

/* Open the ODS PDF destination to generate the PDF report.
   Specify NOGTITLE and NOGFOOTNOTE so that the report title
   and footnote appear in the printer output. */
ods pdf file=pdfout nogtitle nogfootnote;

/* Replay the three sales charts to the PDF file. */
proc greplay tc=work.tempcat igout=work.gseg nofs
   template=newtemp;
   treplay 1:Region des="Total Sales by Region";
   treplay 1:State des="Total Sales by State";
   treplay 1:Location des="Total Sales by Location";
run; quit;

/* Close ODS PDF, and then open ODS HTML (not required
   in SAS Studio. */
ods pdf close;
ods html; /* Not required in SAS Studio */

/* Reset the graphics options. */
goptions reset=all;
```

In order to add a title and footnote to the PDF report, TITLE1 and FOOTNOTE1 statements are used to define the title and footnote text. The NOGTITLE and NOGFOOTNOTE options are then specified in the ODS PDF statement, which add the title and footnote to the PDF output. For more information about the ODS PDF statement NOGTITLE and NOGFOOTNOTE options, see *SAS Output Delivery System: User's Guide*.

20

Troubleshooting Web Output

Overview

This chapter contains information that you can use to resolve rendering problems on client workstations. Included is a table listing solutions to common problems as well as various tips ranging from resolving web server access issues to modifying HTML character entities.

Solutions to Common Rendering Problems

If you or a member of your audience cannot display your presentation, then refer to the following table for solutions.

Note: To view your web presentation, your audience must view the presentation through one of the supported browsers. For a list of supported browsers, refer to the SAS web site Install Center. Select the System Requirements link for the appropriate operating system environment and search for the section about viewing HTML pages created for Java and ActiveX.

Note: For information about software required to open HTML output that is generated by the ACTIVEX device, see Chapter 15, "Generating Interactive ActiveX Graphics," on page 191.

Table 20.1 *Web Troubleshooting*

Symptom	Cause	Remedy
Cannot access the HTML file.	Incorrect URL.	Check the URL in the browser.
	Network access denied.	Check operating environment permissions for the HTML file.
		Check firewall access permissions for Internet clients.
Browser cannot display the file.	Browser might not meet requirements.	Check the requirements. To view your web presentation, your audience must view the presentation through one of the supported browsers. For a list of supported browsers, refer to the SAS web site Install Center. Select the System Requirements link for the appropriate operating system environment and search for the section about viewing HTML pages created for ActiveX. For information about software required to open HTML output that is generated by the ACTIVEX device, see "Installing the SAS/GRAPH ActiveX Control" on page 197.

Symptom	Cause	Remedy
	ActiveX control might not have been installed or might be out of date.	Install the ActiveX control manually. See "Installing the SAS/GRAPH ActiveX Control" on page 197. Consider updating the presentation to prompt users to install the control. See "Configuring Your ActiveX Presentation to Install the SAS/GRAPH ActiveX Control from Your Local Network" on page 199.
	User attempting to run the ActiveX control in a browser other than Internet Explorer.	Switch to the required version of the Internet Explorer web browser. In the Microsoft Edge browser, reload the page in Internet Explorer mode, if that feature is enabled.
	User has not been authenticated for that browser and that web page.	Check to see whether authentication is needed, and then authenticate. See "Connecting to Web Servers That Require Authentication" on page 316.
	Browser does not recognize the file as HTML.	Ensure that the type of the HTML file is correctly specified.
		Ensure that the DOCTYPE and MIME tags are correctly formatted.
	Browser permissions too restrictive.	Check browser permissions. See "Checking Browser Permissions" on page 315.
	Browser cannot run the ActiveX control.	Specify the location of the ActiveX control files. See "Specifying the Location of the ActiveX Control Files (CODEBASE= Option)" on page 1645. Check browser permissions for running Java scripts.

Symptom	Cause	Remedy
		Check browser permissions for running Java scripts. See "Checking Browser Permissions" on page 315.
Browser displays pop-up message `Error: Not enough virtual memory to produce plot.`	Client RAM is insufficient for rendering.	Generate a new graph using a smaller data set or a simpler graph. If using PROC GMAP, consider using PROC GREDUCE.
Graph is not rendering as specified by the ODS graph style.	A style attribute might not be enabled for your ODS destination.	Ensure that the attribute is enabled for your ODS destination. For example, the URL attribute is not enabled for the PS destination. Refer to the table of style attributes for the STYLE statement of the TEMPLATE procedure in *SAS Output Delivery System: User's Guide*.
	A style attribute might be overridden by a global option, global statement option, procedure option, or statement option.	Specify the minimum options needed for your graph, for example:`goptions reset=all device=svg;`
In ActiveX, the user gets the message `There is a pending reboot for this machine...`	1 Virus-scanning software might be interfering with the installation of the control. 2 Other instances of the control might be running.	1 Turn off any virus-scanning software before installing the control. 2 Close all instances of Internet Explorer, and then install the SAS/GRAPH ActiveX control. For ActiveX control installation instructions, see "Installing the SAS/GRAPH ActiveX Control" on page 197.
Text font is incorrect.	Java font is defined differently.	Change browser fonts or change the SAS/GRAPH program. See "Correcting Text Fonts" on page 317.

Symptom	Cause	Remedy
Text in browser shows incorrect characters.	Browser misinterpreting special characters.	Replace special characters with character entities. See "Using HTML Character Entities" on page 316.
Graph in browser differs from graph in SAS.	A graphics option or global statement might be unsupported or partially supported for the ActiveX control.See also "Resolving Differences between Graphs Generated with Different Technologies" on page 317.	Refer to the descriptions for the options that you are using and to Appendix 1, "Summary of ActiveX and Java Support," on page 1581 for information about whether a statement or option is supported.
	A default value in the ActiveX control is overriding a default option value.	Specify a value for the option rather than relying on the default. See "Resolving Differences between Graphs Generated with Different Technologies" on page 317.
In ActiveX, black-and-white image is not displayed	ActiveX does not enable 8-bit grayscales images.	Convert the image to 24-bit monochrome.
Graph loses attributes after graph type is changed in the web browser.	Some attribute loss is inherent in graph type changes.	Select the **Refresh** button in the web browser to restore the original graph.
Changes made through the Data Options dialog box cause the graph to revert to its original view.	The graph discards subsetting information if you make changes through the Data Options dialog box.	Make any changes needed through the Data Options dialog box before subsetting the graph.

Checking Browser Permissions

Access permissions vary from browser to browser, but some form of access control is enforced in most browsers. To check your permissions, open the browser's Preferences or Internet Options window. Then look for the advanced options. Use

your browser's Help system and contact your system support representative as needed to ensure that the browser permissions allow the following:

- Style sheets
- JavaScripts

In the **Security** tab of the Internet Explorer's Internet Options window, make sure that the selected web content zone enables access to the web presentation.

Using HTML Character Entities

A special character in your web presentation might not resolve in the browser. The character might need to be changed to a character entity in the source file or in the SAS program. A character entity is a standardized string of characters that represents a special character. The browser recognizes the string and replaces it with the special character when it is formatting the display. One common character entity is > . This entity represents the greater-than symbol (>).

Lists of standard character entities are provided in HTML reference books and in HTML references on the Worldwide web.

Connecting to Web Servers That Require Authentication

If you are unable to install the ActiveX control, then you might be trying to access a web server that requires authentication. To resolve this problem, access a different file on that server and enter your user ID and password. Redisplaying your web presentation should now enable you to access that web server.

Setting the SAS_ALT_DISPLAY Variable for X Window Systems on UNIX

You might need to define a special environment variable, SAS_ALT_DISPLAY, because some server features require a valid X Windows System graphics display. This environment variable is used to locate a graphic display when the value of the environment variable commonly used by the X Window System, DISPLAY, has not been set. The value of SAS_ALT_DISPLAY must refer to a display that is always available during the operation of a SAS server. For example, if the server machine on which SAS servers are running also runs an X server, then set the value of SAS_ALT_DISPLAY to the name of the server machine. To set the

SAS_ALT_DISPLAY environment variable, edit the file *!SASROOT/bin/sasenv* and substitute your display name for *value:0.0* in the line,

```
SAS_ALT_DISPLAY=value:0.0
```

If an X server is not available on the server machine, an alternative is to use the X virtual frame buffer (Xvfb) as supplied by the operating system vendor. Refer to your vendor-supplied documentation for information about the use of Xvfb.

Correcting Text Fonts

Your presentation might display an incorrect text font on a given client computer. The cause might be that the client computer maps a logical font name such as Courier to a different physical font set. If the logical font is not mapped to any physical font, the JAVAIMG device uses a default font.

When you are using the ACTIVEX or ACTXIMG device, the actual fonts used are determined at run time. The fonts are resolved based on the fonts available on the system where the graph is viewed. When you use the ACTIVEX device, the fonts specified by the styles are also specified in the HTML or RTF file that is generated. When the file is viewed, if a font is not available, the font mapper on the system where the file is viewed determines the font that is substituted.

It is recommended that you specify system fonts whenever possible. See “Determining What Fonts Are Available” on page 351 and “TrueType Fonts Supplied by SAS” on page 350 for more information.

Resolving Differences between Graphs Generated with Different Technologies

Graphics output that is rendered with the ACTIVEX, ACTXIMG, or JAVAIMG device is rendered using Java or ActiveX technology. Graphics output that is rendered with other devices such as PNG, GIF, or SVG is rendered with SAS technology.

Because of technological differences between SAS, Java, and ActiveX, output generated with these different technologies might differ from each other. This is true even if the output is generated with the same SAS procedure code. The graphs might differ in appearance, in the default values used for certain options, or in the availability of certain features. See Chapter 15, “Generating Interactive ActiveX Graphics,” on page 191.

For example, differences might occur if you are using a global statement or procedure option that is not enabled for the ActiveX control. Most global statement and procedure options are fully supported by the ACTIVEX, ACTXIMG, and JAVAIMG device drivers. Exceptions are identified in the procedure and statement documentation and summarized in Appendix 1, “Summary of ActiveX and Java Support,” on page 1581.

In certain cases, differences between graphs can occur when the ActiveX control overrides the default value of a procedure option. To resolve this issue, specify a value for the option rather than relying on the default.

PART 4

Controlling the Appearance of Your Output

21

Using ODS Styles, Device Parameters, and Options

Overview

The appearance of SAS/GRAPH output is determined by ODS styles by default. Along with table and page attributes, ODS styles contain a collection of graphical attributes such as color, marker shape, line pattern, fonts, and so on. Many carefully designed styles that enhance the visual impact of the graphics are shipped with SAS. In addition to creating visually appealing graphics, the styles ensure that different groups of data can be easily distinguished from one another. They also ensure that data of equal importance is given equal visual emphasis.

These styles produce professional-looking graphics without additional code in your SAS programs and without modifying the styles themselves. However, you can use SAS/GRAPH statement options to override specific elements in the styles, or you can modify style elements to create a customized style for yourself or your organization.

Table 21.1 *Controlling Graph Appearance*

Method	Description	Level of Complexity	Reference
Specify a different style template.	Specify a style template with the STYLE= option to change the appearance of the entire graph. Requires no further modification.	Low	"Changing the Current Style By Using the STYLE= Option in ODS Destination Statements" (p. 328)
Use appearance options.	Specify an appearance option using SAS/GRAPH procedure options or global statement options to change various aspects of your graph. This method requires modification of your SAS/GRAPH program.	Medium	"Overriding Style Attributes with SAS/GRAPH Statement Options" (p. 329)
Modify individual style elements.	Specify or change style attributes in order to modify a style element. This requires the use of PROC TEMPLATE style statements.	High	"Modifying a Style" (p. 334)

You can turn off the use of styles if needed. In this case, the default appearance of your output is controlled by device entry parameters. See "Style Attributes versus Device Entry Parameters" on page 323 and "Turning Off Styles" on page 347 for more information.

Style Attributes versus Device Entry Parameters

The default appearance of SAS/GRAPH output is determined by either style attributes or device entry parameters. Which one depends on the setting of the GSTYLE system option and on the device that is being used.

By default, the GSTYLE system option is in effect, and the appearance of all SAS/GRAPH output is determined by style attributes. If the NOGSTYLE system option is in effect, then the device entry parameters govern the appearance of SAS/GRAPH output for all devices except the ActiveX device. The ActiveX device always uses styles to determine appearance. The setting of the GSTYLE system option has no effect on the ActiveX device.

Table 21.2 *The GSTYLE System Option and Default Appearance*

Current Device	GSTYLE	NOGSTYLE
ActiveX device	style	style
All other devices	style	device entry parameters

For information about device entries, see "What Is a SAS/GRAPH Device?" on page 90 and "Viewing and Modifying Device (DEV) Entries" on page 104. See also "Output from Different Devices and the GSTYLE/NOGSTYLE System Options" on page 330, "Changing the Appearance of Output to Match That of Earlier SAS Releases" on page 347, and "Turning Off Styles" on page 347.

About Style Templates

An ODS style is a collection of named *style elements* that provides specific visual attributes for your graphical and tabular SAS output. Each style element is a named collection of *style attributes* such as background color, text color, marker symbol, line style, font face, font size, as well as many others. Each graphical element of a plot, such as a marker, a bar, a line or a title, derives its visual attributes from a specific style element from the current style. Note that the style that a destination uses is applied to tabular output as well as graphical output.

ODS Destinations and Default Styles

Every ODS output destination, except the Document and Output destinations, is associated with a default style. These styles are customized for each destination. Therefore, your output might look different depending on which destination you use. If your program does not specify a style, SAS uses the styles listed in Table 21.3 on page 324.

Table 21.3 *Default Style Templates*

ODS Destination	Default Style Name
DOCUMENT	Not applicable
LISTING	Listing
OUTPUT	Not applicable
HTML	Default (Styles.Default)
LATEX	Default (Styles.Default)
PRINTER	Printer
RTF	Rtf
Measured RTF	Rtf

The default style for each destination is set in the SAS registry. Changing the style specified in the SAS registry can be a convenient way to apply a company's style to all output sent to all destinations. See “Changing the Default Style in the SAS Registry” on page 328.

Chapter 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 55 shows examples of graphs using several styles, including the default styles for the most commonly used destinations. “Examples of Output Using Different Styles” on page 325 shows examples of graphs and tables using the Printer, Rtf, Analysis, and Journal styles.

Recommended Styles

SAS provides a set of styles that have been designed by GUI experts to address the needs of different situations. Table 21.4 on page 325 describes a subset of the styles provided by SAS that are particularly well-suited to displaying graphics.

Table 21.4 Recommended Style Templates

Desired Output	Recommended Styles	Comments
Full Color	Default (Styles.Default)	Gray background, optimized for HTML output
	Analysis	Yellow background
	Statistical	White background, colored fills
	Listing	White background, optimized for color format on white paper
	Printer	White background; serif fonts; optimized for PS and PDF output
	Rtf	Similar to Printer; optimized for RTF output
Black and White	MonochromePrinter	Black and white output; patterned fills; optimized for PCL output
	Journal2	Interior filled areas have no color
Gray Scale	Journal	Interior filled areas are gray scale

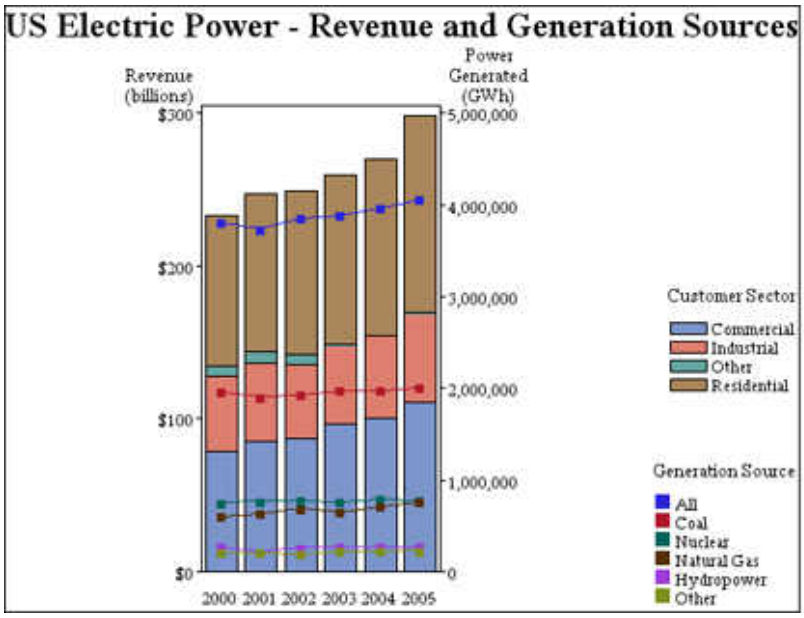
Note: Certain ODS styles map textures onto graph elements.

Chapter 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 55 shows examples of graphs using several styles, including the default styles for the most commonly used destinations. “Examples of Output Using Different Styles” on page 325 shows examples of graphs and tables using the Printer, Rtf, Analysis, and Journal styles.

Examples of Output Using Different Styles

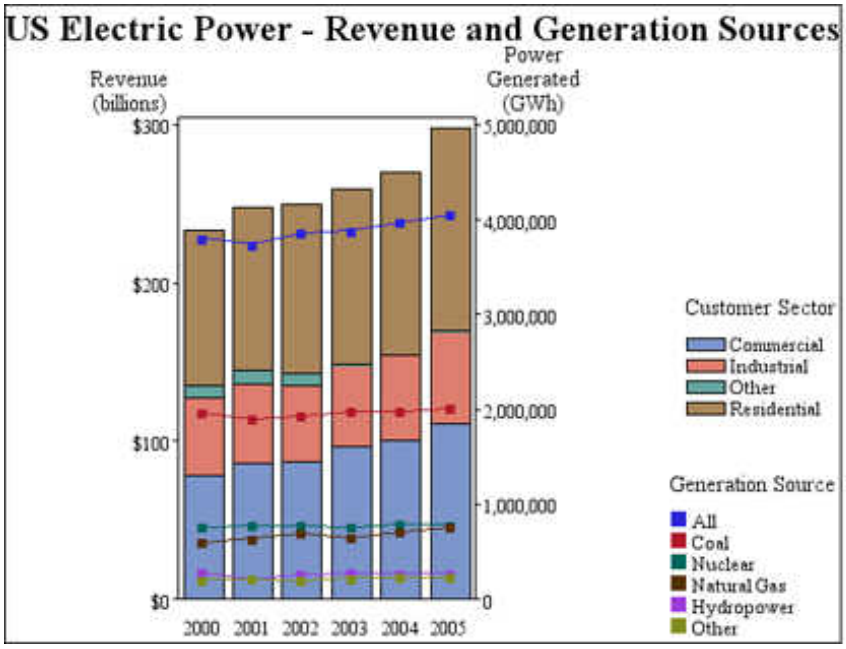
Each of the following sets of output was created using a different style. Additional examples of output in Chapter 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 55.

Figure 21.1 Output Using the Printer Style



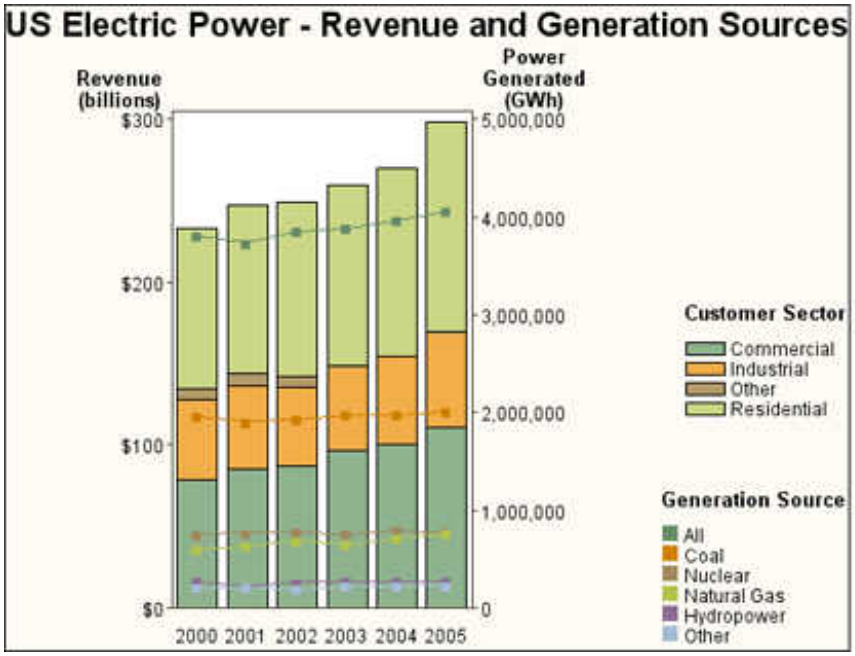
Obs	Customer	Revenue	Year
1	Commercial	$63	1994
2	Industrial	$48	1994
3	Other	$7	1994
4	Residential	$85	1994
5	Commercial	$66	1995

Figure 21.2 Output Using the RTF Style



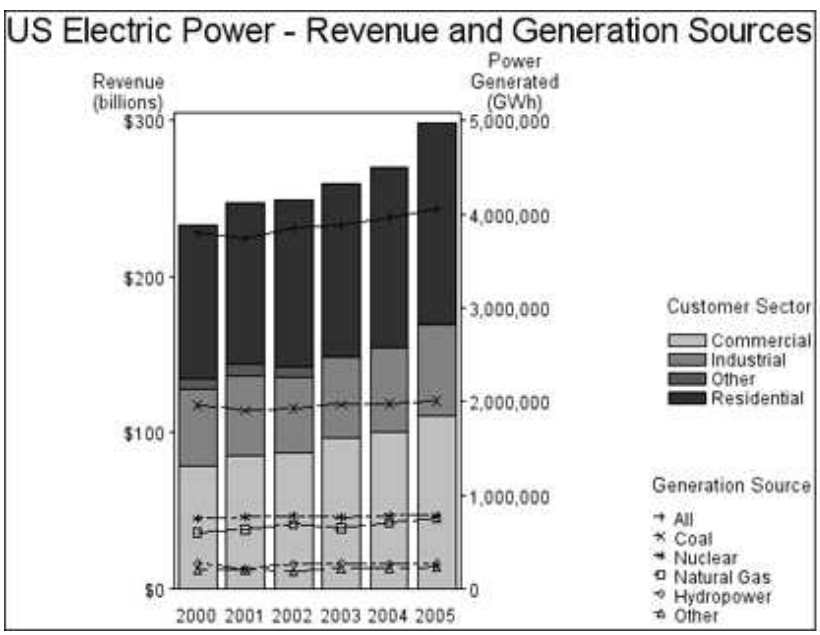
Obs	Customer	Revenue	Year
1	Commercial	$63	1994
2	Industrial	$48	1994
3	Other	$7	1994
4	Residential	$85	1994
5	Commercial	$66	1995

Figure 21.3 *Output Using the Analysis Style*



Obs	Customer	Revenue	Year
1	Commercial	$63	1994
2	Industrial	$48	1994
3	Other	$7	1994
4	Residential	$85	1994
5	Commercial	$66	1995

Figure 21.4 *Output Using the Journal Style*



Obs	*Customer*	*Revenue*	*Year*
1	Commercial	$63	1994
2	Industrial	$48	1994
3	Other	$7	1994
4	Residential	$85	1994
5	Commercial	$66	1995

Note: The table in Figure 21.4 on page 327 was sent to the PDF destination.

Specifying a Style

Changing the Current Style By Using the STYLE= Option in ODS Destination Statements

Changing the current style for an ODS destination is the easiest, simplest way of changing the appearance of your output. Changing the current style requires only the use of the STYLE= option in an ODS destination statement. By specifying only STYLE=*style-definition* in your ODS destination statement, you can create an entirely different appearance for your graphs. For example, you can specify that ODS apply the Styles.Journal style template to all HTML output with one of the following statements:

```
ods html style=styles.journal;
ods html style=journal;
```

This style is applied to all output for that destination until you change or close the destination or start a new SAS session.

Changing the Default Style in the SAS Registry

By default, the SAS registry applies a default style to the output for each ODS destination.The default styles for each destination are listed in Table 21.3 on page 324. To permanently change the default style associated with a destination, you can change the setting of Selected Style in the SAS registry.

CAUTION
If you make a mistake when you modify the SAS registry, then your system might become unstable or unusable. See "Managing the SAS Registry" in *SAS Programmer's Guide: Essentials*.

Note: You might have more than one SAS registry. Each site has a SAS registry in SASHELP. Each directory from which you run SAS has an individual registry in SASUSER. If you run SAS from multiple locations, and you want to change default styles via the SAS registry, you might need to change it in multiple locations. For more information, see "Introduction to the SAS Registry" in *SAS Programmer's Guide: Essentials*.

For more information about ODS and the SAS registry, see "Changing SAS Registry Settings for ODS" in *SAS Output Delivery System: User's Guide* in *SAS Output Delivery System: User's Guide*.

Overriding Style Attributes with SAS/GRAPH Statement Options

By default, the attributes of various elements of the graph are derived from specific style elements (or from device entry parameters if the NOGSTYLE system option is in effect). Exceptions occur when the attributes are explicitly overridden with procedure or global statement options. For example, you can use the CTITLE= and CTEXT= options in the GOPTIONS global statement to change the color of the text in all of your graphs. You can use the SYMBOL statement to specify colors for markers. The settings remain in effect until you change them or end your SAS session. For information about GOPTIONS, see "GOPTIONS Statement" on page 430 and "Specifying Colors in a GOPTIONS Statement" on page 365. See the examples in "SAS/GRAPH Statements" on page 397.

Instead of specifying global options, which affect all of your SAS/GRAPH output, you can specify options on specific action statements that affect only the output produced by that statement. Values that you specify on procedure action statements override default style attributes (or device entry parameters) and global options. For an example, see "Example 5: Modifying the Appearance of Radar Charts" on page 1405.

The documentation for each option that overrides a style element includes the name of the style element and attribute. For example, the documentation for the "CAXIS=*grid-color*" on page 1005 GCHART procedure includes the following style reference information:

CAXIS=

Style reference Color attribute of the GraphAxisLines element

You might want to change the color of the same graphical elements that are affected by the CAXIS= option by modifying a style. In this case, you need to modify the Color attribute of the GraphAxisLines element. See "Modifying a Style" on page 334 for more information.

Attributes that are used repeatedly might be best specified in an ODS style. However, if you have created a customized style, be aware that you might need to make this style available to anyone to whom you send your SAS code.

Attributes that are used only once or occasionally are best specified using SAS/GRAPH statements.

Precedence of Appearance Option Specifications

When you specify options that override style attributes or device parameters, the general order of precedence that SAS/GRAPH uses is as follows:

1. options in a SAS/GRAPH procedure action statement
2. options in AXIS, FOOTNOTE, LEGEND, NOTE, PATTERN, SYMBOL, or TITLE statements
3. graphics options in a GOPTIONS statement
   a. color options in the GOPTIONS statement that control specific graph elements such as the background color or title text color
   b. the color list specified with the COLORS= option in the GOPTIONS statement
4. attributes specified in the current style or, if the NOGSTYLE option is in effect, device parameters in a device entry for the current device
5. default hardware settings for a device

SAS/GRAPH uses the first specification that it finds in this list. Any exceptions to this rule are noted in the documentation for the specific option as described in “Overriding Style Attributes with SAS/GRAPH Statement Options” on page 329.

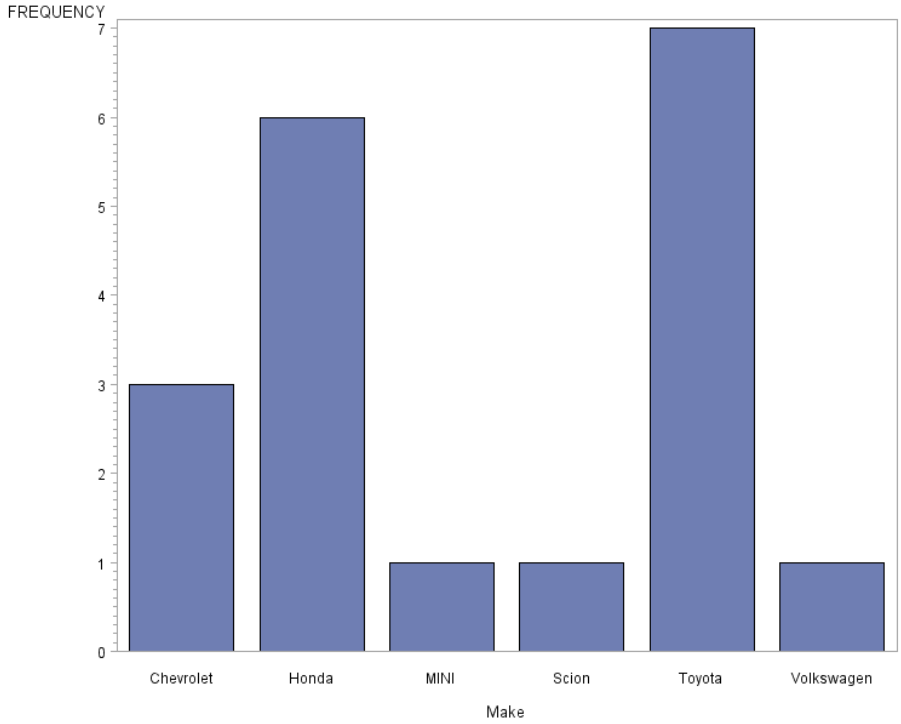
Output from Different Devices and the GSTYLE/NOGSTYLE System Options

The GSTYLE and NOGSTYLE system options might or might not have an effect on the output from different devices. The non-interactive devices, namely SVG, PNG, GIF, and TIFF all support ODS styles. However, the ACTXIMG and JAVAIMG devices are not affected by the GSTYLE|NOGSTYLE system option. An ODS style is applied to an ACTXIMG or JAVAIMG image regardless of the setting of this system option. Let us demonstrate the impact of the GSTYLE and NOGSTYLE system options on the device output. Here is an example that uses the GSTYLE system option with the default ODS destination and graphics output device.

```
options gstyle;
proc gchart data=sashelp.cars;
   vbar Make;
    where MPG_Highway >= 37;
   run;
```

```
quit;
```

Figure 21.5 *A Bar Chart Using the GIF Device with the Statistical Style and GSTYLE System Option*

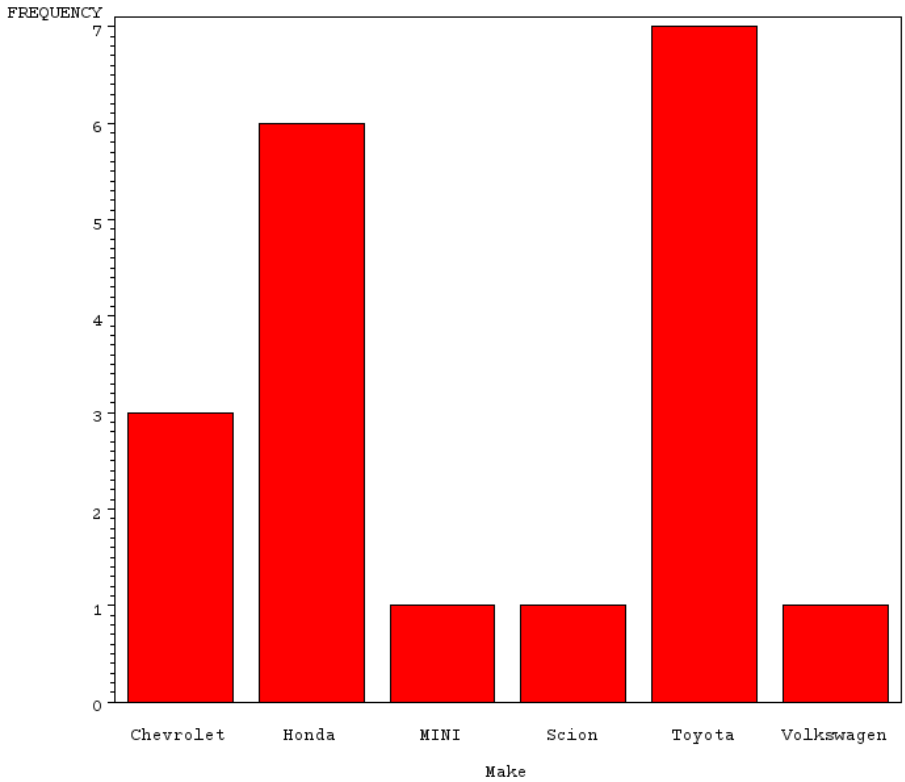


The output is similar for the other destinations and devices.

Here is the previous example with the NOGSTYLE system option on:

```
options nogstyle;
proc gchart data=sashelp.cars;
        vbar make;
        where MPG_Highway >= 37;
run;
quit;
```

Figure 21.6 A Bar Chart Using the GIF Device with the Statistical Style and the NOGSTYLE System Option



Notice that the default style is overridden by the NOGSTYLE system option and that no style is applied to the graph. The NOGSTYLE system option is valid only for the SVG, PNG, GIF, and TIFF devices. For the ACTXIMG and JAVAIMG devices, the NOGSTYLE system option has no effect. In this example, if the DEVICE=JAVAIMG or DEVICE=ACTXIMG graphics option is used, the default style is applied even though the NOGSTYLE system option is on.

Additional Considerations with ODS Styles

- Some ODS styles, such as Dtree, Netdraw, and BlockPrint, include background images. Not all devices support these images. You can suppress the display of these images by specifying the NOIMAGEPRINT graphics option. See "Disabling and Enabling Image Output" on page 393.

  Note: Starting with SAS 9.4M5, the BlockPrint ODS style is removed from suggested use. This style still functions when used in SAS programs, but it no longer appears in the list of available styles.

- Not all the styles work with all the chart types. For example, the default colors for axis labels might blend into the background color of the style. Specify a different style.

Viewing the List of Styles Provided by SAS

You can view the styles that SAS provides using the TEMPLATE procedure or through the Templates window.

Using the TEMPLATE Procedure

To view the list of all styles available, submit the following code:

```
proc template;
     list styles;
run;
```

SAS writes the list of available styles in the Output window.

Using the Templates Window

To view the list of all styles available, follow these steps:

1. Open the Templates window. You can open the Templates window in two ways:
   - Enter the `odstemplates` command on the SAS command line.
   - In the Results window, select the Results folder. Right-click and select **Templates** to open the Templates window.

   The Templates window contains the item stores **Sasuser.Templat** and **Sashelp.Tmplmst**.
2. Double-click an item store, such as **Sashelp.Tmplmst**, to expand the list of directories where ODS templates are stored. The templates that SAS provides are in the item store **Sashelp.Tmplmst**.
3. Double-click **Styles** to view the list of styles defined in the selected item store.
4. Double-click the style that you want to view. For example, the HTMLBlue style is the default style for HTML output. Similarly, the Rtf style is the default style for RTF output.

To view the actual style template, double-click a style name. The style template is displayed in the Template Browser window.

Modifying a Style

Using the TEMPLATE Procedure

Within the TEMPLATE procedure, you can use the DEFINE STYLE statement to create a completely new style or you can start from an existing style. When you create styles from existing styles, you can modify the individual style elements.

For complete documentation about using PROC TEMPLATE to modify and create styles, see "TEMPLATE Procedure: Creating a Style Template" in *SAS Output Delivery System: Procedures Guide* in *SAS Output Delivery System: User's Guide*.

Example: Modifying a Style Element

The style element GraphData1 is defined in the Default style as follows:

```
proc template;
   define style Styles.Default;
        ...more style elements...
   class GraphData1 /
           markersymbol = "circle"
           linestyle = 1
           contrastcolor = GraphColors('gcdata1')
           color = GraphColors('gdata1');
```

You can use the DEFINE STYLE statement in the TEMPLATE procedure to create a new style from the Default style and modify the GraphData1 style element. The following program creates the new style MyStyleDefault, which inherits all of its style elements and style attributes from the Default style, and modifies the GraphData1 style element:

```
proc template;
   define style MyStyleDefault;
   parent=Styles.Default;
   style GraphData1 from GraphData1 /
         markersymbol = "triangle"
         linestyle = 2
         contrastcolor = GraphColors("gcdata1")
         color = GraphColors("gdata1");
    end;
run;
```

The new GraphData1 uses the same colors as the original GraphData1, but specifies a different marker symbol and line style.

To use the new MyStyleDefault style for HTML output, specify the STYLE= option:

```
ods html style=MyStyleDefault;
```

Ways to Modify Graph Fonts or Colors Specified by Styles

There are different ways to change the fonts or colors used by a style. Which method you choose depends on how extensively you want to change the font or color specifications used in your output. You can do any of the following:

- Modify a specific style element that controls a specific graphical element. For example, the GraphValueText element specifies the font and color for tick mark values and legend value descriptions. You could change the font or color specified by the GraphValueText element for the Analysis style. Changes to specific style elements affect only the graphical elements that they control and affect them in only the styles where you change them. See "Style Elements for Use with SAS/GRAPH Output" on page 339 for information about the specific style elements that you can modify.
- Modify the font or color specifications in the GraphFonts or GraphColors style elements for a specific style. The settings specified in GraphFonts and GraphColors are referenced by specific style elements elsewhere in the style. Other style elements that reference the GraphFonts or GraphColors style elements use the modified settings. See "The GraphFonts Style Element" on page 338 and "The GraphColors Style Element" on page 337 for more information. A single change in the specifications in the GraphFonts or GraphColors style elements can potentially change the appearance of several graphical elements. And it can affect output of any style that refers to GraphFonts or GraphColors.
- Modify the font settings for one or more subkeys in the SAS registry. Many styles refer to the font settings in the SAS registry to determine the fonts to use for various graphical elements. Modifying the SAS registry settings changes the fonts used for all styles that refer to the subkeys that you change. For more information, see "Changing SAS Registry Settings for ODS" in *SAS Output Delivery System: User's Guide*. (Colors used by the styles supplied by the company are not controlled through the SAS registry.)

Modifying the GraphFonts and GraphColors Style Elements

The attributes in the GraphFonts and GraphColors style elements are used as the values for specific style elements elsewhere in the style. In other words, the GraphFonts and GraphColors elements are *abstract elements*. They are used to assign values to other elements.

For example, the GraphFonts element could be defined follows:

```
class GraphFonts
         "Fonts used in graph styles" /
         'GraphDataFont' = ("<sans-serif>, <MTsans-serif> ",7pt)
         'GraphValueFont' = ("<sans-serif>, <MTsans-serif>",9pt)
```

```
        'GraphLabelFont' = ("<sans-serif>, <MTsans-serif> ",10pt,bold)
        'GraphFootnoteFont' = ("<sans-serif>, <MTsans-serif>",10pt)
        'GraphTitleFont' = ("<sans-serif>, <MTsans-serif>",11pt,bold);
```

Each attribute, GraphDataFont, GraphValueFont, GraphLabelFont, and so on, defines a list of fonts for use by SAS/GRAPH whenever the corresponding attribute is referenced. These attributes are specified elsewhere in the style as the value of a another font attribute. (For information about the syntax used in the GraphFonts style element, see "Font Specifications in the GraphFonts Style Element" on page 338.)

For example, the GraphValueText element specifies the font and color for tick mark values and legend value descriptions. Suppose the GraphValueText element is defined as follows:

```
class GraphValueText /
        font = GraphFonts('GraphValueFont')
        color = GraphColors('gtext');
```

The font and color for GraphValueText are specified by elements in the GraphFonts and GraphColors style elements.

`GraphFonts('GraphValueFont')`
: tells SAS/GRAPH to use the font specified by the GraphValueFont attribute in the GraphFonts style element.

`GraphColors('gtext')`
: tells SAS/GRAPH to use the color specified by the gtext attribute in the GraphColors style element.

To change the font and color for tick mark values and legend value descriptions, you could modify either of the following:

- the FONT= and COLOR= attributes in the GraphValueText element
- the GraphValueFont attribute in the GraphFonts style element and the gtext attribute in the GraphColors style element.

However, because elements in GraphFonts and GraphColors are referred to by other elements in the style, changing the values in GraphFonts and GraphColors result in more extensive changes. A targeted change involves modifying a specific style element such as GraphValueText directly. If you modify the GraphValueText element directly, your modifications affect only the items controlled by GraphValueText. If you modify the GraphValueFont or gtext attributes, then your modifications might affect other portions of the graph in addition to tick mark values and legend value descriptions. This list includes pie labels, regression equations, data point labels, bar labels, and graph titles.

The styles supplied with SAS/GRAPH are designed to provide a consistent visual appearance for all graphical elements in your output. Modifying attributes in the GraphFonts or GraphColors elements instead of modifying several specific style elements makes it easier to maintain the consistent appearance in your output.

The tables listed in "Graphical Style Element Reference for Device-Based Graphics" on page 337 describe the portions of SAS/GRAPH output that are affected by elements and attributes defined in the styles.

Graphical Style Element Reference for Device-Based Graphics

The GraphColors Style Element

The GraphColors style element specifies the colors that are used for different categories of graphical elements.

Table 21.5 on page 337 lists the style attributes that are defined in the GraphColors style element and the graphical elements that they affect by default.

Table 21.5 *GraphColors Attributes for Device-Based Output*

GraphColors Attribute[1]	Portion of Graph Affected
gaxis	Axis lines and tick marks
gborderlines	Border around the graph wall, legend border, and borders to complete axis frame
gconnectLine	Line for connecting boxes
gfloor	Graph floor
ggrid	Grid lines
glabel	Axis labels and legend titles
glegend	Background of the legend
goutline	Outlines for data primitives such as bars, pie slices, and boxes
gshadow	Drop shadows used with text
gtext[2]	Graph titles, tick mark values, and legend value descriptions
gwalls	Frame area in two-dimensional graphs and vertical walls in three-dimensional graphs
gdata1–gdata12 gcdata1–gcdata12	Data items; gdata1–gdata12 apply to filled areas; gcdata1–gcdata12 apply to markers and lines

GraphColors Attribute[1]	Portion of Graph Affected
gramp2cstart gramp2cend	Gradient contours, surfaces, continuous choropleth maps, and continuous block maps when areas are not used
gconramp2cstart gconramp2cend	Continuous block maps when areas are used

1 Elements in the GraphColors style element that are not included in this table are used with template-based (ODS Graphics) output only.

2 The gtext attribute does not affect text that is not rendered as part of the graph. See also "Controlling Titles and Footnotes" on page 115.

The GraphFonts Style Element

The GraphFonts style element specifies the fonts that are used for different categories of graphical elements. Table 21.6 on page 338 lists the style attributes that are defined in the GraphFonts style element and the graphical elements that they affect by default.

Table 21.6 *GraphFonts Attributes for Device-Based Output*

GraphFonts Attributes[1]	Portion of Graph Affected
GraphDataFont	Contour labels
GraphValueFont	Axis tick mark labels, legend value description labels, data values in statistics tables, pie labels, regression equations, data point labels, bar labels
GraphLabelFont	Axis labels, legend labels, column headings in statistics tables
GraphFootnoteFont	Footnotes
GraphTitleFont	Titles

1 The GraphUnicode and GraphAnnoFont attributes are used with ODS graphics only.

Font Specifications in the GraphFonts Style Element

Font definitions in the GraphFonts style element can refer to registry entries, they can specify a specific font, or they can specify a font family. For example:

```
'GraphLabelFont' = ("<MTsans-serif>, Arial, sans-serif",10pt,bold)
```

`<MTsans-serif>`
specifies the font family identified by the `MTsans-serif` subkey in the SAS registry. The less than and greater than signs tell SAS that this is the name of a subkey in the SAS registry. Because it is the first font listed, SAS uses this font if possible. To view the font settings in the SAS registry, select **ODS** ⇨ **FONTS** in the SAS registry.For more information, see "Changing SAS Registry Settings for ODS" in *SAS Output Delivery System: User's Guide* in *SAS Output Delivery System: User's Guide*.

`Arial`
specifies the Arial font family. If SAS cannot find the first font listed, it tries to find the second font listed.

`sans-serif`
specifies the sans-serif font family. If SAS cannot find the specific fonts listed, then it looks for a font in the sans-serif font family.

`10pt,bold`
specifies the weight and style that should be used.

In this example, if the SAS registry entry for the `MTsans-serif` subkey specifies Albany AMT, then SAS/GRAPH first tries to use the Albany AMT 10 point bold font. If it cannot find this font, then it tries to use Arial 10 point bold, and so on.

Note: SAS might not be able to find a specific font unless it is registered with the FONTREG procedure. The fonts provided by SAS are already registered. If you want to add additional fonts, see *SAS Programmer's Guide: Essentials* for information about registering TrueType fonts. See *Base SAS Procedures Guide* for information about the FONTREG procedure.

Style Elements for Use with SAS/GRAPH Output

The style elements listed in the following tables affect SAS/GRAPH output and can be used in styles. These tables list each style element, the portion of the graph that it affects or was created to use with, and its attribute values. Attribute values can be changed with PROC TEMPLATE, as described in "Using the TEMPLATE Procedure" on page 334 and "Example: Modifying a Style Element" on page 334 .For complete documentation about the style attributes that can be specified in each style element, see "About Style Attributes" in *SAS Output Delivery System: Procedures Guide*.

Table 21.7 *Device-Based Graph Style Elements: General Graph Appearance*

Style Element	Portion of Graph Affected	Recognized Attributes
DropShadowStyle	Used with text types	Color
Graph	Graph size and outer border appearance	OutputWidth OutputHeight BorderColor

Style Element	Portion of Graph Affected	Recognized Attributes
		BorderWidth CellPadding CellSpacing
GraphAxisLines	X, Y, and Z-axis lines	Color LineStyle LineThickness
GraphBackground	Background of the graph	Transparency BackgroundColor Gradient_Direction StartColor EndColor BackgroundImage Image VerticalAlign TextAlign
GraphBorderLines	Border around graph wall, legend border, borders to complete axis frame	Color LineThickness LineStyle
GraphCharts	All charts within the graph	Transparency BackgroundColor Gradient_Direction StartColor EndColor BackgroundImage Image VerticalAlign TextAlign
GraphDataText	Text font and color for point and line labels	Font or *font-attributes*[1] Color
GraphFloor	3-D floor	BackgroundColor Transparency Gradient_Direction StartColor EndColor BackgroundImage

Style Element	Portion of Graph Affected	Recognized Attributes
		Image VerticalAlign TextAlign
GraphFootnoteText	Text font and color for footnotes	Font or *font-attributes*[1] Color
GraphGridLines	Horizontal and vertical grid lines drawn at major tick marks	Color LineStyle LineThickness Transparency displayopts
GraphGridLines	Horizontal and vertical grid lines drawn at major tick marks	Color LineStyle LineThickness Transparency displayopts
GraphLegendBackground	Background color of the legend	Color FrameBorder Transparency
GraphOutlines	Outline properties for fill areas such as bars, pie slices, and box plots.	Color LineStyle LineThickness
GraphTitle1Text	Text font and color for the first title	Font or *font-attributes*[1] Color
GraphTitleText	Text font and color for titles subsequent to the first title	Font or *font-attributes*[1] Color
GraphValueText	Text font and color for axis tick values and legend values	Font or *font-attributes*[1] Color
GraphWalls	Vertical walls bounded by axes	Transparency BackgroundColor Gradient_Direction StartColor EndColor BackgroundImage

Style Element	Portion of Graph Affected	Recognized Attributes
		Image

1 *Font-attributes* can be one of the following: FONTFAMILY=, FONTSIZE=, FONTSTYLE=, FONTWEIGHT=.

Table 21.8 *Style Elements Affecting Device-Based Non-Grouped Graphical Data Representation*

Style Element	Portion of Graph Affected	Default Attributes
GraphCutLine	Cutline attributes for a dendogram	Color LineStyle
GraphFinal	Final data for the waterfall chart. Color applies to filled areas.	Color ContrastColor LineStyle LineThickness MarkerSize MarkerSymbol TextColor
GraphInitial	Initial data for the waterfall chart. Color applies to filled areas.	Color ContrastColor LineStyle LineThickness MarkerSize MarkerSymbol TextColor
GraphOther	Other data for the graph. Color applies to filled areas.	Color ContrastColor LineStyle LineThickness MarkerSize MarkerSymbol TextColor
GraphOverflow	Overflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.	Color ContrastColor LineStyle LineThickness MarkerSize MarkerSymbol TextColor

Style Element	Portion of Graph Affected	Default Attributes
GraphUnderflow	Underflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.	Color ContrastColor LineStyle LineThickness MarkerSize MarkerSymbol TextColor
ThreeColorAltRamp	Line contours, markers, and data labels with segmented range color response	StartColor NeutralColor EndColor
ThreeColorRamp	Gradient contours, surfaces, markers, and data labels with continuous color response	StartColor NeutralColor EndColor
TwoColorAltRamp	Line contours, markers, and data labels with segmented range color response	StartColor EndColor
TwoColorRamp	Gradient contours, surfaces, markers, and data labels with continuous color response	StartColor EndColor

Table 21.9 *Style Elements Affecting Device-Based Grouped Graphical Data Representation*

Style Element	Portion of Graph Affected	Default Attributes
GraphData1	Primitives related to 1st grouped data items. Color applies to filled areas. ContrastColor applies to markers and lines.	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData2	Primitives related to 2nd grouped data items	BackGroundImage ContrastColor

Style Element	Portion of Graph Affected	Default Attributes
		Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData3	Primitives related to 3rd grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData4	Primitives related to 4th grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData5	Primitives related to 5th grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image

Style Element	Portion of Graph Affected	Default Attributes
		LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData6	Primitives related to 6th grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData7	Primitives related to 7th grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData8	Primitives related to 8th grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol

Style Element	Portion of Graph Affected	Default Attributes
		StartColor
GraphData9	Primitives related to 9th grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData10	Primitives related to 10th grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData11	Primitives related to 11th grouped data items	BackGroundImage ContrastColor Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor
GraphData12	Primitives related to 12th grouped data items	BackGroundImage ContrastColor

Style Element	Portion of Graph Affected	Default Attributes
		Color EndColor Gradient_Direction Image LineStyle LineThickness MarkerSize MarkerSymbol StartColor

Turning Off Styles

To turn off styles, specify the SAS system option NOGSTYLE. To change the setting of the SAS system option from GSTYLE to NOGSTYLE, you can do either of the following:

- Submit the following OPTIONS statement:

  ```
  OPTIONS NOGSTYLE;
  ```

- Enter **OPTIONS** on the SAS command line, or select **Tools** ⇨ **Options** ⇨ **System** to open the SAS System Options window. Expand **Graphics**, and select **Driver settings**. Right-click **Gstyle**, select **Modify value**, and select *0=False* as the new value.

Changing the Appearance of Output to Match That of Earlier SAS Releases

Starting with SAS/GRAPH 9.2, the introduction of many new features significantly change the default appearance of your SAS/GRAPH output. To produce output that looks as if it were produced with previous versions of SAS/GRAPH, do the following:

- Specify the NOGSTYLE system option. This option turns off the use of ODS styles. See “Turning Off Styles” on page 347.
- Specify the FONTRENDERING=HOST_PIXELS system option. This option specifies whether devices that are based on the SASGDGIF, SASGDTIF, and SASGDIMG modules render fonts by using the operating system or by using the FreeType engine. This option applies to certain traditional SAS/GRAPH devices. (See “Device Categories and Modifying Default Output Attributes” on page 98.)

For example, this option works for GIF, TIFFP, and JPEG devices, but it is not applicable to PNG, SVG, SASPRT* devices.

- Specify DEVICE=ZGIF or DEVICE=ZPNG in the GOPTIONS statement when you are sending output to the HTML destination.

Note: The ZPNG device is disabled starting with SAS 9.4M2.

See "Compatibility Device Drivers (Z Drivers)" on page 106 for more information.

Related Topics

- "Setting the Size of Your Graph" on page 129
- "Setting the Resolution of Your Graph" on page 130
- Chapter 13, "Using SVG Graphics," on page 159

22

Specifying Fonts in SAS/GRAPH Programs

Introduction: Specifying Fonts in SAS/GRAPH Programs

SAS/GRAPH provides access to a variety of fonts, or typefaces, to display text and special characters for your graphics output. SAS provides a number of TrueType fonts that you can use in your applications. By default, ODS styles use system fonts,

including the TrueType fonts shipped with SAS, for the various titles, labels, and other text in SAS/GRAPH output. You can modify the default fonts by modifying the styles, by specifying graphics options, or by using font options in procedure action statements. You can specify special characters using character codes or hexadecimal codes.

For additional information about fonts, see "Managing Fonts in Universal Printing" in *SAS 9.4 Universal Printing*.

SAS/GRAPH, System, and Device-Resident Fonts

There are three types of fonts that you can use when you generate output with SAS/GRAPH.

SAS/GRAPH fonts
: fonts stored in the SASHELP.FONTS catalog, and fonts created by the user and stored in a GFONT*n* catalog. These fonts can be used only by SAS/GRAPH procedures or other procedures that generate GRSEG output files. Examples of SAS/GRAPH fonts include Swiss, Simulate, and Marker. These fonts are provided for specialized purposes only. For information about these fonts, see Appendix 3, "Using SAS/GRAPH Fonts," on page 1657.

system fonts
: fonts that can be used by any SAS procedure and by other software, such as Microsoft Word. These fonts include TrueType and Type1 fonts. Examples of system fonts include Albany AMT, Monotype Sorts, and Arial. Some system fonts, such as Helvetica, can also be present as device-resident fonts. System fonts are installed on the operating system and then registered with SAS using the FONTREG procedure. System fonts generally provide the highest quality output. SAS/GRAPH installs and registers a set of TrueType fonts, and it is recommended that you use these fonts whenever possible. See "TrueType Fonts Supplied by SAS" on page 350 for more information.

device-resident fonts
: fonts that are burned into the chips in a device's hardware. These fonts are specific to the device being used and are not portable between devices. Some device-resident fonts, such as Helvetica, can also be present as system fonts.

TrueType Fonts Supplied by SAS

When you install SAS, a number of TrueType fonts are available based on choices that were made during the installation. These fonts are referred to collectively as system fonts. You can use these fonts in your SAS programs by assigning the font name to font options, enclosed in quotation marks. For example, you can specify the following:

```
goptions ftext="Thorndale AMT";
```

For information about the TrueType fonts that are provided by SAS, see "TrueType Fonts Supplied by SAS" in *SAS 9.4 Universal Printing*

Determining What Fonts Are Available

The fonts listed in "TrueType Fonts Supplied by SAS" on page 350 are available on all systems where SAS is installed. It is recommended that you use these fonts when possible. Additional system fonts that are available to your application and the methods for determining those fonts depend on the following:

- the operating environment that you are working in
- the device or universal printer that you are using

For more information about determining what fonts are available, see *SAS 9.4 Universal Printing* and the SAS documentation for your operating environment.

You can add additional fonts to your system for use by SAS/GRAPH, but all fonts must be registered with the FONTREG procedure. See *Base SAS Procedures Guide* for more information.

All of the fonts that have been registered with the FONTREG procedure are listed in the SAS registry. To view the list of registered fonts, follow these steps:

1. Open the registry editor by either selecting **Solutions** ⇨ **Accessories** ⇨ **Registry Editor** or by issuing the command REGEDIT in the command line.
2. Select **CORE** ⇨ **PRINTING** ⇨ **FREETYPE** ⇨ **FONTS**.

The SAS/GRAPH fonts are available on all systems where SAS/GRAPH is installed, but they are provided primarily for special uses. See Appendix 3, "Using SAS/GRAPH Fonts," on page 1657 for more information.

When you are deciding what font to use, consider all operating environments in which your SAS code will be run. For example, if you specify a font, such as Arial, that is available only on Windows systems, then your output will appear different on other systems. If you specify one of the fonts that are installed with SAS (see "TrueType Fonts Supplied by SAS" on page 350), then your output will appear the same on all systems. It is recommended that you use system fonts whenever possible.

Default Fonts

Many of the default fonts are specified in the SAS registry. See "Viewing Font Specifications in the SAS Registry" on page 353. The SAS registry is localized, so fonts that are specified by the SAS registry are dependent on your locale.

For most devices, when you are using styles (the GSTYLE system option is in effect), fonts are specified by the current style. Each style specifies fonts for various

graph elements such as axis labels, graph titles, tick mark labels, and so on. See "Modifying the GraphFonts and GraphColors Style Elements" on page 335 and "The GraphFonts Style Element" on page 338 for more information about the font specifications in the styles. See "Style Attributes versus Device Entry Parameters" on page 323 for more information about the GSTYLE system option.

Table 22.1 on page 352 shows the fonts used when styles are not active (the NOGSTYLE system option is in effect).

Table 22.1 *Fonts Used by Default When NOGSTYLE System Option Is in Effect*

Device	TITLE1	All other text
SVG, PNG, GIF, TIFF, SASPRTx, PCL, PS, PDF	Swiss	Font specified by `<MTmonospace>` subkey in the SAS registry
BMP, ZGIF, and ZPNG[1] in all environments except z/OS EMF on Windows Display devices for Graph window: WIN, XCOLOR, IBMPCGX	Swiss	Font specified by DMSFont
SASEMF and EMF in other operating environments	Swiss	Font specified by `<MTmonospace>` subkey in the SAS registry
CGM	Swiss	Font specified by the device entry

1 The ZPNG device is disabled starting with SAS 9.4M2.

The DMS font is controlled by the FONT= option on Windows, by the Xdefaults X resources on UNIX, and by the host display code on z/OS. For more information, see the SAS documentation for your operating environment.

Note: In some cases, SAS/GRAPH can switch to Simulate. When styles are turned off (the NOGSTYLE system option is in effect), the only default font that is scalable is the font used for TITLE1. If the height specified for other fonts is not equal to one, then SAS/GRAPH switches to Simulate. See "The SIMULATE Font" on page 1667 for more information.

The Java and ActiveX devices ignore the NOGSTYLE system option; they always use styles. When you are using the Java and ActiveX devices (which always use styles), the fonts are determined at run time. The fonts are resolved based on the fonts available on the system where the graph is viewed. When you use the ACTIVEX device, the fonts specified by the styles are also specified in the HTML or RTF file that is generated. When the file is viewed, if a font is not available, the font mapper on the system where the file is viewed determines the font that is substituted.

See "Specifying a Font" on page 353 and "Methods for Specifying Fonts" on page 359 for information about how to override default font specifications.

Viewing Font Specifications in the SAS Registry

To view the font settings in the SAS registry, follow these steps:

1. Open the registry editor by either selecting **Solutions** ⇨ **Accessories** ⇨ **Registry Editor** or by issuing the command REGEDIT in the command line.
2. Select **ODS** ⇨ **Fonts**.

Each entry in the registry consists of a name, such as `<MTsans-serif>` or `<MTmonospace>` followed by its value, such as “Albany AMT” or “Cumberland AMT”.

Note: The Fonts key contains subkeys that specify which fonts to use based on the locale.

For more information, see “Introduction to the SAS Registry” in *SAS Programmer’s Guide: Essentials*.

Specifying a Font

Specifying the FONT= Option

To specify a font in your SAS program, include a font name, enclosed in quotation marks, anywhere fonts are supported. For example, you can specify Thorndale AMT as the font for legend labels as follows:

```
legend label=(font="Thorndale AMT"  "Generation Source");
```

You can change between fonts, specify font modifiers such as `/bold`, and specify special characters. Font names are not case-sensitive. For example, the following FOOTNOTE statement prints $E=mc^2$.

```
footnote font="Thorndale AMT/bold" "E=mc" font="Albany AMT" "b2"x;
```

Specifying Font Modifiers (/bold, /italic, and /unicode)

To add a modifier such as bold or italic to a font, follow the font name with */modifier*. For example:

```
axis1 value=(font="Cumberland AMT/bold/italic" );
```

SAS/GRAPH recognizes three font modifiers.

`/bold` or `/bo`
: specifies bold text.

`/italic` or `/it`
: specifies italic text.

`/unicode` or `/unic`
: specifies special characters using Unicode code points. See "Specifying International Characters (Unicode Encoding)" on page 355 for more information.

Note: The `/unicode` modifier is not supported by the Java or ActiveX devices.

Note: With the ACTIVEX and ACTXIMG devices, you can specify only one modifier at a time. Specifying font modifiers is not supported by the JAVAIMG device.

Note: You cannot specify font modifiers if you specify the font using a registry subkey.

Using a Registry Subkey

You can specify a font by specifying a registry subkey such as `<MTsans-serif>` or `<MTmonospace>` instead of specifying a font name. For example:

```
title font="<MTsans-serif>" "My Title";
```

The font specified by the `<MTsans-serif>` registry subkey is used for the title.

The SAS registry is localized. If you specify a font using a registry subkey, the actual font that is used is the localized value specified in your registry.

See also "Viewing Font Specifications in the SAS Registry" on page 353 and "Font Specifications in the GraphFonts Style Element" on page 338.

Specifying International Characters (Unicode Encoding)

You can use the `/unicode` modifier with a hexadecimal code to print any character in a font that supports Unicode encoding. Most of the TrueType fonts listed in "TrueType Fonts Supplied by SAS" on page 350 support Unicode encoding.

For example, the following statement uses the `/unicode` modifier and a hexadecimal code (see "Specifying Special Characters Using Character and Hexadecimal Codes" on page 355) to display the symbol for the Euro sign.

```
title "Euro Symbol" font="Albany AMT/unicode" "20ac"x;
```

Unicode Character Code Charts can be found on the Unicode website at http://www.unicode.org/charts. See also "International Character Support" in *SAS 9.4 Universal Printing*.

The Java and ActiveX devices do not support the `/unicode` modifier.

Specifying Special Characters Using Character and Hexadecimal Codes

Some fonts contain characters that are not mapped to the keyboard and cannot be entered directly into a text string. To display these special characters, substitute a character code or a hexadecimal value in the text string. Hexadecimal values are recommended over character codes.

Note: You can also display special characters using *Unicode code points*. Unicode code points are specified with the `/unicode` font modifier followed by a hexadecimal value. See "Specifying International Characters (Unicode Encoding)" on page 355 for more information.

Character codes include the letters, numbers, punctuation marks, and symbols that are commonly found on a keyboard. They are usually associated with symbols or national alphabets. These codes enable you to display the character by specifying the font and using the keyboard character in the text string. For example, on Windows operating environments, to produce the character ζ, you can specify the Symbol MT font and the character code `z` in the text string.

```
title font="Symbol MT" "z";
```

Hexadecimal values are any two-digit hexadecimal numbers enclosed in quotation marks, followed by the letter x (for example, "3D"x). In double-byte character sets, the hexadecimal values contain four digits (for example, "4E60"x). Unicode characters also contain four digits.

You display characters with hexadecimal values the same way you display them with character codes. You specify the font that contains the special character and place the hexadecimal value in the text string. For example, this TITLE statement uses hexadecimal A9 to produce © in the Albany AMT font.

```
title font="Albany AMT" "a9"x;
```

Note: The character code or hexadecimal value associated with characters in a font might be dependent on the key map that is currently being used. Keymaps are not used if the `/unicode` modifier is specified, a symbol font is specified, or NOKEYMAP is specified in the font header. Contact Technical Support if you need assistance with creating or modifying key maps.

To determine the hexadecimal codes that you need to specify for a specific character, you can use the program shown in Example Code 22.1 on page 356. This program displays 224 characters of a font together with the hexadecimal codes for each character. As shown here, it displays the characters in the Symbol font. You can change the font displayed by this program to any font available on your system. Also, some fonts have many more characters than those displayed by the program below.

Note: Some fonts, such as Albany AMT, display variations due to the national characters for that locale. Symbol fonts, such as Monotype Sorts, are not affected by your locale encoding. For double-byte encodings, the second half of the table might be blank or show small rectangles.

Example Code 22.1 *SAS Program for Displaying Hexadecimal Codes for Special Characters*

```
goptions reset=all;

/***************************************************/
/* Generate the hexadecimal values.  The A values  */
/* do not include 0 and 1 because these values are */
/* reserved for commands in most hardware fonts.   */
/***************************************************/

data one;
   do a="2","3","4","5","6","7","8","9","a","b","c","d","e","f";
     do
b="0","1","2","3","4","5","6","7","8","9","a","b","c","d","e","f";
        char=input(a||b,$hex3.);
        output;
        end;
     end;
run;

/***************************************************/
/* Create annotation data set to show the          */
/* hexadecimal values and the corresponding font   */
/* characters underneath the hexadecimal value.    */
/***************************************************/

data anno;
   length text $2. style $ 25.;
   retain xsys  "3"   ysys  "3"
          tempy 95    x     0
          size  1.5   count 0
          y     0     position "6";
   set one;
```

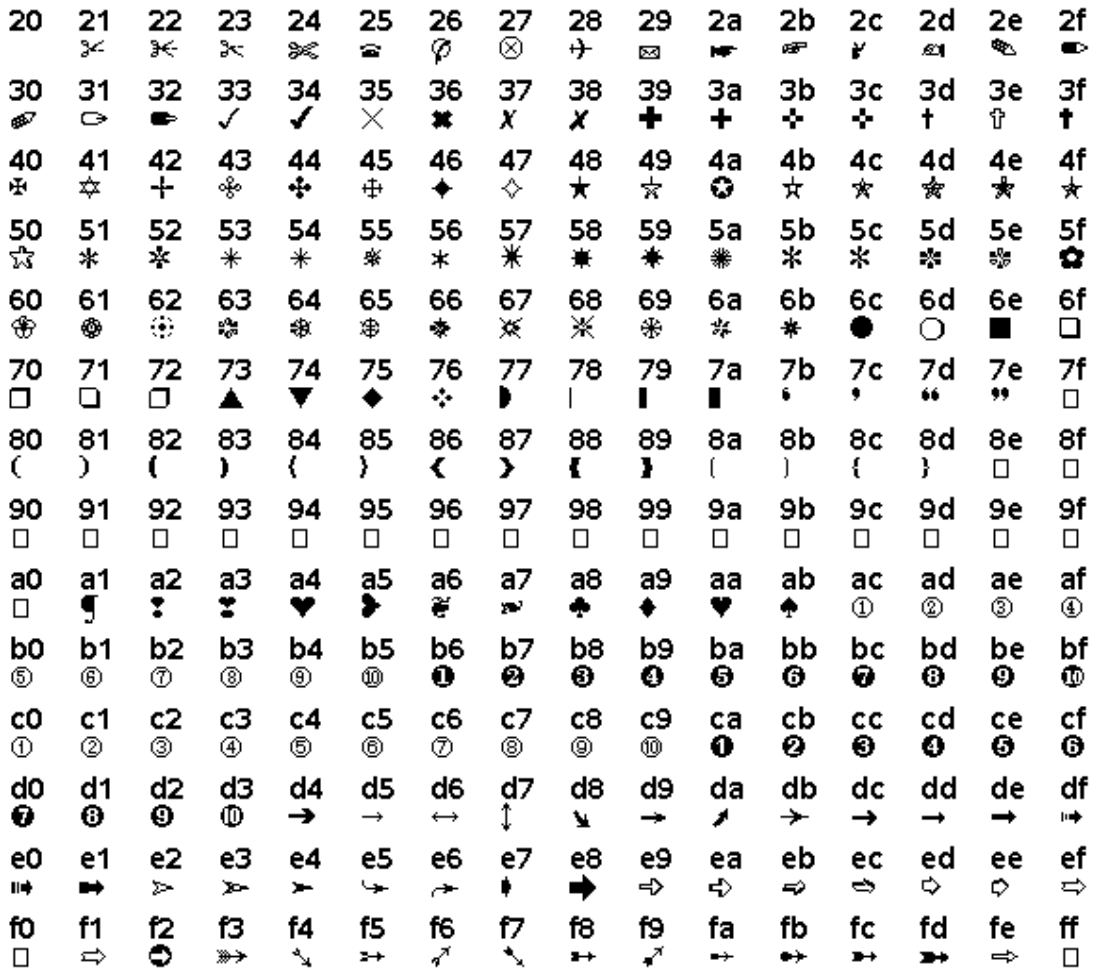
```
   count = count + 1;
   x = x + 4;

   y     = tempy;
   text  = compress(a||b);
   style = "Albany AMT/bold";
   output;

   y        = tempy - 3;
   function = "label";

   /* Modify this statement to use the */
   /* font that you want to display.   */
   style    = "Monotype Sorts";
   text     = char;
   output;

   if int(count/16) = (count/16)
      then do;
            x = 0;
                tempy = tempy - 6;
        end;
 run;

/*****************************************************/
/* Create the table.  The symbol is shown below its */
/* hexadecimal value.  For example, a circle with   */
/* the number one inside is the hexadecimal value   */
/* AC in the Monotype Sorts system font. To use     */
/* this symbol, specify:                            */
/* font="Monotype Sorts" "AC"x;                     */
/*****************************************************/

proc ganno anno=anno;
run;
quit;
```

Figure 22.1 on page 358 and Figure 22.2 on page 359 show the output of the program above for the TrueType fonts Symbol MT and Monotype Sorts.

Figure 22.1 *Symbol MT Font*

20	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**2a**	**2b**	**2c**	**2d**	**2e**	**2f**
	!	∀	#	∃	%	&	∋	(	)	∗	+	,	−	.	/
30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**3a**	**3b**	**3c**	**3d**	**3e**	**3f**
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**4a**	**4b**	**4c**	**4d**	**4e**	**4f**
≅	Α	Β	Χ	Δ	Ε	Φ	Γ	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
50	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**5a**	**5b**	**5c**	**5d**	**5e**	**5f**
Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω	Ξ	Ψ	Ζ	[	∴	]	⊥	_
60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**6a**	**6b**	**6c**	**6d**	**6e**	**6f**
‾	α	β	χ	δ	ε	φ	γ	η	ι	φ	κ	λ	μ	ν	ο
70	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**	**7a**	**7b**	**7c**	**7d**	**7e**	**7f**
π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{	|	}	~	□
80	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**8a**	**8b**	**8c**	**8d**	**8e**	**8f**
□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
90	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**9a**	**9b**	**9c**	**9d**	**9e**	**9f**
□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
a0	**a1**	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**a9**	**aa**	**ab**	**ac**	**ad**	**ae**	**af**
□	ϒ	′	≤	⁄	∞	ƒ	♣	♦	♥	♠	↔	←	↑	→	↓
b0	**b1**	**b2**	**b3**	**b4**	**b5**	**b6**	**b7**	**b8**	**b9**	**ba**	**bb**	**bc**	**bd**	**be**	**bf**
°	±	″	≥	×	∝	∂	•	÷	≠	≡	≈	…	|	—	↵
c0	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**	**c9**	**ca**	**cb**	**cc**	**cd**	**ce**	**cf**
ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
d0	**d1**	**d2**	**d3**	**d4**	**d5**	**d6**	**d7**	**d8**	**d9**	**da**	**db**	**dc**	**dd**	**de**	**df**
∠	∇	®	©	™	∏	√	·	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
e0	**e1**	**e2**	**e3**	**e4**	**e5**	**e6**	**e7**	**e8**	**e9**	**ea**	**eb**	**ec**	**ed**	**ee**	**ef**
◊	〈	®	©	™	∑	⎛	⎜	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪
f0	**f1**	**f2**	**f3**	**f4**	**f5**	**f6**	**f7**	**f8**	**f9**	**fa**	**fb**	**fc**	**fd**	**fe**	**ff**
□	〉	⎮	⎮	⎮	⎮	⎮	⎮	⎮	⎮	⎮	⎮	⎮	⎮	⎮	□

Figure 22.2 Monotype Sorts Font



Methods for Specifying Fonts

Different Ways to Specify Fonts

In general, there are four ways to specify fonts. The method that you choose depends on how extensively you want to change font specifications used in your program.

- Many procedures support font options that enable you to specify the fonts for certain graph elements. For example, with the GCHART procedure, you can use the FONT= suboption with the PLABEL= option to control the font for the pie slice labels. With the GKPI procedure, you can use the BFONT= option to specify the font for boundary labels. Changes specified using procedure options affect the output of the current invocation of the procedure only.

For information about the font options that are available for a specific procedure, see the documentation for the procedure.

- You can specify fonts in the AXIS, LEGEND, or SYMBOL global statements. Fonts specified with these statements affect the output of any procedure that references those statements. See "Using SAS/GRAPH Global Statement Options to Specify Fonts" on page 360.
- You can specify fonts in the GOPTIONS statement. The GOPTIONS statement is also a global statement, and specifications in the GOPTIONS statement affect all output in the current SAS session. Using the FTEXT= graphics option is frequently the best solution if you are dealing with any of the following situations.
  - You want to specify the fonts only for the current SAS session.
  - You want to specify the fonts only for a specific application.
  - You do not need all of your output to use the same style.
  - You do not want your code to be dependent on registry settings or a customized style. For example, you might want to run your program as a stored process or send it to others who might not have the same registry settings.

  See "Using GOPTIONS to Specify Fonts" on page 361.
- If you want all of your output to use the same ODS style, you can create a new style by copying and modifying an existing style and changing the font settings. Your new style can be used for all your ODS output at your site to ensure a consistent appearance. If you always want all of your output to have a specific appearance, then modifying a style might be the best alternative. See "Changing the Font Specifications Used by a Style" on page 361.

Using SAS/GRAPH Global Statement Options to Specify Fonts

Font options on SAS/GRAPH AXIS, LEGEND, and SYMBOL global statements enable you to specify fonts for the following:

- axis labels, reference line labels, and tick mark values
- legend labels and legend value descriptions
- contour line labels and plot point labels

For example, the following statement could be used to label contour lines:

```
symbol value="Deep" font="CUMBERLAND AMT/bold/italic";
```

See "Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend" on page 1152 for an example that uses SYMBOL statements to label contour lines.

As with the options specified in the GOPTIONS statement, options specified with these global statements remain in effect until you change them or until you start a new SAS session.

For specific information about each of the global statements, see "SAS/GRAPH Statements" on page 397.

Using GOPTIONS to Specify Fonts

The GOPTIONS statement has several options that can be used to specify fonts for your graphs.

- FBY= sets the BY line font in your graphs.
- FTEXT= sets the font for all the text in your graphs.
- FTITLE= sets the font for the first title in your graphs.

For example, to specify Cumberland AMT for all of the text in your graphs, use

```
goptions ftext="Cumberland AMT";
```

Settings specified in the GOPTIONS statement remain in effect until you change them, until you specify `reset=all`, or until you close the SAS session.

If you want most or all of the text in your output to use a single font, specifying this font with the FTEXT= graphics option is frequently the best alternative. Using the FTEXT= option in the GOPTIONS statement instead of adding font specifications to several procedure action statements in addition to other global statements makes your code easier to maintain.

Note: The FBY= option is not supported by the Java or ActiveX devices. For specific information about the GOPTIONS statement, see "GOPTIONS Statement" on page 430. Information for specific graphics options is in "Graphics Options and Device Parameters Dictionary" on page 577.

Note: When you are sending SAS/GRAPH output to the HTML or RTF destinations (MARKUP destinations), titles and footnotes can be rendered as part of your graph image or as part of the HTML or RTF files. Where your titles and footnotes are rendered determines the fonts that are used for them. See "Controlling Titles and Footnotes" on page 115 for information about the GTITLE and GFOOTNOTE destination options and the ODS USEGOPT statement.

Changing the Font Specifications Used by a Style

There are three ways to change the font specifications used by a style. Which method you choose depends on how extensively you want to change the fonts used in your output.

- You can modify the style element that controls a specific graph element such as graph titles or contour line labels.
- You can modify the abstract font specifications in the GraphFonts class. These font specifications can be referenced in multiple places in a style and affect several graph elements.

- You can modify the font settings in the SAS registry that the styles use to determine the default fonts. Changes to the SAS registry affect the fonts used by all styles that reference the SAS registry entry.

Modifying an existing style to use different fonts might be the best alternative if you need to create a style for all of your company's output. If you want to change only the fonts used in a few applications, then using the GOPTIONS statement is a better alternative.

For information about changing the font specifications used by the styles, see “Ways to Modify Graph Fonts or Colors Specified by Styles” on page 335.

Precedence of Font Specifications

When SAS/GRAPH is trying to determine the font to use for a specific graph element, it uses the first font that it finds from the following list.

1. fonts specified on procedure action statement options such as the PLABEL= option in the PIE statement in the GCHART procedure
2. fonts specified on the AXIS, LEGEND, or SYMBOL statements
3. fonts specified with the GOPTIONS global statement
4. default fonts as described in “Default Fonts” on page 351

23

Using Colors in SAS/GRAPH Programs

Colors, Styles, and Device Parameters

The appearance of SAS/GRAPH output is determined by the current ODS style by default. Styles set the overall appearance of your output, including the colors and fonts that are used. Some styles also add an image to the background of your graphs.

You can turn off the use of styles if needed. In this case, the default appearance of your output is controlled by device entry parameters. See "Style Attributes versus Device Entry Parameters" on page 323 and "Turning Off Styles" on page 347 for more information.

Whether you are using ODS styles or using device parameters, you can override the default colors and change them as described in "Specifying Colors in SAS/GRAPH Programs" on page 364. You can add images to your output as described in Chapter 24, "Adding Images to SAS/GRAPH Output," on page 383.

For additional information about colors, see "Color Support for Universal Printers" in *SAS 9.4 Universal Printing*.

Specifying Colors in SAS/GRAPH Programs

How to Specify Colors

SAS/GRAPH enables you to set colors in several ways. You can do any of the following:

- specify colors in procedure action statements for any procedures that create graphics output. For example, the CAXIS= option in the HBAR statement specifies a color for the response and midpoint axis lines. These options are described in the documentation for the individual procedures.
- specify colors in global statements that enhance procedure output: AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, and TITLE. You can also specify colors in the NOTE statement, which is a local statement, not a global statement. See "SAS/GRAPH Statements" on page 397.
- use options in the GOPTIONS statement that define colors for specific graphics elements. See "Specifying Colors in a GOPTIONS Statement" on page 365.
- define a color list with the GOPTIONS COLORS= option. See "COLORS Device Parameter and Graphics Option" on page 597.

- specify a different style, modify an existing style, or create a custom style. See Chapter 21, “Using ODS Styles, Device Parameters, and Options,” on page 321 for more information about styles.
- modify the color list in the device entry for the device that you want to use. However, the colors listed in the device entry are not used unless styles are turned off.See “Using a Device's Color List” on page 366 and Chapter 38, “GDEVICE Procedure,” on page 1163 for more information.

See “Precedence of Appearance Option Specifications” on page 330 for information about which settings take precedence when colors are set in more than one way.

Specifying Colors in a GOPTIONS Statement

The GOPTIONS statement has several graphics options that set colors for specific graphical elements. These colors are used unless they are overridden by more specific options specified on other global statements or on procedure statements.

Option	Purpose
CBACK=	Specifies the color of background for graphics output
CBY=	Specifies the color of BY lines in graphics output
CPATTERN=	Specifies the color of fill patterns
CSYMBOL=	Specifies the color of SYMBOL definitions
CTEXT=	Specifies the color of all text and the border in graphics output
CTITLE=	Specifies the color of border, plus all titles, footnotes, and notes

You can also use the COLORS= option in a GOPTIONS statement to specify a list of colors rather than specific colors for individual graphical elements. Refer to “Graphics Options and Device Parameters Dictionary” on page 577 for complete information about each of these graphics options.

Defining and Using a Color List

Introduction to the Device Color Lists

Each device is associated with a list of colors that it can use. This list is defined in the device entry for the device. You can modify this list as needed. However, this device-specific list of colors is not used unless you turn off styles by specifying the NOGSTYLE system option. See “Using a Device's Color List” on page 366.

You can also use the GOPTIONS statement to specify a list of colors for SAS/GRAPH to use instead of the device-specific color list or the colors defined by the current style. Colors specified in the GOPTIONS statement are always used regardless of the setting of the GSTYLE or NOGSTYLE system option. See “Building a Color List with the GOPTIONS COLORS= Option ” on page 366 for more information.

The color selected from a color list varies depending on the procedure using the color and graphical element that it is drawing. Usually, the first color in the list is used. However, certain procedures can select other colors. For example, if the CAXIS= option is not specified in the GCONTOUR procedure's PLOT statement, the procedure selects the second color from the color list to draw the axes. See the documentation for individual procedures for more information.

Using a Device's Color List

If you specify the NOGSTYLE system option and you do not define a color list with the COLORS= graphics option, then SAS/GRAPH uses the color list from the current device. This color list is found in the device entry of the specified device. The color list might change if you select a different device during a SAS session.

When SAS/GRAPH assigns colors from the current device's color list, this assignment uses some of the colors that you can specify for a graph. The limit on the number of colors that can be used in your output is set by the current device. For example, the PNG device is a true color device and can use up to 16 million different colors. However, the GIF device is limited to 256 colors.

To view, create, or modify a device's color list, use the GDEVICE procedure. See Chapter 38, “GDEVICE Procedure,” on page 1163.

To reset a color list back to the default color list, for the current device driver, specify the COLORS= option without specifying any colors.

```
goptions colors=;
```

Building a Color List with the GOPTIONS COLORS= Option

To build a color list, use the COLORS= option in the GOPTIONS statement. A color list specified with the COLORS= option overrides the color list of the current device. Building a color list is useful for selecting a subset of colors in a specific order for graphics output. For example, to ensure that the colors red, green, and blue are available in that order, you can specify any of the following:

```
goptions colors=(red green blue);
goptions colors=(CXFF0000 CX00FF00 CX0000FF);
goptions colors=(medium_red medium_green medium_blue);
```

You can specify colors in any color-naming schemes described in “Color-Naming Schemes” on page 368. Each value specified in a color list must be one of the following:

- a valid color name, not to exceed 64 characters

- a valid color code, not exceed eight characters

Note: The COLORS= graphics option provides only a default lookup table. Anytime you explicitly select any other colors in your SAS/GRAPH program, those colors are used to draw the graphical elements for which you have specified them.

See "COLORS Device Parameter and Graphics Option" on page 597 for more information.

Using Transparency

There are two primary uses for transparency.

- The first use is to produce output with a transparent background. For example, you might want to add your output to a web page that already has a specific background color or image. Output that has a transparent background allows the background on the web page to show through. The background has a low color opacity. The following devices produce a transparent background by default.
  - PNGT
  - SVGT

  In addition, the following devices produce output with a transparent background when you also specify the TRANSPARENCY option in the GOPTION statement:
  - EMF and EMFDUAL
  - PNG and PNG300
  - SVG, SVGVIEW, and SVGZ
  - ACTIVEX and ACTXIMG when the output is used in a Microsoft PowerPoint presentation
- The second use is to produce output with semi-transparent colors. Semi-transparent colors are particularly useful when you want to add annotations to your output while allowing the underlying graph or map to show through the annotations. Semi-transparent colors, when annotated on top of each other, blend together (alpha blending). In the following images, the red, green, and blue circles are all semi-transparent (in this case, 50% opaque). The yellow background in the first image is totally opaque. You can specify semi-transparent colors with the RGBA color model. See "RGBA Color Codes" on page 370 for information.

  The following devices use semi-transparent colors when the transparency is specified with RGBA color codes.
  - GIF

- EMF and EMFDUAL
- PCL5c
- PNG, PNGT, and PNG300
- SVG, SVGT, SVGVIEW, and SVGZ
- TIFF

Color-Naming Schemes

Overview of Color-Naming Schemes

The valid color-naming schemes in SAS are as follows:

- RGB (red green blue)
- RGBA (red green blue transparency)
- CMYK (cyan magenta yellow black)
- HLS (hue lightness saturation)
- HSV (hue saturation brightness), also called HSB
- Gray scale
- SAS color names (from the SAS Registry)
- SAS Color Naming System (CNS)

Table 23.1 on page 368 shows examples of each color-naming scheme.

Table 23.1 *Examples of SAS Color-Naming Schemes*

Color-Naming Scheme	Example
RGB	`COLORS=(CX98FB98 CXDDA0DD CXFFDAB9 CXDB7093 CXB0E0E6)`
RGBA	`COLORS=(A98FB9833 ADDA0DD66 AFFDAB999 ADB7093CC AB0E0E6FF)` **Note:** RGBA color values are not supported by Java devices. RGBA color values are supported by ActiveX devices when the output is used in Microsoft applications.
CMYK	`COLORS=("FF00FF00" "00FFFF00" "FFFFFF00")`
HLS	`COLORS=(H14055FF H0F060FF H0B485FF H07880FF)`
HSV	`COLORS=(V0F055FF V010FFFF V03BFFFF V12C55E8)`

Color-Naming Scheme	Example
Gray Scale	`COLORS=(GRAY4F GRAY6D GRAY8A GRAYC3)`
SAS Registry Colors	`COLORS=(palegreen plum peachpuff palevioletred powderblue)`

CNS Color Names	`COLOR=("very light purplish blue")`
	`COLOR=(VeryLightPurplishBlue)`
	`COLOR=(Very_Light_Purplish_Blue)`

You can also mix color-naming schemes in the same statement. For example:

```
goptions colors=(CXEE0044 "vivid blue" darkgreen);
```

Note: The hardware characteristics of your output device might cause some colors with different color definitions to appear the same. The same color is likely to appear differently on different devices and might not appear correctly on some devices. To determine whether your device supports a specific color-naming scheme, refer to your graphics device documentation.

Each of the color-naming schemes has its advantages and disadvantages based on how the output is used. For example, if you are creating a report that will be viewed only online, then specifying colors using the RGB naming scheme or the SAS color names defined in the registry might produce better results. If you are creating a report for publishing in printed form, you might want to use the CMYK color-naming scheme.

Note: Invalid color names, such as a misspelled color name, are mapped to gray, and a NOTE is issued to the SAS log. A valid color name that is not supported by the current device is mapped to the closest color that is supported by the device.

For additional information about color-naming schemes, see *Effective Color Displays: Theory and Practice* by David Travis, *Computer Graphics: Principles and Practice* by Foley, van Dam, Feiner, and Hughes, and http://colorbrewer2.org.

RGB Color Codes

The RGB color-naming scheme is usually used to define colors for a display screen. This color-naming scheme is based on the properties of light. An RGB color code defines a color by combining red, green, and blue colors in different ratios. All the colors combined together create white. The absence of all color creates black.

Color names are in the form CX*rrggbb*, where the following is true:

- CX indicates to SAS that this is an RGB color specification.
- *rr* is the red component.
- *gg* is the green component.
- *bb* is the blue component.

The components are hexadecimal numbers in the range 00–FF (0% to 100%). Each hexadecimal number indicates how much of the red, green, or blue is included in the color. Lower percentage values are darker and higher values are lighter. This scheme allows for up to 256 levels of each color component (more than 16 million different colors).

Table 23.2 *Examples of RGB Color Values*

Color	RGB Value
Red	CXFF0000
Green	CX00FF00
Blue	CX0000FF
White	CXFFFFFF
Black	CX000000

Any combination of the color components is valid. Some combinations match the colors produced by predefined SAS color names. See "Predefined Colors" on page 1683.

RGBA Color Codes

The RGBA color-naming scheme is the same as the RGB color scheme, except that RGBA includes an additional hexadecimal number for the alpha channel. This number specifies a percentage of opacity (indicating the transparency of the color). For a description of the primary uses of transparency, see "Using Transparency" on page 367.

Color names are in the form RGBA*rrggbbaa* or A*rrggbbaa*, where the following is true:

- RGBA or A indicates to SAS that this is an RGBA color specification.
- *rr* is the red component.
- *gg* is the green component.
- *bb* is the blue component.
- *aa* is the alpha channel specification, which represents a percentage of opacity.

The components are hexadecimal numbers in the range 00–FF. The alpha channel specification defines the intensity (opacity) of the color. The value 00 is transparent (0% opacity), and FF is opaque (100% opacity).

Table 23.3 *Examples of RGBA Color Values*

Color	RGBA Value
Red, very transparent	RGBAFF000015
Red, 20% opaque	AFF000033
Green, 40% opaque	A00FF0066
Blue, 60% opaque	A0000FF99
Blue, 80% opaque	A0000FFCC
Black, completely opaque	A000000FF

Any combination of the color components is valid. Some combinations match the colors produced by predefined SAS color names. See "Predefined Colors" on page 1683.

CMYK Color Codes

The CMYK color-naming scheme is used in four-color printing. CMYK is based on the principles of objects reflecting light. Combining equal values of cyan, magenta, and yellow produces process black, which might not appear as pure black. The black component (K) of CMYK can be used to specify the level of blackness in the output. A lack of all colors produces white when the output is printed on white paper.

To specify the colors from a printer's Pantone Color Lookup Table, you can use the CMYK color-naming scheme. Specify colors in terms of their cyan, magenta, yellow, and black components. Color names are in the form CMYK*ccmmyykk* or K*ccmmyykk*, where the following is true:

- CMYK or K is an optional prefix that indicates to SAS that this is a CMYK color specification.
- *cc* is the cyan component.
- *mm* is the magenta component.
- *yy* is the yellow component.
- *kk* is the black component.

The color-value components are hexadecimal numbers in the range 00–FF, where higher values are darker and lower values are brighter. This scheme allows for up to 256 levels of each color component. When the color value begins with a letter (A–F), you can omit the CMYK or K prefix. When the color value begins with a number (0–9), you must use the CMYK or K prefix, or enclose the value in quotation marks.

Table 23.4 *Examples of CMYK Color Values*

Color	CMYK Value
Red	"00FFFF00"
Green	FF00FF00
Blue	CMYKFFFF0000
White	"00000000"
Process black (using cyan, magenta, and yellow ink)	KFFFFFF00
Pure black (using only black ink)	"000000FF"

Note: You can specify a CMY value by specifying zero (00) for *kk*, the color's black component.

CMYK color specifications are for output devices that support four colors. If you specify CMYK colors on an output device that supports three colors such as RGB, the CMYK colors are converted to colors in the three-color space. Because the four-color space supports many more colors than a three-color space, the CMYK colors might map to different colors in the three-color space. To preserve your CMYK colors in that case, specify a device that supports the CMYK color space. Some of the SAS universal printers support CMYK colors. For more information, see "Color Support for Universal Printers" in *SAS 9.4 Universal Printing*.

HLS Color Codes

The HLS color scheme specifies colors in terms of hue, lightness, and saturation levels. It is based on the Tektronix Color Standard. HLS color names are of the form H*hhhllss*, where the following is true:

- H indicates to SAS that this is an HLS color specification.
- *hhh* is the hue component, which is expressed as an angle.
- *ll* is the lightness component.
- *ss* is the saturation component.

The components are hexadecimal numbers. The hue component is in the range 000–168 hexadecimal, which represents an angular value in the range 0–360 decimal. Hue starts with the primary color blue at 0 degrees, and then progresses to red at 120 degrees, to green at 240 degrees, and back to blue at 360 degrees. Both the lightness and saturation components are in the range 00–FF hexadecimal (0–255 decimal), which provides 256 levels that represent 0% to 100% for each component.

Table 23.5 *Examples of HLS Color Codes*

Color	HLS Color Code
Red	H07880FF
Green	H0F080FF
Blue	H00080FF
Light gray	H000BB00
White[1]	H*xxx*FF00, such as H000FF00
Black[1]	H*xxx*0000 such as H0000000

1 When the saturation is set to 00, the color is a shade of gray that is determined by the lightness value. Therefore, white is defined as H*xxx*FF00 and black as H*xxx*0000, where *xxx* can be any hue.

HSV (or HSB) Color Codes

The HSV color-naming scheme specifies colors in terms of hue, saturation, and value (or brightness) components. HSV color names are of the form V*hhhssvv*, where the following is true:

- V indicates to SAS that this is an HSV color specification.
- *hhh* is the hue component, which is expressed as an angle.
- *ss* is the saturation component.
- *vv* is the value or brightness component.

The components are hexadecimal numbers. The hue component is in the range 000–168 hexadecimal, which represents an angular value in the range 0–360 decimal. Hue starts with the primary color red at 0 degrees, and then progresses to green at 120 degrees, to blue at 240 degrees, and back to red at 360 degrees. Both the saturation and value (brightness) components are in the range 00–FF hexadecimal (0–255 decimal), which provides 256 levels for each component.

Table 23.6 *Examples of HSV (or HSB) Color Codes*

Color	HSV Color Code
Red	V000FFFF
Green	V078FFFF
Blue	V0F0FFFF
Light gray[1]	V*xxx*00BB such as V07900BB

Color	HSV Color Code
White[1]	V*xxx*00FF such as V07900FF
Black[1]	V*xxx*00000 such as V0790000

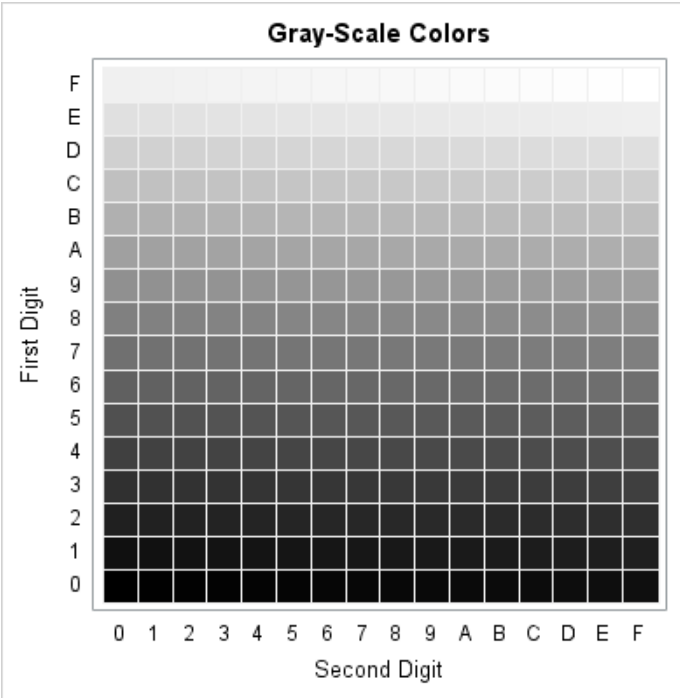
1 When the saturation is set to 00, the color is a shade of gray. The value component determines the intensity of gray level. The *xxx* value can be any hue.

Gray-Scale Color Codes

Gray-scale colors are specified using the word GRAY and a lightness value in the form GRAY*hh*. The value *hh* is the lightness of the gray and is a hexadecimal number in the range 00–FF, which provides 256 levels on the gray scale.

Note: GRAY, without a lightness value, is a SAS color name defined in the SAS registry. (See "SAS Color Names and RGB Values in the SAS Registry" on page 376.) Its value is CX808080. Invalid color specifications are mapped to GRAY.

The following figure shows the gray-scale color for each hexadecimal value.



Color Naming System (CNS) Values

For CNS, you specify a color value by specifying lightness, saturation, and hue, in that order, using the terms shown in the following table.

Table 23.7 *Color Naming System Values*

Lightness	Saturation	Hue
Black	Gray	Blue
Very Dark	Grayish	Purple
Dark	Moderate	Red
Medium	Strong	Orange/Brown
Light	Vivid	Yellow
Very Light		Green
White		

Follow these rules when you are determining the CNS color name:

- You should not use the lightness values Black and White with saturation or hue values.
- If you do not specify default values, medium is the default lightness value and vivid is the default saturation value.
- Gray is the only saturation value that can be used without a hue.
- Unless the color that you want is black, white, or some form of gray, you must specify at least one hue.

You can use one or two hue values in the CNS color name. When you use two hue values, the hues must be adjacent to each other in the following list: blue, purple, red, orange/brown, yellow, green, and then returning to blue. Two hue values result in a color that is a combination of both colors. Use the suffix “ish” to reduce the effect of a hue when two hues are combined. Reddish purple is less red than red purple. The color specified with the “ish” suffix must precede the color without the “ish” suffix.

You can write color names in the following ways:

- without space separators between words
- with an underscore to separate words
- with a space to separate words, enclosed in quotation marks

For example, the following color specifications are valid:

- verylightmoderatepurplishblue

- very_light_moderate_purplish_blue
- "very light moderate purplish blue"

Note: If a CNS color name is also a color name in the SAS Registry, the SAS Registry color value takes precedence. Some CNS color names and color names in the SAS Registry have different color values. Dark blue is an example.

To use a CNS color value when the color name is also in the SAS Registry, do one of the following:

- Include a space to separate the words and enclose the entire color name in quotation marks. Here is an example:

```
color="dark blue"
```

- Include an underscore to separate the words. Here is an example:

```
color=dark_blue
```

SAS Color Names and RGB Values in the SAS Registry

The SAS Registry provides a set of color names and RGB values that you can use to specify colors. These color names and RGB values are common to most web browsers. You can specify the name itself or the RGB value associated with that color name. To view the color names as associated RGB values that are defined in the registry, submit the following code:

```
proc registry list
 startat="COLORNAMES";
run;
```

SAS prints the output in the SAS log. Here is a partial listing.

```
NOTE: Contents of SASHELP REGISTRY starting at subkey [COLORNAMES]
[  COLORNAMES]
  Active="HTML"
[    HTML]
    AliceBlue=hex: F0,F8,FF
    AntiqueWhite=hex: FA,EB,D7
    Aqua=hex: 00,FF,FF
    Aquamarine=hex: 7F,FD,D4
    Azure=hex: F0,FF,FF
    Beige=hex: F5,F5,DC
    ...
```

For a list of the color names that are defined in the SAS Registry, see “Predefined Colors” on page 1683.

You can also create your own color values by adding them to the SAS registry. For information about viewing and modifying the list of color names, see “The SAS Registry” in *SAS Programmer’s Guide: Essentials*.

Converting Color Values between Color-Naming Schemes

The SAS/GRAPH software provides several macros that enable you to perform selected conversions between color-naming schemes. The following table shows the conversions that are possible using these macros.

From Color Value	To Color Value	Conversion Macro
CMY	RGB	%CMY
CNS	HLS	%CNS
RGB	HLS	%RGB2HLS
HLS	RGB	%HLS2RGB

For information about these macros, see "Graphics Options and Device Parameters Dictionary" on page 577.

Converting Numeric Color Component Values to Color Names

The SAS/GRAPH software provides macros that enable you to perform selected conversions between color-naming schemes. The following table shows the macros that you can use for this purpose.

From Numeric Color Component Values	To Color Name	Conversion Macro
cyan, magenta, yellow, black	CMYK	%CMYK
hue, lightness, saturation	HLS	%HLS
hue, saturation, value	HSV	%HSV
red, green, blue	RGB	%RGB

Converting Color Values from Other Applications

Many software programs enable you to change or customize various colors. A dialog box typically provides a means of selecting a different color or modifying the attributes of an existing color. In the SAS ODS Graphics Editor, for example, the More Colors dialog box shown in the following figure serves this purpose.

Figure 23.1 *More Colors Dialog Box*

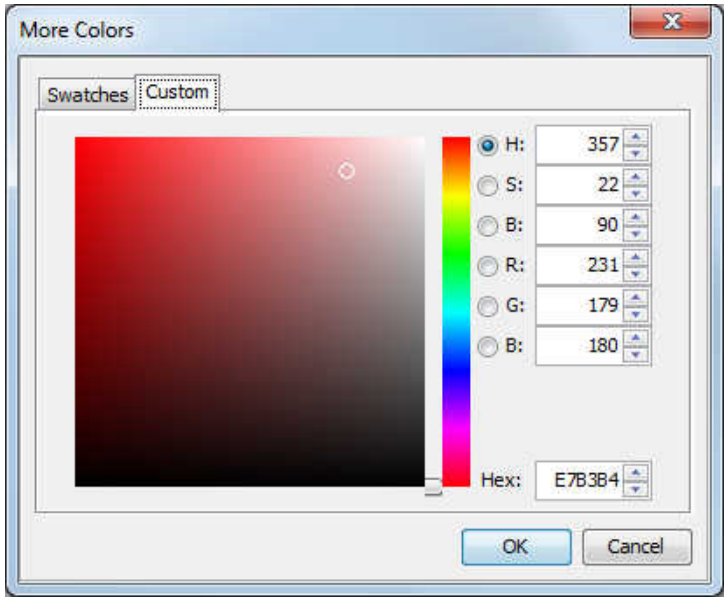


Figure 23.1 on page 378 shows the HSB and RGB numeric color component values for the currently selected color. It also shows the RGB values in hexadecimal. To use the RGB specification for this color in a SAS program, add the CX prefix to the **Hex** value E7B3B4. If the application provides only the numeric component values, you must convert the decimal component values to hexadecimal. In Figure 23.1 on page 378, the **H** value is in the range 0–360 degrees, and the **S** and **B** values are each in the range 0–100 percent. The **R**, **G**, and **B** values are in the range 0–255 each. You can convert the component values manually. (See "Understanding Hexadecimal Values" on page 1721.) Be aware that you must first convert the **S** and **B** values from percentages to 255-based values by rounding the result of the following computation to the nearest integer:

$$\left(\frac{Value}{100}\right) \times 255$$

You can also use the SAS/GRAPH color utility macros to convert the values. The %HSV color utility macro converts HSV (HSB) numeric color component values to an HSV color name. Likewise, the %RGB macro converts RGB numeric color component values to an RGB color name. The following example shows how to use the %HSV color utility macro to convert the **H**, **S**, and **B** color component values in Figure 23.1 on page 378 to an HSV color name.

```
%COLORMAC;
data _null_;
  put "%HSV(357,22,90)";
```

```
run;
```

Note: The %COLORMAC macro compiles all of the SAS/GRAPH color utility macros. You need to run it only once during a SAS session.

Because the %HSV macro accepts values in the range 0–100 as a percentage of 255 for saturation and value, use the **S** and **B** values as shown. The result is V16538E6.

The following example shows how to use the %RGB color utility macro to convert the **R**, **G**, and **B** numeric color component values in Figure 23.1 on page 378 to an RGB color name.

```
/* Compute the RGB percentages */
data _null_;
  r = 231;
  g = 179;
  b = 180;
  call symputx("r", round((r/255)*100));
  call symputx("g", round((g/255)*100));
  call symputx("b", round((b/255)*100));
run;

/* Convert to RGB color name */
%COLORMAC;
data _null_;
  put "%RGB(&r,&g,&b)";
run;
```

Because the %RGB color utility macro accepts integer values in the range 0–100 as a percentage of 255, you must convert the 255-based values to integer percentages in order to use them in the %RGB macro call. The result is CXE8B3B5. The result from the %RGB macro is not exact due to rounding. If you want more exact results, use the following program.

```
data _null_;
  r = 231;
  g = 179;
  b = 180;
  rgb="CX" || put(r,hex2.) || put(g,hex2.) || put(b,hex2.);
  put rgb;
run;
```

The result is CXE7B3B4.

To convert the RGB color name to an HLS color name, use the %RGB2HLS macro as shown in the following program.

```
%COLORMAC;
data _null_;
  put "%RGB2HLS(CXE7B3B4)";
run;
```

The result is H077CD84.

For more information about the color utility macros, see “Color Utility Macro Reference” on page 1697.

Using the Color Utility Macros

The color utility macros enable you to define colors for a specific color-naming scheme and convert color values between color-naming schemes. These macros are useful when specifying colors in your SAS/GRAPH programs. Table 23.8 on page 380 provides a summary of the color utility macros that are available. A complete description is in "Color Utility Macro Reference" on page 1697.

Table 23.8 *Color Utility Macros*

Macro	Description
%COLORMAC	Creates the color utility macros in the current SAS session. You must run this macro in your current SAS session before you can use any of the macros that are listed in this table.
%HELPCLR	Writes Help information for the color utility macros to the SAS log.
%CMY	Creates an RGB color name from numeric CMY component values.
%CMYK	Creates a CMYK color name from numeric CMYK component values.
%CNS	Creates a SAS/GRAPH HLS color name from a color-naming scheme (CNS) color name.
%HLS	Creates an HLS color name from numeric HLS component values.
%HSV	Creates an HLS color name from numeric HSV component values.
%RGB	Creates an RBG color name from numeric RGB component values.
%HLS2RGB	Creates an RGB color name from an HLS color name.
%RGB2HLS	Creates an HLS color name from an RGB color name.

Processing Limitations for Colors

Note: Using colors in SAS/GRAPH is limited by the number of colors that you can use in one graph and by the capabilities of your device.

Maximum Number of Colors Displayed on a Device

The number of colors that you can display is limited by the graphics output device. If you create a graph with more colors than the device can display, the colors are mapped to an existing color for display. You might also receive a note in the SAS log telling you when a color is mapped to another color, along with the name of the replacement color.

If your device can support 16 million colors, it might not let you use all of them at once. The MAXCOLORS device parameter tells SAS/GRAPH the maximum number of colors that it can display simultaneously. MAXCOLORS is the number of foreground colors plus the background color. If you use more than the number of colors set by the MAXCOLORS device parameter, the excess colors are remapped. This can occur when a photograph is used as the background image for a graph or when gradient shading is used in a 3-D graph. In that case, use the MAXCOLOR device parameter to increase the maximum number of colors. See “MAXCOLORS Device Parameter” on page 659.

Note: The MAXCOLORS device parameter defaults to the number of colors that the basic model of each graphics device supported can display. If your graphics device can display more colors than the base model, use the PENMOUNTS= graphics option to specify the number of colors your graphics device can display. You can also use the GDEVICE procedure to modify the value of the MAXCOLORS device parameter.

Replaying Graphs on a Device That Displays Fewer Colors

You can use the GREPLAY procedure to display previously created graphs. Sometimes you might need to replay the graphs on a device that cannot display as many colors as the device on which the graph was originally developed. Use the CMAP statement (see “CMAP Device Parameter” on page 596) to control some of the remapping.

When you replay graphs on devices that display fewer colors than are in the graph, two situations can cause problems:

- Colors are specified that the device does not support.
- More colors are specified than the device can display at one time.

If you specify colors on a device that does not support the colors requested, the colors are remapped to gray. A note is issued to the SAS log telling you when a color is mapped gray.

The number of colors that your device can display affects the actual colors displayed. If your graphics output device can create a maximum of 64 distinct colors, and your graph contains 256 colors, then the 65th through the 256th color specifications are remapped to the colors specified in the current style. If the NOGSTYLE system option is in effect, the colors are remapped to the device's available colors and might not be displayed as the color that you specify.

You can use the TARGETDEVICE= graphics option to preview how a graph is going to look on a different device. Set the device entry name of the device driver to this graphics option. The graph is displayed as close as possible to the display when the other device is used.

Note: When you use the TARGETDEVICE= graphics option, SAS/GRAPH uses the color list of the target device as the default color list; any color that you explicitly use is displayed when you preview the graph, although the color might be mapped by the target device. Refer to "TARGETDEVICE Graphics Option" on page 700 for complete information about the TARGETDEVICE= graphics option.

24

Adding Images to SAS/GRAPH Output

Introduction

By default, the appearance of SAS/GRAPH output is determined by the current ODS style. Some styles add an image to your graph, but most do not. You can use SAS/GRAPH statements and options to add images to your graphs as needed. In SAS 9.4M7 and earlier releases, you can place an image in the background area of a graph, in the backplane of graphs that support frames, on the bars of two-dimensional bar charts, or on the bars of three-dimensional bar charts when using the ACTIVEX device or the ACTXIMG device. You can also apply images at specified graph-coordinate positions using the Annotate facility. The images that you add to your graphs can be SAS files or external files, in a range of image formats.

Image File Types Supported by SAS/GRAPH

For displaying images in your graphs, SAS/GRAPH supports the image file types shown in the following table.

Table 24.1 *Image File Types Supported by SAS/GRAPH*

File Type	Description
BMP (Microsoft Windows Device Independent Bitmap)	Supports color-mapped and true color images stored as uncompressed or run-length encoded. BMP was developed by Microsoft Corporation for storing images under Windows 3.0.
DIB (Microsoft Windows Device Independent Bitmap)	See the description of BMP.
GIF (Graphics Interchange Format)	Supports only color-mapped images. GIF is owned by CompuServe, Inc.
JPEG (Joint Photographic Experts Group)	Supports compression of images with the use of JPEG File Interchange Format (JFIF) software. JFIF software is developed by the Independent Joint Photographic Experts Group.
PBM (Portable Bitmap Utilities)	Supports gray, color, RGB, and bitmap files. The Portable Bitmap Utilities is a set of free utility programs that were primarily developed by Jeff Poskanzer.
PCD (Photo CD)	Kodak Photo CD format, which supports multiple image resolutions.
PCX (PC Paintbrush)	Supports bitmap, color-mapped, and true color images. PCX and PC Paintbrush are owned by Zsoft Corporation.
PNG (Portable Network Graphic)	Supports truecolor, gray-scale, and 8-bit images.
SVG (Scalable Vector Graphics)	Is an XML language for describing two-dimensional vector graphics. SVG

File Type	Description
	supports true color images and transparency.
TGA (Targa)	Supports true color images. Targa is owned by Truevision, Inc.
TIFF (Tagged Image File Format)	Internally supports a number of compression types and image types, including bitmap, color-mapped, gray-scale, and true color. TIFF was developed by Aldus Corporation and Microsoft Corporation.
XBM (X Window Bitmaps)	Supports bitmap images only. XBM is owned by MIT X Consortium.
XWD (X Window Dump)	Supports all X visual types (bitmap, color-mapped, and true color.) XWD is owned by MIT X Consortium.

Displaying an Image in a Graph Background

To place an image on the graph background, use the IBACK= option in a GOPTIONS statement. Specify either the path to the image file in quotation marks or a fileref that has been defined to point to the image file as follows:

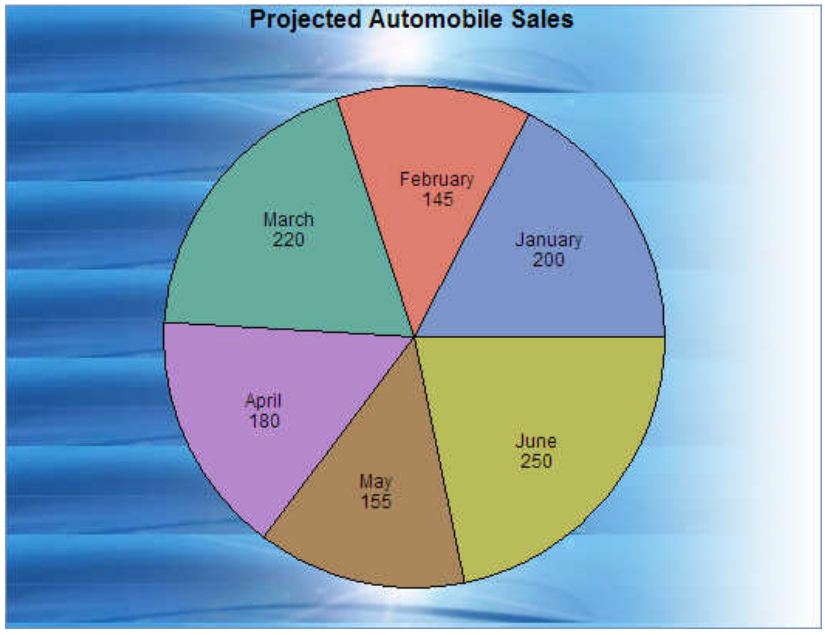
```
goptions iback="external-image-file" | fileref;
```

For example, the following program creates a pie chart with a background image:

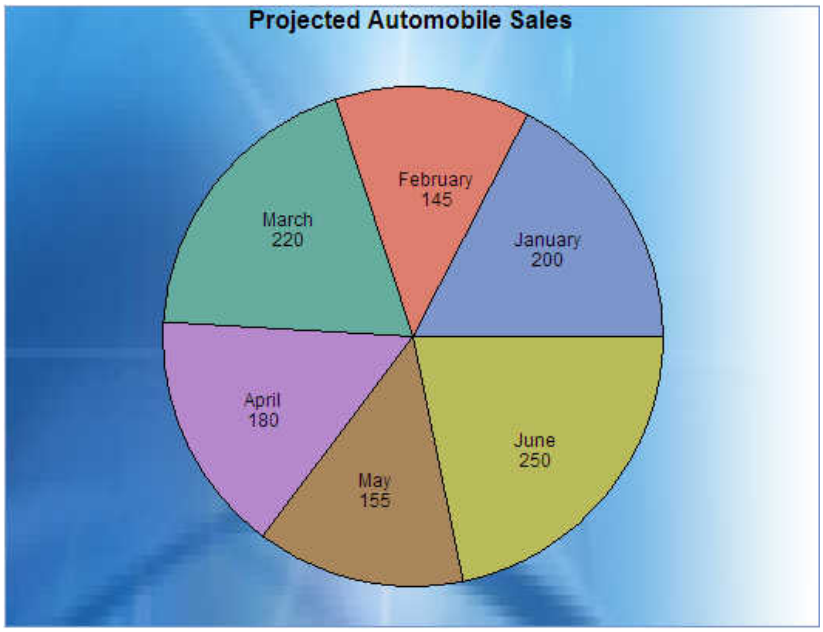
```
goptions reset=all
          htitle=1.25
         colors=(cx7c95ca cxde7d6f  cx66ada0
                 cxb689cd cxa9865b cxbabc5c)
         iback="external-image-file";
title "Projected Automobile Sales";
data sales;
    input Month Amount;
    informat month monyy.;
   datalines;
jan08 200
feb08 145
mar08 220
apr08 180
may08 155
```

```
jun08 250
;
proc sort;
   by month;
proc gchart;
   format month monname8.;
   pie month / discrete freq=amount value=inside
               noheading coutline=black;
run;
quit;
```

Because the default value for the IMAGESTYLE= graphics option is TILE, the image is copied as many times as needed to fill the background area.



You can specify IMAGESTYLE=FIT in the GOPTIONS statement to stretch the image so that a single image fits within the entire background area.



Displaying an Image in Graph Frame

Procedure action statements that support the IFRAME= support frames, which are the backplanes behind the graphs. The backplane is the area within the graph axes. To place an image on the backplane of a graph, specify the IFRAME= option in the procedure action statement that generates the graph. On the IFRAME= option, specify either the path to the image file in quotation marks or a fileref that has been defined to point to the image file:

```
iframe=fileref | "external-image-file";
```

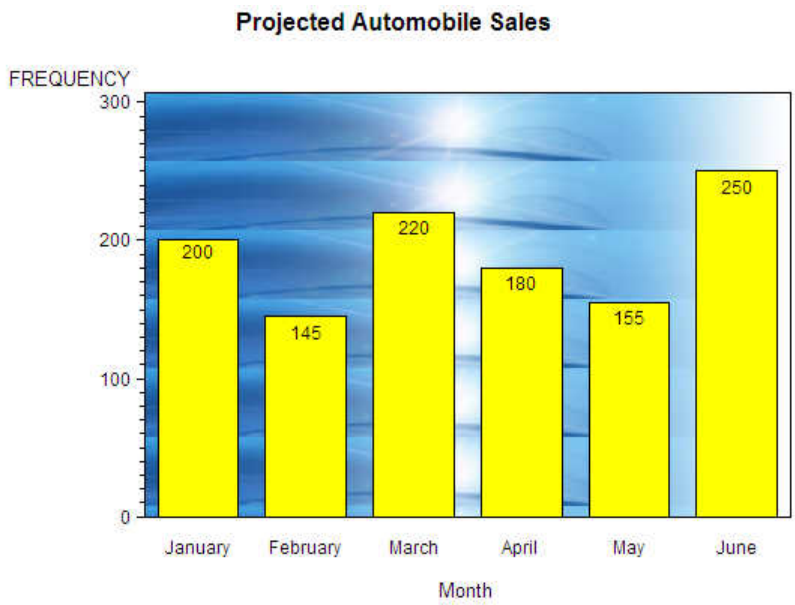
Note: If you specify the NOFRAME option in the procedure action statement or if you specify STYLE=0 in the AXIS statement, the IFRAME= option is ignored.

For example, the following program creates a vertical bar chart and adds an image to the graph frame:

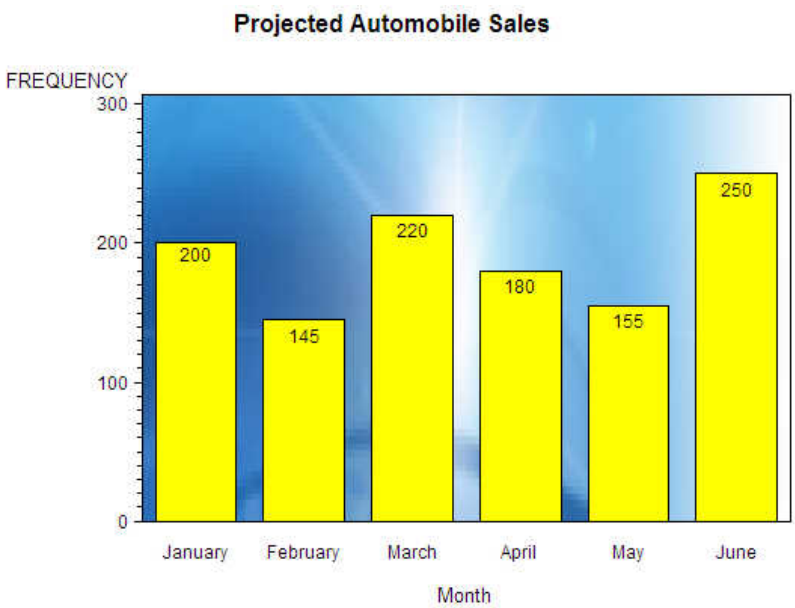
```
goptions reset=all htitle=1.25 colors=(yellow cxde7d6f);
title "Projected Automobile Sales";
data sales;
    input Month Amount;
    informat month monyy.;
   datalines;
jan08 200
feb08 145
mar08 220
apr08 180
may08 155
jun08 250
;
```

```
proc sort;
   by month;
proc gchart;
   format month monname8.;
   vbar month / discrete freq=amount inside=freq
                coutline=black
iframe="external-image-file";
run;
quit;
```

Because the default value for the GPLOT procedure's IMAGESTYLE= option is TILE, the image is copied as many times as needed to fill the frame area.



You can specify IMAGESTYLE=FIT in the GOPTIONS statement to stretch the image so that a single image fits within the entire frame area.



Displaying Images on Data Elements

You can place images on the bars in two-dimensional bar charts generated by the GCHART HBAR or VBAR statements. You can also place images on the bars of three-dimensional bar charts if you are using the ACTIVEX device or the ACTXIMG device.

In the IMAGE= option of the PATTERN statement, specify either the path to the image file in quotation marks or a fileref that has been defined to point to the image file.

```
pattern image=fileref | "external-image-file";
```

By default, the image is tiled on the bar, which means that the image is copied as many times as needed to fill each bar. Specify IMAGESTYLE=FIT in the PATTERN statement to stretch the image as needed to fill each bar.

```
   pattern image="external-image-file"
   imagestyle=fit;
```

To tile subsequent images, reset the PATTERN statement or by specify IMAGESTYLE=TILE.

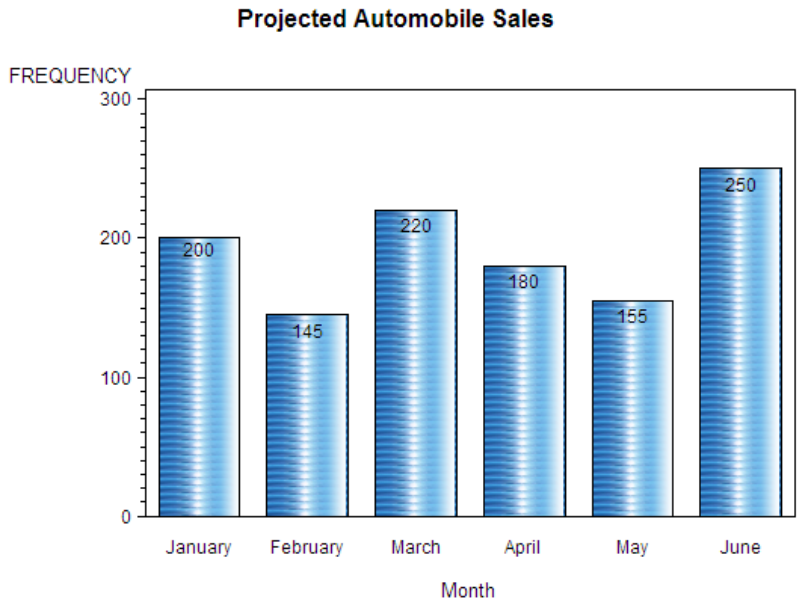
Note: Images are supported on bar charts generated by the HBAR and VBAR statements. If an image is specified in a PATTERN statement that is used with another type of chart, then the PATTERN statement is ignored and default pattern rotation is affected. If you submit a PIE statement when an image has been specified in the PATTERN= option, the default fill pattern is used for the pie slices. Each pie slice displays the same fill pattern.

The following example places an image on the bars of a vertical bar chart:

```
goptions reset=all htitle=1.25 colors=(yellow cxde7d6f);
title "Projected Automobile Sales";
data sales;
    input Month Amount;
    informat month monyy.;
   datalines;
jan08 200
feb08 145
mar08 220
apr08 180
may08 155
jun08 250
;
proc sort;
   by month;
pattern1 image="external-image-file";
proc gchart;
   format month monname8.;
   vbar month / discrete freq=amount inside=freq
               coutline=black;
 run;
```

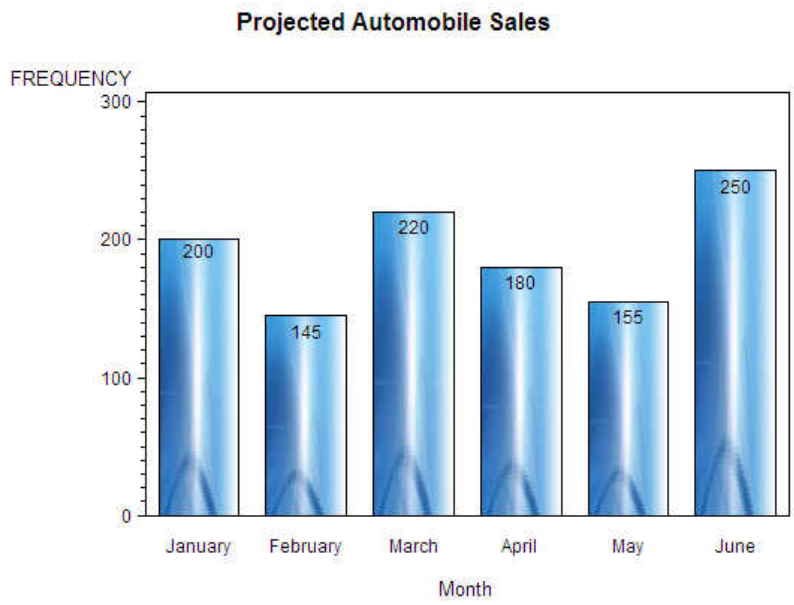
```
quit;
```

The image is tiled to fill each bar.



If the PATTERN IMAGESTYLE=FIT option is used, the image is stretched to fill each bar.

```
pattern=fileref | "external-image-file" imagestyle=fit;
```



Displaying Images Using Annotate

The Annotate facility enables you to display an image at the coordinate location that you specify with the X and Y variables. To display an image, do the following:

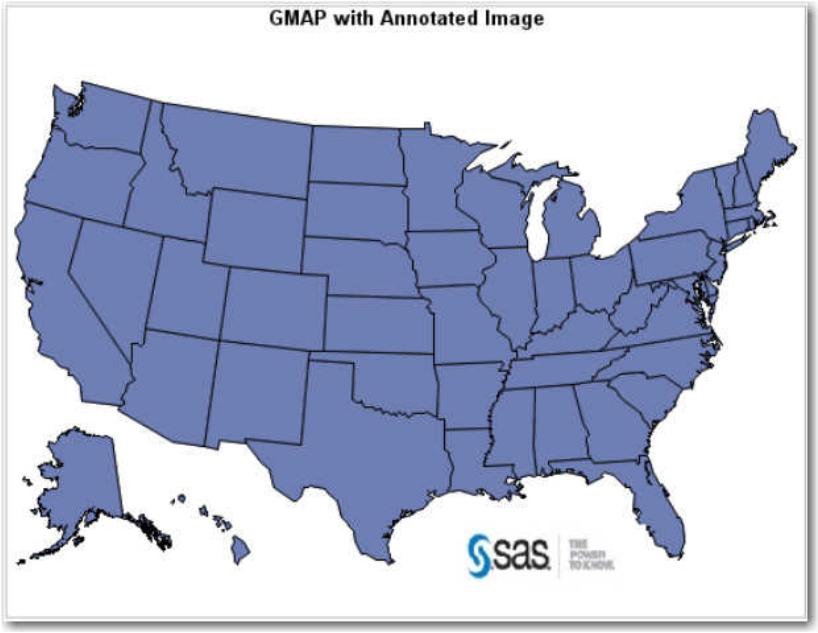
- Specify the image file in quotation marks on the IMGPATH variable.
- Set the image coordinates with the X and Y variables.
- Specify the IMAGE function.

One corner of the image is located by the current X and Y position. The opposite corner is located by the X and Y variables associated with the IMGPATH variable. Here is an example that uses the Annotate facility to place the SAS logo on a map of the United States. You must specify the path to the image file before running the code.

Note: The following program uses the GMAP procedure, which is described in *SAS/GRAPH and Base SAS: Mapping Reference*.

```
goptions reset=all border htitle=1.75cells
    hsize=5.5in vsize=4.2in;
data my_anno;
   length function $8;
   xsys="3";  ysys="3"; when="a";
function="move";  x=55;  y=5;  output;
function="image";  style="fit";  imgpath="external-image-file";
         x=x+25;  y=y+10;  output;
run;
title1 "GMAP with Annotated Image";
proc gmap data=mapsgfk.us map=mapsgfk.us anno=my_anno;
   id state;
   choro state/
   levels=1
   nolegend
   statistic=freq;
run;
quit;
```

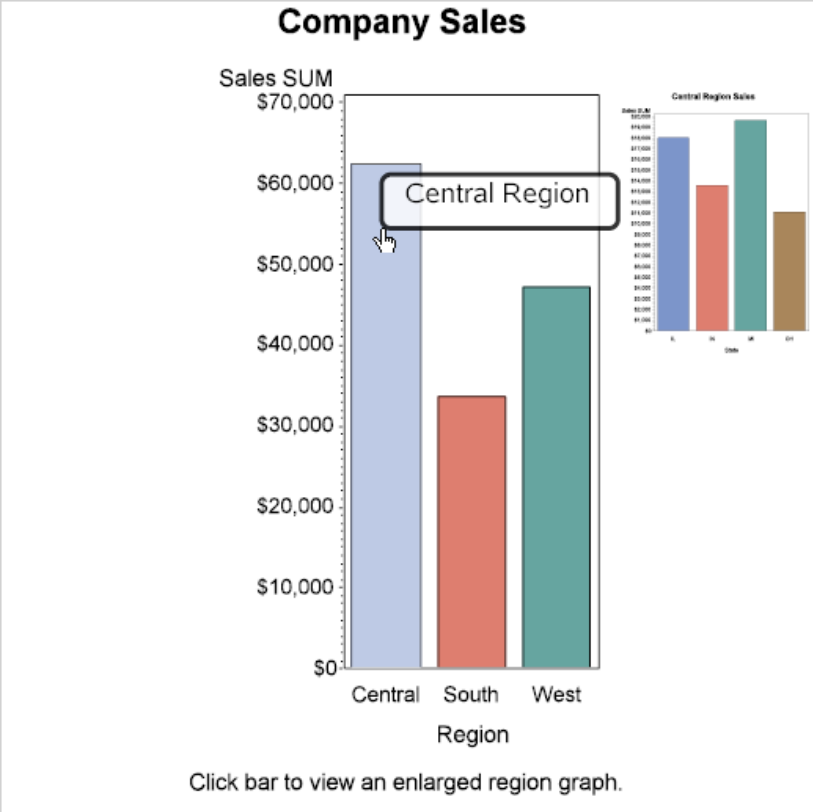
The `style="fit"` variable on the IMAGE function stretches the image as needed to fill the area.



To tile the image to fill the area where the logo is positioned, set the STYLE variable equal to `"tile"`. Starting with SAS 9.4M5, you can specify STYLE=`"single"` to center a single instance of the image on the specified coordinates.

Displaying Pop-Up Images Using the SVG Graphics Devices

When you use an SVG graphics device, the HTML= option enables you to display an image in a specific area of a graph when the mouse pointer is positioned on a linked shape. The pop-up image is displayed only as long as the mouse pointer is positioned on the linked shape. The following figure shows an example of a drill-down bar chart that displays a pop-up preview image of the Central Region Sales drill-down chart when the mouse pointer is positioned on the Central Region bar.



The pop-up images appear in the top right corner of the graphics output area. A pop-up image is displayed only as long as the mouse pointer is positioned on a bar. It disappears when the mouse pointer is moved off of the bar. The pop-up image can be placed anywhere in the graphics output area and can be sized as needed.

For more information about creating pop-up images using the SVG graphics devices, see "Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" on page 262.

Disabling and Enabling Image Output

The NOIMAGEPRINT graphics option disables image output without removing code from your SAS/GRAPH program. Also, some ODS styles (such as Dtree, Netdraw, and BlockPrint) include background images, but not all devices support these images. You can suppress the images with the NOIMAGEPRINT option.

```
goptions noimageprint;
```

Note: Starting with SAS 9.4M5, the BlockPrint ODS style is removed from suggested use. This style still functions when used in SAS programs, but it no longer appears in the list of available styles.

To enable image output, reset the GOPTIONS statement or specify the IMAGEPRINT graphics option.

```
goptions imageprint;
```

PART 5

Statements and Options Used by SAS/GRAPH

25

SAS/GRAPH Statements

Overview of Global Statements

SAS/GRAPH programs can use some of the SAS language statements that you typically use with the Base SAS procedures or with the DATA step, such as LABEL,

WHERE, and FORMAT. These statements are described in the *SAS DATA Step Statements: Reference*.

In addition, SAS/GRAPH has its own set of statements that affect only graphics output generated by the SAS/GRAPH procedures and the SAS/GRAPH Annotate facility. Most of these statements are *global statements*. That is, they can be specified anywhere in your program and remain in effect until explicitly changed, overridden, or canceled. These are the SAS/GRAPH global statements:

AXIS
: modifies the appearance, position, and range of values of axes in charts and plots.

FOOTNOTE
: adds footnotes to graphics output. This statement is like the TITLE statement and is described in that section.

GOPTIONS
: submits graphics options that control the appearance of graphics elements by specifying characteristics such as colors, fill patterns, fonts, or text height. Graphics options can also temporarily change device settings.

LEGEND
: modifies the appearance and position of legends generated by procedures that produce charts, plots, and maps.

NOTE
: adds text to the graphics output. This statement is an exception because it is not global but local, meaning that it must be submitted within a procedure. Otherwise, the NOTE statement is like the TITLE statement and is described in that section. When including text in a NOTE in output from the GMAP procedure, it is displayed behind the polygons of a map. The NOTE statement is ignored when using the SGMAP procedure.

PATTERN
: controls the color and fill of patterns assigned to areas in charts, maps, and plots.

SYMBOL
: specifies the shape and color of plot symbols as well the interpolation method for plot data. It also controls the appearance of lines in contour plots.

TITLE
: adds titles to graphics output. The section describing the TITLE statement includes the FOOTNOTE and NOTE statements.

The above statements are described in this chapter, as well as the following two Base language statements that have a special effect when used with SAS/GRAPH procedures:

BY
: processes data according to the values of a classification (BY) variable and produces a separate graph for each BY-group value. This statement is not a global statement. It must be specified within a DATA step or a PROC step.

ODS HTML
: generates one or more files written in Hypertext Markup Language (HTML). If you use it with SAS/GRAPH procedures or Base SAS mapping procedures, you can specify one of the device drivers, such as GIF, PNG, or SVG. With the GIF device driver, the graphics output is stored in GIF files. The HTML files that are generated reference the graphics output. When viewed with a web browser, the

HTML files can display graphics and non-graphics output together on the same web page.

For more information about the BY, LABEL, and WHERE statements in Base SAS software, see *SAS DATA Step Statements: Reference*. For more information about the OPTIONS statement, see *SAS Global Statements: Reference*.

Specifying Units of Measurement

When the syntax of an option includes *units*, use one of these:

CELLS
: character cells

CM
: centimeters

IN
: inches

PCT
: percentage of the graphics output area

PT
: points

If you omit *units*, a unit specification is derived in this order:

1. The GUNIT= option in a GOPTIONS statement.
2. The default unit, CELLS.

Dictionary

AXIS Statement

Controls the location, values, and appearance of the axes in plots and charts.

Used by:	GBARLINE, GCHART, GCONTOUR, GPLOT, GRADAR, G3D
Type:	Global
Restrictions:	For the G3D procedure, the AXIS statement is supported by the ActiveX device only. For the GCHART procedure, the AXIS statement and its options are ignored when used in conjunction with the BLOCK statement.
Video:	Watch the video: “Control the Graph Axis Using the Axis Statement”.

Syntax

AXIS<1 …99> <*options*>;

Summary of Optional Arguments

Appearance options

COLOR=*axis-color*
: specifies the color for all axis components (the axis line, all tick marks, the frame outline, and all text) unless you include a more explicit AXIS statement color specification.

LENGTH=*axis length <units >*
: specifies the length of the axis in number of units.

NOBRACKETS
: suppresses the printing of group brackets drawn around the values on the group axis in a bar chart.

NOPLANE
: removes either the horizontal or vertical three-dimensional axis plane in bar charts produced by the HBAR3D and VBAR3D statements.

OFFSET=(<*n1*><,*n2*>)<*units* > | (<*n1*<*units*>><,*n2*<*units*>>)
: specifies the distance from the first and last major tick marks or bars to the ends of the axis line.

ORIGIN=(<*x*><,*y*>)<*units*> | (<*x*<*units*>><,*y*<*units*>>)
: specifies the *x* coordinate and the *y* coordinate of the origin of the axis.

STAGGER
: offsets the axis values on a horizontal axis.

STYLE=*line-type*
: specifies a line type for the axis line.

WIDTH=*thickness-factor*
: specifies the thickness of the axis line.

Axis scale options

INTERVAL=EVEN | UNEVEN | PARTIAL
: The INTERVAL option affects the LOGBASE option in the AXIS statement.

LOGBASE=*base* | E | PI
: scales the axis values logarithmically according to the value specified.

LOGSTYLE=EXPAND | POWER
: specifies whether the values displayed on the logarithmic axis are the values of the base or the values of the power.

ORDER=(*value-list*)
: specifies the order in which data values appear on the axis.

Text options

LABEL=(*text-argument(s)*) | NONE
: modifies an axis label.

REFLABEL=(*text-argument(s)*) | NONE
: creates and defines the appearance of a reference-line label.

SPLIT="*split-char(s)*"

specifies one or more characters that the AXIS statement uses to break axis values into multiple lines.

VALUE=(*text-argument(s)*) | NONE
modifies the major tick mark values.

Tick mark options

MAJOR=(*tick-mark-suboptions*) | NONE
modifies the major tick marks.

MINOR=(*tick-mark-suboptions*) | NONE
modifies the minor tick marks that appear between major tick marks.

Optional Arguments

COLOR=*axis-color*
specifies the color for all axis components (the axis line, all tick marks, the frame outline, and all text) unless you include a more explicit AXIS statement color specification.

The following table lists the SAS/GRAPH statement options that can be used to override the COLOR= specification. The table also lists the name of the style reference associated with each of the options.

Table 25.1 *SAS/GRAPH Statement Options per Axis Component*

Option	Graph Element	Style Reference
AXIS statement:		
LABEL= (COLOR=*color*)	Axis label	GraphLabelText
REFLABEL= (COLOR=*color*)	Reference-line labels	
VALUE= (COLOR=*color*)	Major tick mark values	GraphValueText
calling procedure:		
CTEXT=	All axis text (AXIS label and major tick mark value descriptions)	GraphLabelText
CAXIS=	Axis line, axis frame outline, and major minor tick marks	GraphAxisLines

If you omit all color options, the AXIS statement looks for a color specification in this order:

1. the CTEXT= graphics option in a GOPTIONS statement
2. the color of all axis components is the color of the default style

Alias C=

Example "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

INTERVAL=EVEN | UNEVEN | PARTIAL

The INTERVAL option affects the LOGBASE option in the AXIS statement. Specifying the options INTERVAL=UNEVEN and LOGBASE=10 permits non-base-10 values to be specified for the ORDER option, while retaining a logarithmic scale for the axis.

Restriction Not supported by ActiveX

Note PARTIAL is an alias for UNEVEN. They have the same effect.

LABEL=(*text-argument(s)*) | NONE

modifies an axis label. *Text-argument(s)* defines the appearance or the text of an axis label, or both. NONE suppresses the axis label. *Text-argument(s)* can be one or more of these:

"text-string"

provides up to 256 characters of label text. By default, the text of the axis label is either the variable name or a previously assigned variable label. Enclose each string in quotation marks. Separate multiple strings with blanks.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVAL*n*
#BYVAL(*BY-variable-name*)

substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n

specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name

names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

#BYVAR*n*
#BYVAR(*BY-variable-name*)

substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the

text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 423

text-description-suboption
modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be any of the following:

- ANGLE=*degrees*
- COLOR=*text-color*
- FONT=*font* | NONE
- HEIGHT=*text-height* <*units* >
- JUSTIFY=LEFT | CENTER | RIGHT
- ROTATE=*degrees*

See "Text Description Suboptions" on page 416 for a complete description of these suboptions.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Style reference Color attribute of the GraphLabelText style element

Restrictions Partially supported by ActiveX

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. This is because BY lines are not created in a DATA step.

Examples "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

"Example 2: Specifying Logarithmic Axes" on page 531

"Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

LENGTH=*axis length* <*units* >
specifies the length of the axis in number of units. If you request a length that cannot fit the display, a warning message is written to the log and your graph might produce unexpected results.

Style reference Color attribute of the GraphLabelText graph element

Restriction Not supported by the GRADAR Procedure

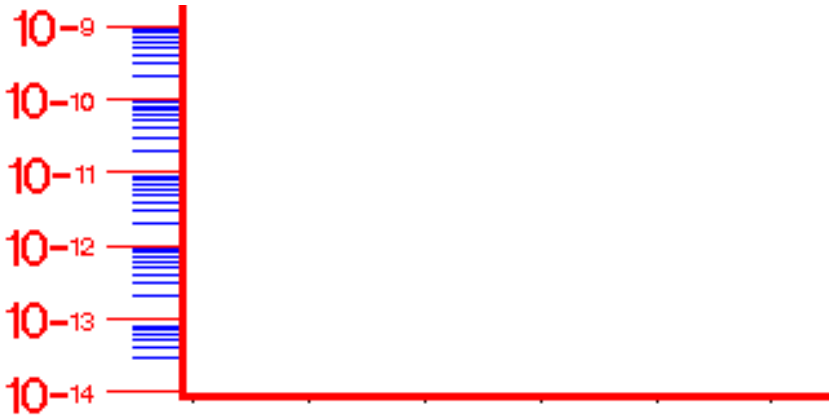
Examples "Example 2: Specifying Logarithmic Axes" on page 531

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

LOGBASE=*base* | E | PI

scales the axis values logarithmically according to the value specified. *Base* must be greater than 1. The number of minor tick marks is a function of the logbase, and is calculated as the logbase minus 2. For example, if logbase=10, there are 8 minor tick marks. If logbase=2, then there are no minor tick marks. Because the value of logbase=e (2.718281828) is so close to 2, it also results in no minor tick marks. How the values are displayed on the axis depends on the LOGSTYLE= option. For example, LOGBASE=10 with the default LOGSTYLE=EXPAND generates an axis like the one in Figure 25.32 on page 404.

Figure 25.1 *Axis Generated with LOGBASE=10 and LOGSTYLE=EXPAND*



Restrictions Not supported by the GRADAR Procedure

When the device specified is ActiveX and LOGBASE=10, formats such as percentage are not applied to the axis values

Example "Example 2: Specifying Logarithmic Axes" on page 531

LOGSTYLE=EXPAND | POWER

specifies whether the values displayed on the logarithmic axis are the values of the base or the values of the power. The LOGSTYLE= option is meaningful only when you use the LOGBASE= option.

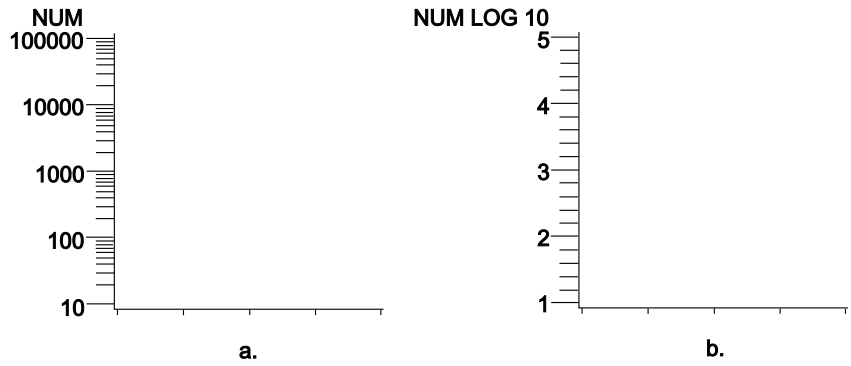
LOGSTYLE=EXPAND specifies that the values displayed are the values of the base raised to successive powers and that the minor tick marks are logarithmically placed. For example, if the base is 10, the values displayed are 10, 100, 1000, 10000, and so on. The default is LOGSTYLE=EXPAND. This statement generates an axis like the one in part (a) of Figure 25.33 on page 405:

```
axis logbase=10 logstyle=expand;
```

LOGSTYLE=POWER specifies that the values displayed are the powers to which the base is raised (for example, 1, 2, 3, 4, 5, and so on). For example, this statement generates an axis like the one in part (b) of Figure 25.33 on page 405:

```
axis logbase=10 logstyle=power;
```

Figure 25.2 *Axes Generated with the LOGSTYLE=option*



If you use the ORDER= option with a logarithmic axis, the values specified by the ORDER= option must match the style specified by the LOGSTYLE= option. For example, if you specify a logarithmic axis with a base of 2 and you want to display the first five expanded values, use this statement:

```
axis logbase=2 logstyle=expand
     order=(2 4 8 16 32);
```

If you use LOGSTYLE=POWER, the values in the ORDER= option must represent the powers to which the base is raised, as in this example:

```
axis logbase=2 logstyle=power order=(1 2 3 4 5);
```

If the values that are specified by ORDER= do not match the type of values specified by LOGSTYLE=, the request for a logarithmic axis is ignored.

Restriction Not supported by the GRADAR Procedure

Example "Example 2: Specifying Logarithmic Axes" on page 531

MAJOR=(*tick-mark-suboptions*) | NONE

modifies the major tick marks. *Tick-mark-suboptions* defines the color, size, and number of the major tick marks. NONE suppresses all major tick marks, although the values represented by those tick marks are still displayed. List all suboptions and their values within one set of parentheses.*Tick-mark-suboption* can be the following:

- COLOR=*tick-color*
- HEIGHT=*tick-height* <*units* >
- NUMBER=*number-of-ticks*
- WIDTH=*thickness-factor*

Restrictions Partially supported ActiveX. HEIGHT is not supported by ActiveX.

AXIS definitions assigned to the group axis of a bar chart by the GAXIS= option ignore MAJOR= because the axis does not use tick marks.

AXIS definitions assigned to the group axis of a block chart ignore MAJOR= because the grid does not use either axes or tick marks.

Note By default, tick marks are placed at three intervals on the spokes of a GRADAR chart. They are placed at the minimum value,

maximum value, and at one value in between. The tick marks on the 12 o'clock spoke are also labeled by default.

See "Tick Mark Description Suboptions" on page 420 for complete descriptions

Examples "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

"Example 2: Specifying Logarithmic Axes" on page 531

"Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

MINOR=(*tick-mark-suboptions*) | NONE

modifies the minor tick marks that appear between major tick marks. *Tick-mark-suboptions* defines the color, number, or size of the minor tick marks. NONE suppresses all minor tick marks. List all suboptions and their values within one set of parentheses. *Tick-mark-suboption* can be the following:

- COLOR=*tick-color*
- HEIGHT=*tick-height* <*units* >
- NUMBER=*number-of-ticks*
- WIDTH=*thickness-factor*

Restrictions Partially supported by ActiveX. HEIGHT is not supported by ActiveX.

Not supported by the GRADAR Procedure

AXIS definitions assigned to the group axis of a bar chart by the GAXIS= option ignore MINOR= because the axis does not use tick marks.

AXIS definitions assigned to the group axis of a block chart ignore MINOR= because the grid does not use either axes or tick marks.

See "Tick Mark Description Suboptions" on page 420 for complete descriptions

Examples "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

"Example 2: Specifying Logarithmic Axes" on page 531

"Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

NOBRACKETS

suppresses the printing of group brackets drawn around the values on the group axis in a bar chart. NOBRACKETS applies only to the group axis of bar charts.

Restrictions Not supported by ActiveX

Not supported by the GRADAR Procedure

See "GROUP=*group-variable*" on page 1030

"GAXIS=AXIS<1 …99>" on page 1030

NOPLANE

removes either the horizontal or vertical three-dimensional axis plane in bar charts produced by the HBAR3D and VBAR3D statements. NOPLANE affects only the axis to which the AXIS statement applies.

To remove selected axis elements such as lines, values or labels, use specific AXIS statement options. To remove all axis elements except the three-dimensional planes use the NOAXIS option in the procedure. To remove the three-dimensional back wall, use the NOFRAME option in the procedure.

Restriction Not supported by the GRADAR Procedure

Example "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

OFFSET=(<*n1*><,*n2*>)<*units* > | (<*n1*<*units*>><,*n2*<*units*>>)

specifies the distance from the first and last major tick marks or bars to the ends of the axis line. The value of (*n1*) is the distance from the beginning (origin) of the axis line to the first tick mark or middle of the first bar. The value of (*n2*) is the distance from the end of the axis line to the last tick mark or middle of the last bar.

On a horizontal axis, the (*n1*) offset is measured from the left end of the axis line and the (*n2*) offset is measured from the right end. On a vertical axis, the (*n1*) offset is measured up from the bottom of the axis line and the (*n2*) offset is measured down from the top of the line.

To specify the same offset for both *n1* and *n2*, use one value, with or without a following comma. For example, either option sets both *n1* and *n2* to 4 centimeters:

```
offset=(4 cm)
offset=(4 cm,)
```

To specify different offsets, use two values, with or without a comma separating them. For example:

```
offset=(4 cm, 2 cm)
```

To specify only the second offset, use only one value preceded by a comma. This option offsets the last major tick mark or bar three centimeters from the right-hand end of the axis line:

```
offset=(,3 cm)
```

You can specify *units* for the *n1*,*n2* pair or for the individual offset values.

Restriction Not supported by the GRADAR procedure

Example "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

ORDER=(*value-list*)

specifies the order in which data values appear on the axis. The values specified by the ORDER= option are the major tick mark values. You can modify the appearance of these values with the VALUE= option.

The way you specify *value-list* depends on the type of variable:

value-list

For numeric variables, *value-list* is either an explicit list of values or a starting value and an ending value with an interval increment, or a combination of both forms:

- *n* < ...*n*>
- *n* TO *n* <BY *increment*>
- *n*< ...*n*> TO *n* <BY *increment* > <*n* < ...*n* > >

If a numeric variable has an associated format, the specified values must be the unformatted values.

Values must be listed in either ascending or descending order. By default the increment value is 1. You can use a negative integer for *increment* to specify a value list in descending order. In all forms, multiple *n* values can be separated by blanks or commas. Here are some examples:

```
order=(2 4 6)
order=(6,4,2)
order=(2 to 10 by 2)
order=(50 to 10 by -5)
order=(0 to 4 by 1, 5 to 19 by 2)
```

There are some instances where the specified range is not evenly divisible by the increment value. In those cases the highest value displayed on the axis is the last incremental value below the ending value for the range. For example, this value list produces a maximum axis value of 9:

```
order=(0 to 10 by 3)
```

value-list

For character variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:

"*value-1*" <" *value-2*" ..." *value-n*">

If a character variable has an associated format, the specified values must be the *formatted* values for PROC GCHART and the *unformatted* values for PROC GPLOT.

You are able to specify up to 256 characters for each variable value.

Character values can be specified in any order, but the character strings must match exactly the variable values in case and spelling. For example:

```
order=("Paris" "London" "Tokyo")
```

Observations can be inadvertently excluded if entries in the *value-list* are misspelled or if the case does not match exactly.

value-list

For date and time values, *value-list* can have the following forms:

- "*SAS-value*"*i* < ..."*SAS-value*"*i*>
- "*SAS-value*"*i* TO "*SAS-value*"*i* <BY *interval*>

"SAS-value"i

is any SAS date, time, or datetime value described for the SAS functions INTCK and INTNX. Enclose the value in quotation marks and specify one of the following for *i*:

D

date

T
time

DT
datetime

interval
is one of the valid arguments for the INTCK or INTNX functions. These are the default intervals:

DAY
default interval for date

SECOND
default interval for time

DTSECOND
default interval for datetime

These value lists use SAS date and time values:

```
order=("25MAY16"d "04JUL16"d "07SEP16"d)
order=("01JUL16"d to "01AUG16"d)
order=("01JUL15"d to "01JAN16"d by week)
order=("9:25"t to "11:25"t by minute)
order=("04JUN16:12:00:00"dt to
       "10JUN16:12:00:00"dt by dtday)
order=("18JAN2015"d to "17DEC2015"d by semimonth)
order=("18JAN2015"d  "01FEB2015"d to "17DEC2015"d by semimonth)
```

With SAS date and time values, use a FORMAT statement so that the tick mark values have an understandable form. For more information about SAS date and time values, see the *SAS Formats and Informats: Reference*For more information about SAS date and time values, see the "About Formats" in *SAS Formats and Informats: Reference*.

Note: The last two value lists result in different axis values. In the first order by semi-month statement, the first axis tick mark value is the starting date of 18JAN2015. The remaining tick marks are generated in 15-day intervals from the 18th. The tick marks are 18JAN2015, 03FEB2015, 18FEB2015, ... 16DEC2015. In the second order by semi-month statement, the starting tick mark is specified at 18JAN2015. This is followed by the start and end dates specified for the SEMIMONTH interval. In this instance the tick marks are 01FEB2015, 15FEB2015, 01MAR2015 ... 15DEC2015.

With any type of *value-list*, specifying values that are not distributed uniformly or are not in ascending or descending order, generates a warning message in the SAS log. The specified values are spaced evenly along the axis even if the values are not distributed uniformly.

Using the ORDER= option to restrict the values displayed on the axis can result in clipping. For example, the data range is 1 to 10 and you specify ORDER=(3 TO 5). Only the data values from 3 to 5 appear on the plot or chart. For charts, the omitted values are still included in the statistic calculation.

Note: Values out of range do not always produce a warning message in the SAS log.

Using the ORDER= option to specify non-uniform intervals for major tick marks can result in no minor tick marks being drawn. It also generates a warning message in the SAS log.

Using the ORDER= option to specify only starting and ending tick mark values without a *BY increment* can result in a recalculation of the requested order. Unless specified, the *BY increment* is 1 by default. The recalculation occurs when more than 20 tick marks are requested. A warning message in the SAS log indicates that the original order request is ignored in favor of using the default order algorithm. This recalculation avoids displaying output with overlapping tick marks.

CAUTION
The ORDER= option does not calculate midpoint values. As a result, it is not interchangeable with the MIDPOINTS= option in the GCHART procedure.

You can use the ORDER= option to specify the order in which the midpoints are displayed on a chart, but do not use it to calculate midpoint values. Make sure that the values that you specify match the midpoint values that are calculated either by default by the GCHART procedure or by the MIDPOINTS= option. For details, see the description of the MIDPOINTS= option for the appropriate statement in Chapter 36, "GCHART Procedure," on page 983.

Restriction Not valid with the ASCENDING, DESCENDING, and NOZEROS options used with the bar chart statements in the GCHART procedure.

Interaction The ORDER= option overrides the suboption NUMBER= described in "Tick Mark Description Suboptions" on page 420 .

Note The ActiveX control supports simple order lists and non-uniform interval values for numerical data and dates.

Examples "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

"Example 5: Filling the Area between Plot Lines" on page 543

"Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

ORIGIN=(<*x*><,*y*>)<*units*> | (<*x*<*units*>><,*y*<*units*>>)
specifies the *x* coordinate and the *y* coordinate of the origin of the axis. The origin of the horizontal axis is the left end of the axis, and the origin of the vertical axis is the bottom of the axis. The ORIGIN= option explicitly positions the axis anywhere on the graphics output area.

If you specify only one value, with or without a comma following it, only the *x* coordinate is set to that value. For example, this specification sets *x* to 4 centimeters:

```
origin=(4 cm,)
```

If you specify two values, with or without a comma separating them, the first value sets the *x* coordinate and the second value sets the *y* coordinate:

```
origin=(2 pct, 4 pct)
```

If you specify one value preceded by a comma, only the *y* coordinate is set to that value, as shown here:

```
origin=(,3 pct)
```

You can specify *units* for the *x*,*y* pair or for the individual coordinates.

Restrictions Not supported by ActiveX

Not supported by the GRADAR procedure

REFLABEL=(*text-argument(s)*) | NONE

creates and defines the appearance of a reference-line label. *Text-argument(s)* defines the appearance or the text of the label, or both. NONE suppresses the reference-line label. *Text-argument(s)* can be one or more of these:

"text-string"

provides up to 256 characters of label text. By default, a reference line does not have a label. Enclose each string in quotation marks. Separate multiple strings with blank spaces. The strings are applied to the reference lines specified by the VREF or HREF option.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVAL*n*
#BYVAL(*BY-variable-name*)

substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n

specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name

names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

#BYVAR*n*
#BYVAR(*BY-variable-name*)

substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n

specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 423

text-description-suboption
modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be the following:

- ANGLE=*degrees*
- AUTOREF
- COLOR=*text-color*
- FONT=*font* | NONE
- FORMAT='*ValidFormat*'
- HEIGHT=*text-height* <*units* >
- JUSTIFY=LEFT | CENTER | RIGHT
- POSITION=TOP| MIDDLE| BOTTOM
- ROTATE=*degrees*
- T=*n*

See "Text Description Suboptions" on page 416 for a complete description of these suboptions.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Style reference Font and Color attributes of the GraphLabelText element

Restrictions Not supported by ActiveX

Not supported by the GRADAR procedure

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

SPLIT="*split-char(s)*"
specifies one or more characters that the AXIS statement uses to break axis values into multiple lines. *Split-char(s)* can be any character value that can be specified in a SAS character variable. Do not delimit when specifying multiple split characters. Each split character must be embedded in the variable values in the data set or sets, or in an associated format. When the AXIS statement encounters the split character, it automatically breaks the value at that point and continues on the next line. For example, suppose the data set contains the value `Berlin, Germany/Europe`, and you specify SPLIT=",/". The value would appear on the axis as follows: If the AXIS statement does not encounter a specified split character, no break in the variable value occurs, and no warning or error is issued.

```
Berlin
Germany
```

```
Europe
```

Axis values specified with VALUE= do not use the split character. For example, suppose you specify this statement:

```
axis1 split="," value=(tick=1 "December, 1999");
```

The value appears on the axis on one line as `December, 1999`. However, any other axis values containing a comma honors the split character.

Restrictions Not supported by ActiveX

Not supported by the GRADAR procedure

Up to 31 split characters are supported. If you specify more than 31 characters, only the first 31 are honored.

Note The split characters themselves are not displayed.

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

STAGGER
offsets the axis values on a horizontal axis. This option is useful when values overlap on an axis. When specifying the ActiveX device, the STAGGER option must sometimes be used in conjunction with the ORDER statement.

STYLE=*line-type*
specifies a line type for the axis line. Valid values for *line-type* are 0 through 46. If you specify STYLE=0, the axis line is not drawn. The default is 1, a solid line.

Style reference Line style attribute of the GraphAxisLine element

Interactions In order for the axis line to be altered by the STYLE= option, the NOFRAME option must also be set. If only the STYLE=option is set, the line framing the axis area is modified.

When specifying both the STYLE= and WIDTH= suboptions, only the WIDTH= specification is used, resulting in a modification to the tick mark thickness but not the axis line type

Using the PNG device with STYLE= results in the specified line type being applied to the line framing the axis area

See Figure 25.52 on page 503 for examples of the available line types

VALUE=(*text-argument(s)*) | NONE
modifies the major tick mark values. That is, this option modifies the text that labels the major tick marks on the axis. *Text-argument(s)* defines the appearance or the text of a major tick mark value, or both. NONE suppresses the major tick mark values, although the major tick marks are still displayed.*Text-argument(s)* can be one or more of these:

"text-string"
provides up to 256 characters of text for the major tick mark value. By default, the value is either the variable value or an associated format value. Enclose each string in quotation marks and separate multiple strings with blanks.

Specified text strings are assigned to major tick marks in order. If you specify only one text string, only the first tick mark value changes, and all the other

tick mark values display the default. If you specify multiple strings, the first string is the value of the first major tick mark. The second string is the value of the second major tick mark, and so on. For example, to change default tick mark values 1, 2, and 3 to `First`, `Second`, and `Third`, use this option:

```
value=("First" "Second" "Third")
```

Note: Although the VALUE= option changes the text displayed at a major tick mark, it does not affect the actual value represented by the tick mark. To change the tick mark values, use the ORDER= option. Also note that with the ActiveX device it is necessary to use the ORDER= option. This ensures that the same number of tick marks are displayed as are with graphics rendered with the other device drivers. For example, specify ORDER=(1 to 12) to ensure that tick marks for all twelve months are displayed.

Note: To change the value of midpoints in bar charts produced with the GCHART procedure, use the MIDPOINTS= option in the procedure.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVAL*n*
#BYVAL(*BY-variable-name*)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

#BYVAR*n*
#BYVAR(*BY-variable-name*)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 423

text-description-suboption
modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be the following:

- ANGLE=*degrees*
- COLOR=*text-color*
- FONT=*font* | NONE
- HEIGHT=*text-height* <*units* >
- JUSTIFY=LEFT | CENTER | RIGHT
- ROTATE=*degrees*
- TICK=*n*

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

For a complete description of these suboptions, see "Text Description Suboptions" on page 416.

Style reference Color attribute of the GraphLabelText graph element

Restrictions Place text description suboptions before the text strings that they modify. Suboptions not followed by a text string affect the default values. To specify and describe the text for individual values or to produce multi-line text, use the TICK= suboption.

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

Interaction This option, with any of its specified suboptions, overrides the RANGE= option when creating vertical bar charts, as with the GBARLINE and GCHART procedures. This is noted in the log as follows:

```
NOTE: Axis value is specified. RANGE option is ignored.
```

Note You might view a graph in the ActiveX control and zoom in on a particular part of a graph for which the VALUE= option is specified. The values are not readjusted in coordination with the zooming.

Examples "Example 2: Specifying Logarithmic Axes" on page 531

"Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

WIDTH=*thickness-factor*

specifies the thickness of the axis line. Thickness increases directly with the value of *thickness-factor*. By default, WIDTH=1.

Style reference LineThickness attribute of the GraphAxisLines element

Interactions In order for the axis line to be altered by the WIDTH= option, the NOFRAME option must also be set. If only the WIDTH= option is set, the line framing the axis area is modified

When specifying both the STYLE= and WIDTH= suboptions, only the WIDTH= specification is used, resulting in a modification to the tick mark thickness but not the axis line type

Example "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

Text Description Suboptions

ANGLE=*degrees*

specifies the angle of the *baseline* with respect to the horizontal. A positive value for *degrees* moves the *baseline* counterclockwise; a negative value moves it clockwise. By default, ANGLE=0 (horizontal) unless the text is automatically angled or rotated to avoid overlapping.

Note: Changing the angle of a vertical axis-label can result in the label being positioned above the graph when using the ActiveX device driver.

Alias A=

Restriction You cannot justify angled text that is used to describe tick marks on a horizontal axis; the JUSTIFY= suboption is not honored when the ANGLE= value is anything other than zero.

See " ROTATE=*degrees* " on page 419

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

AUTOREF

automatically labels each reference line on an axis with the response value at the reference line's position. The AUTOREF suboption is used only with the REFLABEL= option, and can be used in conjunction with the FORMAT= suboption to refine the labels. When the FORMAT= suboption applies a valid format to a response variable value, AUTOREF displays those formatted values as the reference line labels. When the FORMAT= suboption is omitted, AUTOREF displays (as reference line labels) the response variable values as is.

The automatic labels are applied only to reference lines without specific labels assigned to them. For example, the following option uses the response-axis value as the label for every reference line except the second reference line, which is assigned the label `two`:

```
reflabel=(autoref t=2 "two")
```

Note: You might simultaneously request automatic labeling with a PLOT or BUBBLE statement (using the AUTOHREF or AUTOVREF option). In this case the automatic labeling can write on top of the custom label that you specified using the AXIS statement. You must ensure that your custom labels specified using the AXIS statement are not at the same position as automatic labels requested with a different statement.

Restriction Not supported by ActiveX

See "FORMAT='*ValidFormat*'" on page 417

COLOR=*text-color*

specifies the color for the text. If you omit the COLOR= suboption, a color specification is searched for in this order:

1. the CTEXT= option for the procedure
2. the CTEXT= option in a GOPTIONS statement
3. the color of the default style

Alias C=

FONT=*font* | NONE

specifies the font for the text. See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349 for details about specifying *font*. If you omit the FONT= option, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the default style font, NONE

You might use a device-resident font for your midpoint axis labels and find that one or more of the axis labels overlap the legend in your graph. In this case the device-resident font for each label that overlaps the legend is replaced with a SAS/GRAPH font. Because device-resident fonts do not support clipping, a SAS/GRAPH font must be substituted in that case. This includes the axis label and the tick mark labels. An overlap can occur when you use the MODE=PROTECT option on your LEGEND statement with labels on your midpoint axis. To correct an overlap condition, you can use the NOLEGEND option on your plot statement to suppress the legend. Or you can use the positioning options on your LEGEND statement to reposition the legend. See " POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)" on page 440.

Alias F=

FORMAT='*ValidFormat*'

specifies a valid format to be applied to the response-axis value shown with the AUTOREF label on each reference line.

This suboption allows for the character or numeric formatting of variable values for the X axis and Y axis. The FORMAT= suboption of the REFLABEL option controls the display of different values at the reference lines relating to conversion rates, scales, and the like. If an invalid format is specified, then a warning message is issued and the resulting labels have response-axis values with no formatting. When FORMAT= is omitted, the default reverts to the AUTOREF suboption behavior of producing labels automatically with unformatted response-axis values.

Restrictions Must be used with the AUTOREF suboption of the REFLABEL option.

Formatting might accommodate response-axis variable values that are up to 256 characters in length.

Not supported by ActiveX

Note Although the AUTOREF suboption ignores any format assigned to an x- or y-axis variable with the FORMAT statement, the FORMAT= suboption provides a way to format each response-axis variable value.

See " AUTOREF" on page 416

HEIGHT=*text-height* <*units* >
specifies the height of the text characters in number of units. By default, HEIGHT=1 CELL. If you omit the HEIGHT= option, a text height specification is searched for in this order:

1. the HTEXT= option in a GOPTIONS statement
2. the default style value, 1

Alias H=

JUSTIFY=LEFT | CENTER | RIGHT
specifies the alignment of the text. The default depends on the option with which it is used and the text that it applies to.

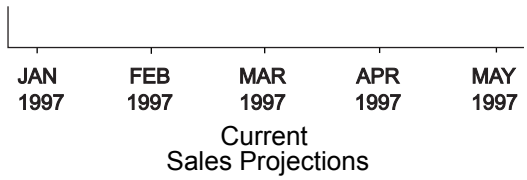
- With the LABEL= option, the following is true:
  - For a left vertical axis label, the default is JUSTIFY=RIGHT.
  - for a right vertical axis label, the default is JUSTIFY=LEFT.
  - for a horizontal axis label, the default is JUSTIFY=CENTER.
- With the REFLABEL= option, the following is true:
  - For a reference line that intersects a vertical axis, the default is JUSTIFY=CENTER. RIGHT places the text string on the right end of the line. CENTER places the text string in the middle of the line. LEFT places the text string to the left of the line.
  - For a reference line that intersects a horizontal axis, the default is JUSTIFY=RIGHT for all procedures except the BAR statement in GBARLINE. For the BAR statement in GBARLINE the default is JUSTIFY=LEFT. RIGHT places the text string just to the right of the line. CENTER centers the text string on top of the line. LEFT places the text string just to the left of the line.
- With the VALUE= option, the following is true:
  - For numeric variables on a vertical axis, the default is JUSTIFY=LEFT, starting with SAS/GRAPH release 9.2. Specify JUSTIFY=RIGHT to align the variable values according to release 9.1 and earlier behavior.
  - For character variables on a vertical axis, the default is JUSTIFY=LEFT.
  - For all variables on a horizontal axis, the default is JUSTIFY=CENTER.

You can use the JUSTIFY= option to print multiple lines of text by repeating the JUSTIFY= option before the text string for each line. You can also use

JUSTIFY= to specify multi-line text at specified major tick marks. For example, this statement produces an axis label and major tick mark values like those shown in Figure 25.34 on page 419.

```
axis label=("Current" justify=c
            "Sales Projections")
     value=(tick=1 "JAN" justify=c "1997"
            tick=2 "FEB" justify=c "1997"
            tick=3 "MAR" justify=c "1997"
            tick=4 "APR" justify=c "1997"
            tick=5 "MAY" justify=c "1997");
```

Figure 25.3 *The JUSTIFY= suboption*



Place text description suboptions before the text strings that they modify.

Alias J=L | C | R

Restriction You cannot justify angled or rotated text that is used to describe tick marks on a horizontal axis; the JUSTIFY= suboption is not honored.

Note With output using ActiveX, text justification is relative to the text string, not the tick mark. For example, left justification means that the left end of the text string is justified with respect to the drawing location, as well as other strings in a multiline label. Because the text is left-justified with respect to the drawing location and not the tick mark, the text string can be placed to the right of a tick mark.

See " TICK=*n* " on page 420

POSITION=TOP | MIDDLE | BOTTOM

specifies the position of a reference-line label relative to the reference line. The default is TOP for both vertical and horizontal reference lines. The POSITION= option is available only on the REFLABEL= option.

- For horizontal reference lines, TOP places the label just above the reference line. MIDDLE places the label on the reference line. BOTTOM places the label just below the reference line.
- For vertical reference lines, TOP places the label at the top end of the reference line. MIDDLE places the label in the middle of the line. BOTTOM places the label at the bottom end of the line.

Restriction Not supported by ActiveX

ROTATE=*degrees*

specifies the angle at which each character of text is rotated with respect to the *baseline* of the text string. A positive value for *degree* rotates the character counterclockwise; a negative value moves it clockwise. By default, ROTATE=0 (parallel to the *baseline*) unless the text is automatically angled or rotated to avoid overlapping.

Alias R=

Restriction You cannot justify angled text that is used to describe tick marks on a horizontal axis; the JUSTIFY= suboption is not honored when the ROTATE= value is anything other than zero.

See " ANGLE=*degrees*" on page 416

TICK=*n*

specifies the *n* reference line or tick mark value. Used only with the REFLABEL= option or the VALUE= option. If neither one is specified, then the TICK= option is ignored.

- With the REFLABEL= option, the TICK= option specifies the nth reference line. It is used to limit modifications to individual reference lines when there are multiple reference lines on an axis. For example, the following option changes the color of only the third reference line's label and leaves all other reference-line labels unchanged:

```
reflabel=(autoref t=3 color=red)
```

  Suboptions that precede the TICK= option affect all the reference-line labels on an axis. Suboptions that follow the TICK= option affect only the specified line's label. For example, the following option assigns the color green to all the reference-line labels on an axis, but left-justifies only the third reference line's label:

```
reflabel(c=green "one" "two" t=3  j=left "three")
```

  In order for the options to be applied to a text string, they must precede the quoted string. In the following option, the `j=left` is ignored because it follows the string:

```
reflabel(c=green "one" "two" t=3 "three" j=left)
```

- With the VALUE= option, the TICK= option specifies the nth major tick mark value. It is used to designate the tick mark value whose text and appearance you want to modify. For example, the following option changes the color of only the third tick mark value and leaves all others unchanged:

```
value=(tick=3 color=red)
```

  Suboptions that precede the TICK= option affect all the major tick mark values. Suboptions that follow the TICK= option affect only the specified value. For example, the following option makes all the major tick mark values four units high and colors all of them blue except for the third one, which is red:

```
value=(height=4 color=blue tick=3 color=red)
```

Alias T=

Note The ActiveX device driver does not support the REFLABEL option.

Tick Mark Description Suboptions

COLOR=*tick-mark-color*

colors the tick marks. If you omit the COLOR= suboption, a color specification is searched for in this order:

1. the COLOR= option in the AXIS statement
2. the CAXIS= option for the procedure

3 the color of the default style

Alias C=

HEIGHT=*tick-height <units>*
specifies the height of the tick mark. The defaults for the HEIGHT= suboption depend on the option with which it is used:

- With the MAJOR= option the default height .5 CELLS.
- With the MINOR= option the default height .25 CELLS.

If you specify a negative number, tick marks are drawn inside the axis.

Alias H=

Restriction Not supported by ActiveX

NUMBER=*number-of-ticks*
specifies the number of tick marks to be drawn. With the MAJOR= option, *number-of-ticks* must be greater than 1. With the MINOR= option, *number-of-ticks* must be greater than 0.

With the MAJOR= option, the NUMBER= suboption can be overridden by a major tick mark specification in the procedure, which in turn can be overridden by the ORDER= option.

With the MINOR= option, the NUMBER= suboption can be overridden by a minor tick mark specification in the procedure.

The NUMBER= option is not valid with logarithmic axes.

Alias N=

WIDTH=*thickness-factor*
specifies the thickness of the tick mark, where *thickness-factor* is a number. Thickness increases directly with *thickness-factor*. By default, WIDTH=1.

Alias W=

Style reference LineThickness attribute of the GraphAxisLines element

Details

Description: AXIS Statement

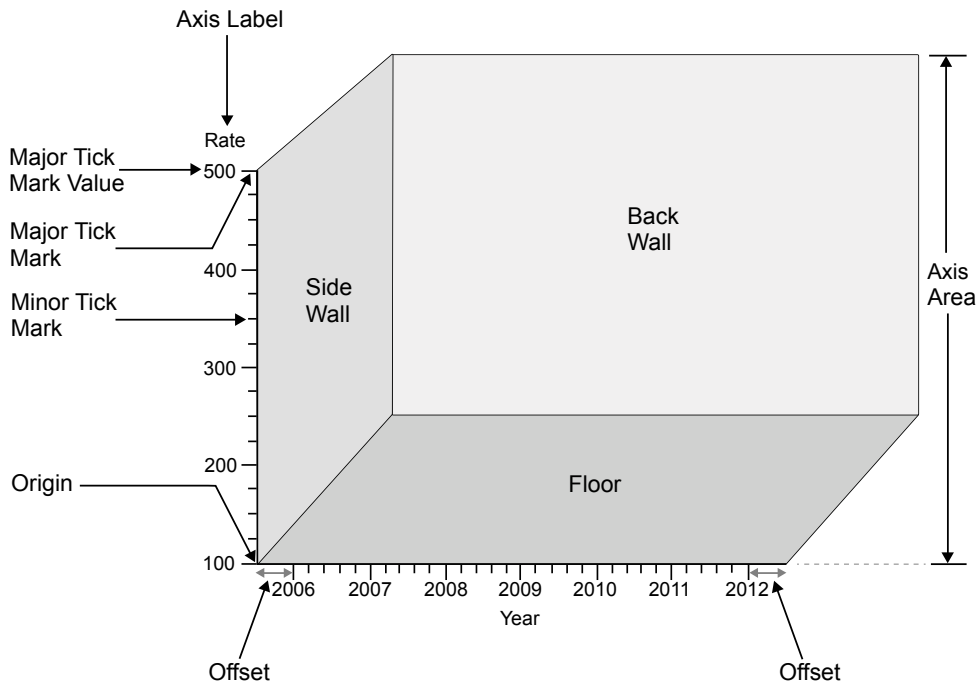
AXIS statements specify the following characteristics of an axis:

- how the axis is scaled
- how the data values are ordered
- the location and appearance of the axis line and the tick marks
- the text and appearance of the axis label and major tick mark values

AXIS definitions are used only when they are explicitly assigned by an option in a procedure that produces graphs with axes.

Figure 25.35 on page 422 illustrates the terms associated with the various parts of axes.

Figure 25.4 *Parts of Axes*



Using Text Description Suboptions

Text description suboptions affect all the strings that follow them unless the suboption is changed or turned off. If the value of a suboption is changed, the new value affects all the text strings that follow it. Consider this example:

```
label=(font=swiss height=4 "Weight"
       justify=right height=3 "(in tons)")
```

FONT=SWISS applies to both `Weight` and `(in tons)`. HEIGHT=4 affects `Weight`, but is respecified as HEIGHT=3 for `(in tons)`. JUSTIFY=RIGHT affects only `(in tons)`.

Using the AXIS Statement

AXIS statements can be defined anywhere in your SAS program. They are global and remain in effect until redefined, canceled, or until the end of your SAS session. AXIS statements are not applied automatically, and must be explicitly assigned by an option in the procedure that uses them.

You can define up to 99 different AXIS statements. If you define two AXIS statements of the same number, the most recently defined statement replaces the previously defined statement of the same number. An AXIS statement without a number is treated as an AXIS1 statement.

Cancel individual AXIS statements by defining an AXIS statement of the same number without options (a null statement):

```
axis4;
```

Canceling one AXIS statement does not affect any other AXIS definitions. To cancel all current AXIS statements, use the RESET= option in a GOPTIONS statement:

```
goptions reset=axis;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current AXIS definitions as well as other settings.

To display a list of current AXIS definitions in the LOG window, use the GOPTIONS procedure with the AXIS option:

```
proc goptions axis nolist;
run;
```

Assigning AXIS Definitions

AXIS definitions must always be explicitly assigned by the appropriate option in the statement that generates the graph. The following table lists the procedures and statements that generate axes, the type of axis, and the statement option that assigns an AXIS definitions to that axis:

Table 25.2 *Procedures and Statements That Generate Axes*

Procedure	Statement that generates an axis	Type of axis	Option that assigns an AXIS definition
GBARLINE	BAR | PLOT	Midpoint axis	MAXIS=
		Response axis	RAXIS=
GCHART	HBAR | VBAR	Group axis	GAXIS=
		Midpoint axis	MAXIS=
		Response axis	RAXIS=
GCONTOUR	PLOT	Horizontal axis	HAXIS=
		Vertical axis	VAXIS=
GPLOT	PLOT	Horizontal axis	HAXIS=
		Vertical axis	VAXIS=
GRADAR	CHART	Star axis	STARAXIS=

Some types of axes cannot use certain AXIS statement options:

- Group and midpoint axes ignore the LOGBASE=, MAJOR=, and MINOR= options.
- Midpoint, horizontal and vertical axes ignore the NOBRACKETS option.

Substituting BY Line Values in a Text String

The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable

name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

- The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.
- There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the "BY Statement" on page 424.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

BY Statement

Processes data and orders output according to the BY group.

Used by: GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT, GRADAR, GTILE, G3D, G3GRID

Type: DATA step statement

Syntax

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …> <NOTSORTED>;

Required Argument

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. By default, the procedure expects observations

in the data set to be sorted in ascending order by all the variables that you specify or to be indexed appropriately.

Optional Arguments

DESCENDING

indicates that the data set is sorted in descending order by the specified variable. The option affects only the variable that immediately follows the option name, and must be repeated before every variable that is not sorted in ascending order. For example, this BY statement indicates that observations in the input data set are arranged in descending order of VAR1 values and ascending order of VAR2 values:

```
by descending var1 var2;
```

This BY statement indicates that the input data set is sorted in descending order of both VAR1 and VAR2 values:

```
by descending var1 descending var2;
```

NOTSORTED

specifies that observations with the same BY value are grouped together, but are not necessarily sorted in alphabetical or numeric order. The observations can be grouped in another way (for example, in chronological order).

NOTSORTED can appear anywhere in the BY statement and affects all variables specified in the statement. NOTSORTED overrides DESCENDING if both appear in the same BY statement.

The requirement for ordering or indexing observations according to the values of BY variables is suspended when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. For NOTSORTED, the procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. Observations with the same value for the BY variables might not be contiguous. The procedure treats each new value that it encounters as the first observation in a new BY group. The procedure creates a graph for that value, even if it is only one observation.

Details

Description: BY Statement

The BY statement divides the observations from an input data set into groups for processing. Each set of contiguous observations with the same value for a specified variable is called a *BY group*. A variable that defines BY groups is called a *BY variable* and is the variable that is specified in the BY statement. When you use a BY statement, the graphics procedure performs the following operations:

- processes each group of observations independently
- generates a separate graph or output for each BY group
- automatically adds a heading called a *BY line* to each graph identifying the BY group represented in the graph
- adds BY statement information below the **Description** field of the catalog entry

By default, the procedure expects the observations in the input data set to be sorted in ascending order of the BY variable values.

Note: The BY statement in SAS/GRAPH is essentially the same as the BY statement in Base SAS. However, the effect on the output is different when it is used with SAS/GRAPH procedures.

Preparing Data for BY-Group Processing

Unless you specify the NOTSORTED option, observations in the input data set must be in ascending numeric or alphabetic order. To prepare the data set, you can sort it with the SORT procedure using the same BY statement that you plan to use in the target SAS/GRAPH procedure. Or you can create an appropriate index on the BY variables.

If the procedure encounters an observation that is out of the proper order, it issues an error message.

If you need to group data in some other order, you can still use BY-group processing. To do so, process the data so that observations are arranged in contiguous groups that have the same BY-variable values and specify the NOTSORTED option in the BY statement.

For an example of sorting the input data set, see "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551.

Controlling BY Lines

Understanding Default Behavior

By default, the BY statement prints a BY line above each graph that contains the variable name followed by an equal sign and the variable value. For example, if you specify BY SITE in the procedure, the default heading when the value of SITE is `London` would be SITE=London.

Suppressing the BY Line

To suppress the entire BY line, use the NOBYLINE option in an OPTION statement or specify HBY=0 in the GOPTIONS statement. See "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551.

Suppressing the Name of the BY Variable

You can suppress the variable name and the equal sign in the heading and leave only the BY value. Use the LABEL statement to assign a null label ("00"X) to the BY variable. For example, this statement assigns a null label to the SITE variable:

```
label site="00"x;
```

Controlling the Appearance of the BY Line

To control the color, font, and height of the BY lines, use the following graphics options in a GOPTIONS statement:

CBY=*BY-line-color*
specifies the color for BY lines.

FBY=*font*
specifies the font for BY lines.

HBY=*n*<*units*>
specifies the height for BY lines.

For a description of each option, see the "Graphics Options and Device Parameters Dictionary" on page 577.

Naming the Catalog Entries

The catalog entries generated with BY-group processing always use incremental naming. This means that the first entry created by the procedure uses the base name and subsequent entries increment that name. The base name is either the default entry name for the procedure (for example, GPLOT) or the name specified with the NAME= option in the action statement. Incrementing the base name automatically appends a number to each subsequent entry (for example, GPLOT1, GPLOT2, and so on). See "Specifying the Catalog Name and Entry Name for Your GRSEGs " on page 138. For an example of incremented catalog names, see "Example 8: Combining Graphs and Reports in a Web Page" on page 557.

Using the BY Statement

Overview

This section describes the following:

- the effect of BY-group processing on the GCHART and GPLOT procedures
- the interaction between BY-group and RUN-group processing
- the requirements for using BY-group processing with the Annotate facility
- how to include BY information in axis labels, reference labels and tick mark values, donut labels, legend labels and values, and titles, notes, and footnotes
- how patterns and symbols are assigned to BY-groups
- the effect of using BY-group processing with the ODS HTML statement

For additional information about any of these topics, refer to the appropriate chapter.

Using the BY Statement with the GCHART Procedure

When you use BY-group processing with the GCHART procedure, you can do the following tasks:

- With the BLOCK, HBAR, and VBAR statements, you can use the PATTERNID=BY option to assign patterns according to BY groups. With PATTERNID=BY, each BY group uses a different PATTERN definition, but all bars or blocks within a BY group use the same pattern. For further information, see "Example: PATTERN and SYMBOL Definitions with BY Groups in the GCHART Procedure" on page 429.
- With the BLOCK statement, you can use the BLOCKMAX= option to produce the same block-height scaling in all block charts in a BY group.
- With the HBAR or VBAR statement, you can use the RAXIS= option. This option produces the same response axis scaling in all horizontal or vertical bar charts in a BY group.
- With the DONUT statement, you can use the LABEL= option to substitute a BY variable value. Or you can substitute a name for #BYVAL or #BYVAR in a text string to display in the donut hole label.

With the PIE and STAR statements, the effect of a BY statement is similar to that of the GROUP= option. The exception is that the GROUP= option enables you to put more than one graph on a single page while the BY statement does not. Do not use a BY variable as the group variable in STAR or PIE statements.

Using the BY Statement with the GPLOT Procedure

You can use the UNIFORM option in the PROC GPLOT statement to produce the same axis scaling for all graphs in a BY group. By default, the range of the axes can vary from graph to graph, but UNIFORM forces the scaling to be the same for all graphs generated by the procedure.

The UNIFORM option applies colors and heights uniformly across all BY-groups.

Using the BY Statement with the RUN Groups

You can use the BY statement with a GCHART or GPLOT procedure that processes data and supports RUN-group processing. In this case, each time you submit an action statement or a RUN statement, you get a separate graph for each value of the BY variable. For example, each of these two RUN-groups produces a separate plot for every value of the BY variable YEAR:

```
/* first run group*/
proc gplot data=sashelp.retail;
   title1 "Sales Summary";
   by year;
   plot sales*year;
run;
   /* second run group */
   plot sales*date;
run;
quit;
```

The BY statement stays in effect for every subsequent RUN group until you submit another BY statement or exit the procedure. Variables in subsequent BY statements replace any previous BY variables.

You can also turn off BY-group processing by submitting a null BY statement (BY;) in a RUN group. Do this with care however, because the null BY statement turns off BY-group processing and the RUN group generates a graph.

For more information, see Chapter 7, “Using Run-Group Processing,” on page 75.

Using the BY Statement with the Annotate Facility

If a procedure that is using BY-group processing also specifies annotation with the ANNOTATE= option in the PROC statement, the same annotation is applied to every graph generated by the procedure.

If you specify annotation with the ANNOTATE= option in the action statements for a procedure, the BY-group processing is applied to the Annotate data set. In this way, you can customize the annotation for the output from each BY group. Include the BY variable in the Annotate data set and use each BY-variable value as a condition for the annotation to be applied to the output for that value.

Using the BY Statement with AXIS, LEGEND, TITLE, FOOTNOTE, and NOTE Statements

AXIS and LEGEND statements can automatically include the BY variable name or BY variable value in the text that they produce for labels, reference labels, values for major tick marks, and legend labels and values. In addition, TITLE, FOOTNOTE, and NOTE statements can automatically include the BY lines in the text that they produce. To insert BY line information into the text strings used by these statements, use the appropriate #BYVAR, #BYVAL, and #BYLINE substitution options. For an example, see “Example 7: Using BY-group Processing to Generate a Series of Charts” on page 551.

Using the BY Statement with PATTERN and SYMBOL Definitions

By default, when using a BY statement, the graph for each BY group uses the same patterns or symbols in their defined order. For example, the BY variable contains four values and there are two response levels for each BY value. In this case the PATTERN1 and PATTERN2 or SYMBOL1 and SYMBOL2 statements are used for each graph . Each BY-group starts over with PATTERN1 or SYMBOL1.

Example: PATTERN and SYMBOL Definitions with BY Groups in the GCHART Procedure

The GCHART procedure, when used with SYMBOL or PATTERN definitions, assigns the symbols or patterns in order to each BY group. For example, if the BY variable REGION has four values—`East`, `North`, `South`, and `West`—the patterns are assigned to the BY-groups in this order:

1. PATTERN1 is assigned to *East.*
2. PATTERN2 is assigned to *North.*
3. PATTERN3 is assigned to *South.*

4 PATTERN4 is assigned to *West.*

You might create sets of graphs from several data sets containing the variable REGION. If you want the same pattern assigned to the same region each time, you must make sure that REGION always has the same four values. Otherwise, the patterns might not be the same across graphs. For example, if the value `North` is missing from the data, the patterns are assigned as follows:

1 PATTERN1 is assigned to *East.*

2 PATTERN2 is assigned to *South.*

3 PATTERN3 is assigned to *West.*

In this case, `South` is assigned pattern 2 instead of pattern 3 and `West` is assigned pattern 3 instead of pattern 4. To avoid this, include the value `North` for the variable REGION, but assign it a missing value for all other variables.

FOOTNOTE Statement

Writes up to 10 lines of text at the bottom of the graph.

Type: Global

See: "TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement" on page 507

Syntax

FOOTNOTE<1 …10> <*text-argument(s)*>;

GOPTIONS Statement

Temporarily sets default values for many graphics attributes and device parameters used by SAS/GRAPH procedures.

Used by: all statements and procedures in a SAS session

Type: Global

Syntax

GOPTIONS <*options-list*>;

Optional Argument

options-list
can be one or more options as listed and described in the "Graphics Options and Device Parameters Dictionary" on page 577.

Details

Description: GOPTIONS Statement

The GOPTIONS statement specifies values for *graphics options*. Graphics options control characteristics of the graph, such as size, colors, type fonts, fill patterns, and symbols. If GOPTIONS are specified, they override the default style. In addition, they affect the settings of device parameters, which are defined in the device entry. Device parameters control such characteristics as the appearance of the display, the type of output produced, and the destination of the output.

The GOPTIONS statement enables you to change these settings temporarily, either for a single graph or for the duration of your SAS session. You can use the GOPTIONS statement to do the following tasks:

- override default values for graphics options that control either graphics attributes or device parameters for a single graph or for an entire SAS session
- reset individual graphics options or all graphics options to their default values
- cancel definitions for AXIS, FOOTNOTE, PATTERN, SYMBOL, and TITLE statements

To change device parameters permanently, you must use the GDEVICE procedure to modify the appropriate device entry or to create a new one. See the Chapter 38, "GDEVICE Procedure," on page 1163 for details.

To review the current settings of all graphics options, use the GOPTIONS procedure. See the Chapter 41, "GOPTIONS Procedure," on page 1249 for details.

Using the GOPTIONS Statement

GOPTIONS statements are global and can be located anywhere in your SAS program. However, for the graphics options to affect the output from a procedure, the GOPTIONS statement must execute before the procedure.

With the exception of the RESET= option, graphics options can be listed in any order in a GOPTIONS statement. The RESET= option should be the first option in the GOPTIONS statement.

A graphics option remains in effect until you specify the option in another GOPTIONS statement, use the RESET= option to reset the values, or end the SAS session. When a session ends, the values of the graphics options return to their default values.

Graphics options are additive. That is, the value of a graphics option remains in effect until the graphics option is explicitly changed or reset or until your SAS session ends. Graphics options remain in effect even after you submit additional GOPTIONS statements specifying different options.

To reset an individual option to its default value, submit the option without a value (a null graphics option.) You can use a comma (but it is not required) to separate a null graphics option from the next one. For example, this GOPTIONS statement sets the values for the background color, the text height, and the text font:

```
goptions cback=blue htext=6 pct ftext="Albany AMT";
```

Note: When there is a space in the font name, surround the name in quotation marks. The quotation marks also indicate use of a system font rather than a software graph font.

To reset only the background color specification to the default and keep the remaining values, use this GOPTIONS statement:

```
goptions cback=;
```

To reset all graphic options to their default values, specify RESET=GOPTIONS:

```
goptions reset=goptions;
```

Alternatively, you can use RESET=ALL, but it also cancels any global statement definitions in addition to resetting all graphics options to default values.

Graphics Option Processing

You can control many graphics attributes through statement options, graphics options, device parameters, or a combination of these. SAS/GRAPH searches these places to determine the value to use, stopping at the first place that gives it an explicit value:

1. statement options
2. the value of the corresponding graphics option
3. the value of a device parameter found in the catalog entry for your device driver

Note: Not every graphics attribute can be set in all three places. See the statement and procedure chapters for the options that can be used with each.

Some graphics options are supported for specific devices or operating environments only. See the SAS Help facility for SAS/GRAPH or the SAS companion for your operating environment for more information.

LEGEND Statement

Controls the location and appearance of legends on two-dimensional plots, contour plots, and charts.

Used by: GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT

Type: Global

Syntax

LEGEND<1 …99> <*options*>;

Summary of Optional Arguments

Appearance options

ACROSS=*number-of-columns*
: specifies the number of columns to use for legend entries.

CBLOCK=*block-color*
: generates and colors a three-dimensional block effect behind the legend.

CBORDER=*frame-color*
: draws a colored frame around the legend.

CFRAME=*background-color*
: specifies the background color of the legend.

CSHADOW=*shadow-color*
: generates and colors a drop shadow behind the legend.

DOWN=*number-of-rows*
: specifies the number of rows to use for legend entries.

FRAME
NOFRAME
: draws a frame around the legend.

FWIDTH=*thickness-factor*
: specifies the thickness of the frame, where *thickness-factor* is a number.

REPEAT=1 | 2 | 3
: specifies how many times the plot symbol is repeated in the legend.

ROWMAJOR
COLMAJOR
: specifies the arrangement of legend entries when there are multiple rows and multiple columns.

SHAPE=BAR(*width*<*units*>,*height*<*units*>) <*units*> | LINE(*length*) <*units*> | SYMBOL(*width*<*units*>,*height*<*units*>) <*units*>
: specifies the size and shape of the legend values displayed in each legend entry.

SPACE=*value-spacing*
: specifies the amount of space between individual legend values along a horizontal baseline.

Position options

MODE=PROTECT | RESERVE | SHARE
: specifies whether the legend is drawn in the procedure output area or whether legend elements can overlay other graphics elements.

OFFSET=(<*x*><,*y*>)<*units*> | (<*x* <*units*>><,*y* <*units*>>)
: specifies the distance to move the entire legend.

ORIGIN=(<*x*><,*y*>)<*units*> | (<*x* <*units* >><,*y* <*units*>>)
: specifies the *x* and *y* coordinates of the lower left corner of the legend box.

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)
positions the legend on the graph.

Text options

LABEL=(*text-argument(s)*) | NONE
modifies a legend label.

ORDER=(*value-list*) | DESCENDING
selects or orders the legend values that appear in the legend.

SPLIT="*split-char(s)*"
specifies one or more characters that the LEGEND statement uses to break a text description string into multiple lines.

VALUE=(*text-argument(s)*) | NONE
modifies the legend value descriptions.

Optional Arguments

ACROSS=*number-of-columns*
specifies the number of columns to use for legend entries. If there are multiple rows and columns in a legend, use the ROWMAJOR and COLMAJOR options to specify the arrangement of legend entries. Specify the ROWMAJOR option to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange entries starting from top to bottom, and then left to right.

See " ROWMAJOR|COLMAJOR" on page 440

Example "Creating a Simple Web Page with the ODS HTML Statement" in *SAS/GRAPH and Base SAS: Mapping Reference*

CBLOCK=*block-color*
generates and colors a three-dimensional block effect behind the legend. The size and position of the block are controlled by the graphics option OFFSHADOW=(*x,y*)..

Restriction The CBLOCK= and CSHADOW= options are mutually exclusive. If both are present, SAS/GRAPH software uses the last one specified.

Interaction The CBLOCK= option is usually used in conjunction with the FRAME, CFRAME=, or CBORDER= options.

See "OFFSHADOW Graphics Option" on page 662

"Creating Drop Shadows and Block Effects" on page 452

CBORDER=*frame-color*
draws a colored frame around the legend.

Style reference Color attribute of the GraphBorderLines graph element

Restriction This option overrides the FRAME option.

Interaction CBORDER= can be used in conjunction with the CFRAME= option.

CFRAME=*background-color*
specifies the background color of the legend. This option overrides the FRAME option.

Style reference Color attribute of the GraphLegendBackground graph element

Restriction If both the CFRAME= and FRAME= options are specified, only the solid background produced by the CFRAME= option is displayed.

Interaction The CFRAME= option can be used in conjunction with the CBORDER= option.

CSHADOW=*shadow-color*
generates and colors a drop shadow behind the legend. The size and position of the shadow is controlled by the graphics option OFFSHADOW=(*x,y*).

Restriction The CSHADOW= and CBLOCK= options are mutually exclusive. If both are present, SAS/GRAPH uses the last one specified.

Interaction The CSHADOW= option is usually specified in conjunction with the FRAME, CFRAME=, or CBORDER= options.

See "OFFSHADOW Graphics Option" on page 662

"Creating Drop Shadows and Block Effects" on page 452

DOWN=*number-of-rows*
specifies the number of rows to use for legend entries. If there are multiple rows and columns in a legend, use the ROWMAJOR and COLMAJOR options to specify the arrangement of legend entries. Specify the ROWMAJOR option to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange entries starting from top to bottom, and then left to right.

Default When there are multiple rows and columns in a legend, the ROWMAJOR option is the default

See " ROWMAJOR|COLMAJOR" on page 440

FRAME | NOFRAME
draws a frame around the legend. The color of the frame is the first color in the color list. NOFRAME suppresses the drawing of a frame, and is the default.

FWIDTH=*thickness-factor*
specifies the thickness of the frame, where *thickness-factor* is a number. The thickness of the line increases directly with *thickness-factor*. By default, FWIDTH=1.

Restriction Not supported by ActiveX

LABEL=(*text-argument(s)*) | NONE
modifies a legend label. *Text-argument(s)* defines the appearance or the text of a legend label, or both. NONE suppresses the legend label in most instances. By default, the text of the legend label is either the variable name or a previously assigned variable label (except in the case of GPLOT with OVERLAY. In that case, the default label is "PLOT"). You can use an overlay variable and suppress the legend label that would display that variable name. Specify a SAS software

font to generate the unprintable hexadecimal character of '00'x, as shown in this example:

```
legend1 label=(font=swiss '00'x);
```

Text-argument(s) can be one or more of these:

"text-string"
provides up to 256 characters of label text. Enclose each string in quotation marks. Separate multiple strings with blanks.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVAL*n*
#BYVAL(*BY-variable-name*)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

#BYVAR*n*
#BYVAR(*BY-variable-name*)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 452

text-description-suboption
modifies a characteristic such as the font, color, or size of the text strings that follows it. *Text-description-suboption* can be as follows:

- ANGLE=*degrees*
- COLOR=*text-color*
- FONT=*font* | NONE
- HEIGHT=*text-height* <*units*>
- JUSTIFY=LEFT | CENTER | RIGHT
- POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)
- ROTATE=*degrees*

For a complete description of these suboptions, see "Text Description Suboptions" on page 445.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Style reference Color attribute of the GraphLabelText graph element

Restrictions Partially supported by ActiveX

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

Starting with SAS 9.4M5, when there is not enough room in the output to display a long label text, the variable name is displayed instead. However, when POSITION=INSIDE and the legend label is too long, the legend is suppressed. In either case, the SAS log displays a warning message.

Examples "Example 3: Rotating Plot Symbols through the Color List" on page 535

"Creating a Simple Web Page with the ODS HTML Statement" in *SAS/GRAPH and Base SAS: Mapping Reference*

MODE=PROTECT | RESERVE | SHARE

specifies whether the legend is drawn in the procedure output area or whether legend elements can overlay other graphics elements. If the space required to display the legend exceeds that of the space required to display the graph, a warning is issued and the legend is suppressed. The MODE= option can take one of these values:

PROTECT

draws the legend in the procedure output area, but a *blanking area* surrounds the legend, preventing other graphics elements from being displayed in the legend. (A blanking area is a protected area in which no other graphics elements are displayed.)

RESERVE

takes space for the legend from the procedure output area, thereby reducing the amount of space available for the graph. If MODE=RESERVE is specified in conjunction with OFFSET=, the legend can push the graph off the graphics output area. RESERVE is valid only when POSITION=OUTSIDE. If POSITION=INSIDE is specified, a warning is issued and MODE= value is changed to PROTECT.

SHARE
draws the legend in the procedure output area. If the legend is positioned over elements of the graph itself, both graphics elements and legend elements are displayed.

By default, MODE=RESERVE unless POSITION=INSIDE. In this case, the default changes to MODE=PROTECT.

If the MODE=PROTECT option is used with labels on the midpoint axis, the axis labels might overlap the legend. In that case, if a device-resident font is used as the label font, the device-resident font for each label that overlaps the legend is replaced with a SAS/GRAPH font. This includes the axis label and the tick mark labels. Because device-resident fonts do not support clipping, a SAS/GRAPH font must be substituted in that case. To correct an overlap condition, you can use the LABEL=NONE and VALUE=NONE options on your midpoint axis statement to suppress the axis labels. Another choice is to use the positioning options on your LEGEND statement to reposition the legend.

Restriction Not supported by ActiveX

See "Positioning the Legend" on page 450

OFFSET=(<*x*><,*y*>)<*units*> | (<*x* <*units*>><,*y* <*units*>>)
specifies the distance to move the entire legend. *x* is the number of units to move the legend right (positive numbers) or left (negative numbers). *y* is the number of units to move the legend up (positive numbers) or down (negative numbers).

To set only the *x* offset, specify one value, with or without a following comma:

```
offset=(4 cm,)
```

To set both the *x* and *y* offset, specify two values, with or without a comma separating them:

```
offset=(2 pct, 4 pct)
```

To set only the *y* offset, specify one value preceded by a comma:

```
offset=(,-3 pct)
```

The OFFSET= option is usually used in conjunction with the POSITION= option to adjust the position of the legend. Moves are relative to the location specified by the POSITION= option, with OFFSET=(0,0) representing the initial position. You can also apply the OFFSET= option to the default legend position.

The OFFSET= option is unnecessary with the ORIGIN= option because the ORIGIN= option explicitly positions the legend and requires no further adjustment. However, if you specify both options, the OFFSET= values are added to the ORIGIN= values, and the LEGEND is positioned accordingly.

Restriction Not supported by ActiveX

See "Positioning the Legend" on page 450

" POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)" on page 440

ORDER=(*value-list*) | DESCENDING
selects or orders the legend values that appear in the legend. DESCENDING specifies that the legend values appear in the legend in reverse of the default order, sorted from maximum to minimum value. Alternatively, use *value-list* to

select only those legend values that you want to display and by their placement in the list, order where they appear in the legend. The way you specify *value-list* depends on the type of variable that generates the legend:

- For numeric variables, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - *n* < ...*n*>
  - *n* TO *n* <BY *increment*>
  - *n* < ...*n*> TO *n* <BY *increment*> <*n* < ...*n*>>

  If a numeric variable has an associated format, the specified values must be the unformatted values.
- For character variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:
  - *"value-1"<" value-2" ..." value-n">*

  If a character variable has an associated format, the specified values must be the formatted values.

For a complete description of *value-list*, see the ORDER= option in the "AXIS Statement" on page 399.

Even though the ORDER= option controls whether a legend value is displayed and where it appears, the VALUE= option controls the text that the legend value displays.

Restrictions Not supported by ActiveX

Up to 256 characters in a variable value might be displayed.

ORIGIN=(<*x*><,*y*>)<*units*> | (<*x* <*units* >><,*y* <*units*>>)

specifies the *x* and *y* coordinates of the lower left corner of the legend box. The ORIGIN= option explicitly positions the legend anywhere on the graphics output area. It is possible to run a legend off the page or overlay the graph.

To set only the *x* coordinate, specify one value, with or without a following comma:

```
origin=(4 cm,)
```

To set both the *x* and *y* coordinates, specify two values, with or without a comma separating them:

```
origin=(2 pct, 4 pct)
```

To set only the *y* coordinate, specify one value preceded by a comma:

```
origin=(,3 pct)
```

The ORIGIN= option overrides the POSITION= option if both are used. Although using the OFFSET= option with the ORIGIN= option is unnecessary, if the OFFSET= option is also specified, it is applied after the ORIGIN= request has been processed.

Restriction Not supported by ActiveX

See "Positioning the Legend" on page 450

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)

positions the legend on the graph. Values for POSITION= are as follows:

OUTSIDE or INSIDE
specifies the location of the legend in relation to the axis area.

BOTTOM or MIDDLE or TOP
specifies the vertical position.

LEFT or CENTER or RIGHT
specifies the horizontal position.

By default, POSITION=(BOTTOM CENTER OUTSIDE). You can change one or more settings. If you supply only one value, then the parentheses are not required. If you specify two or three values and omit the parentheses, SAS/GRAPH accepts the first value and ignores the others.

Once you assign the initial legend position, you can adjust it with the OFFSET= option.

The ORIGIN= option overrides the POSITION= option. The value of the MODE= option can affect the behavior of the POSITION= option.

Restriction Starting with SAS 9.4M5, when there is not enough room in the output to display a long label text, the variable name is displayed instead. However, when POSITION=INSIDE and the legend label is too long, the legend is suppressed. In either case, the SAS log displays a warning message.

See " OFFSET=(<*x*><,*y*>)<*units*> | (<*x* <*units*>><,*y* <*units*>>)" on page 438

" MODE=PROTECT | RESERVE | SHARE" on page 437

REPEAT=1 | 2 | 3

specifies how many times the plot symbol is repeated in the legend. Valid values are 1 to 3. The default value is 3 .

Restriction Not supported by ActiveX

ROWMAJOR | COLMAJOR

specifies the arrangement of legend entries when there are multiple rows and multiple columns. Specify the ROWMAJOR option (the default) to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange the entries starting from top to bottom, and then left to right.

See " ACROSS=*number-of-columns*" on page 434

" DOWN=*number-of-rows*" on page 435

SHAPE=BAR(*width*<*units*>,*height*<*units*>) <*units*> | LINE(*length*) <*units*> | SYMBOL(*width*<*units*>,*height*<*units*>) <*units*>

specifies the size and shape of the legend values displayed in each legend entry. The SHAPE= value that you specify depends on which procedure generates the legend.

BAR(*width*,*height*)<*units*>
(*width*<*units*>,*height*<*units*>) <*units*>

is used with the GCHART procedure, with the GPLOT procedure if you use the AREAS= option, and with the GCONTOUR procedure if you use the PATTERN option. Each legend value is a bar of the specified width and height.

Default *width* is 5, *height* is 0.8, and *units* are CELLS.

Note You can specify the width and height in units of character cells (CELLS), centimeters (CM), inches (IN), percentage of the graphics output area (PCT), or points (PT). There are approximately 72 points in an inch.

Tip You can specify *units* for the *width*,*height* pair or for either or both of the individual coordinates.

LINE(*length*) <*units*>

is used with the GPLOT and GCONTOUR procedures. Each legend value is a line of the length that you specify. Plotting symbols are omitted from the legend values. You can specify *units* for *length*.

Default *length* is 5 and *units* are CELLS.

Note You can specify the length in units of character cells (CELLS), centimeters (CM), inches (IN), percentage of the graphics output area (PCT), or points (PT). There are approximately 72 points in an inch.

SYMBOL (*width*,*height*)<*units*>
(*width*<*units*>,*height*<*units*>) <*units*>

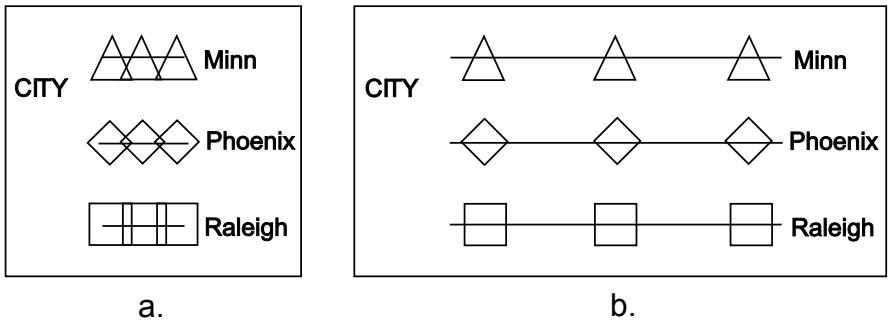
is used with the GPLOT procedure. Each legend value (not each symbol), is the width and height that you specify. For example, this specification produces legend values like the ones in Figure 25.36 on page 441 (a):

```
shape=symbol(.5,.5)
```

This specification produces legend values like the ones in Figure 25.36 on page 441 (b):

```
shape=symbol(2,.5)
```

Figure 25.5 *Legend Values Produced with SHAPE= SYMBOL*



Default *width* is 5, *height* is 1, and *units* are CELLS.

Restriction Not supported by ActiveX

Note You can specify the width and height in units of character cells (CELLS), centimeters (CM), inches (IN), percentage of the graphics output area (PCT), or points (PT). There are approximately 72 points in an inch.

Tips You can specify *units* for the *width*,*height* pair or for either or both of the individual coordinates.

Use the SHAPE=SYMBOL option when using the SYMBOL statement option INTERPOL=NEEDLE to modify the width or height values of the legend entry separately from the size of the GPLOT procedure's plot needles.

SPACE=*value-spacing*

specifies the amount of space between individual legend values along a horizontal baseline. *Value-spacing* can be any nonnegative number, including decimal values. Units are only character cells. You cannot specify a unit of measure, such as inches or percent. The LEGEND statement calculates spacing based on the available display area, and the number and width of legend entries. If the spacing requested exceeds the display area, the legend values and their descriptions are rearranged to fit.

The SPACE= option is ignored if the following is true:

- SPACE=0 is specified.
- The specified spacing is requested in conjunction with vertically stacked columns.

Restriction Not supported by ActiveX

SPLIT="*split-char(s)*"

specifies one or more characters that the LEGEND statement uses to break a text description string into multiple lines. *Split-char(s)* can be any character value that can be specified in a SAS character variable. Do not delimit when specifying multiple split characters. When the LEGEND statement encounters the split character, it automatically breaks the value at that point and continues on the next line. For example, suppose the legend text description contains the string `Berlin, Germany/Europe`, and you specify SPLIT=",/". The legend breaks the text string into top-aligned lines as follows:

```
Berlin
Germany
Europe
```

If the LEGEND statement does not encounter a specified split character, no break in the text occurs, and no warning or error is issued.

Restrictions Not supported by ActiveX

Not supported by the GRADAR procedure

Up to 31 split characters are supported. If you specify more than 31 characters, only the first 31 are honored.

Note The split characters themselves are not displayed.

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

VALUE=(*text-argument(s)*) | NONE

modifies the legend value descriptions. *Text-argument(s)* defines the appearance or the text of the value descriptions. By default, value descriptions are the values of the variable that generates the legend or an associated format value. Numeric values are right-justified and character values are left-justified.

NONE suppresses the value descriptions although the legend values (bars, lines, and so on) are still displayed. *Text-argument(s)* can be one or more of these:

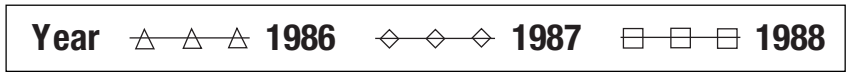
"text-string"

provides up to 256 characters of text for the value description. Enclose each string in quotation marks. Separate multiple strings with blanks. Value description text that is too long to fit in a legend area can result in the entire value not being displayed.

Specified text strings are assigned to the legend values in order. If you submit only one string, only the first legend entry uses the value of that string. If you specify multiple strings, the first string is the text for the first entry; the second string is the text for the second entry; and so on. For example, this specification produces legend entries like those shown in Figure 25.37 on page 443:

```
value=("1986" "1987"
"1988")
```

Figure 25.6 *Specifying Value Descriptions with the VALUE= Option*



In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVAL*n*
#BYVAL(*BY-variable-name*)

substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n

specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name

names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

#BYVAR*n*
#BYVAR(*BY-variable-name*)

substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the

text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 452

text-description-suboption
modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be as follows:

- ANGLE=*degrees*
- COLOR=*text-color*
- FONT=*font* | NONE
- HEIGHT=*text-height* <*units* >
- JUSTIFY=LEFT | CENTER | RIGHT
- POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)
- ROTATE=*degrees*
- TICK=*n*

Place text description suboptions before the text strings that they modify. Suboptions not followed by a text string affect the default values. To specify and describe the text for individual values or to produce multi-line text, use the TICK= suboption.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

To order or select legend entries, use the ORDER= option.

Restrictions Partially supported by ActiveX

ActiveX control does not support changing the font in the middle of specifying descriptive text for a legend.

NONE is not supported by ActiveX

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

See "Text Description Suboptions" on page 445

" ORDER=(*value-list*) | DESCENDING" on page 438

Text Description Suboptions

ANGLE=*degrees*
specifies the angle of the legend label or legend value description text with respect to the horizontal. A positive value for degrees moves the text counterclockwise; a negative value moves it clockwise. By default, ANGLE=0 (horizontal).

Note: Some settings of ANGLE= in the LEGEND statement might result in undesirable text positioning.

Alias A=

Restriction Not supported by ActiveX

See "ROTATE=*degrees*" on page 447

COLOR=*text-color*
specifies the color of the text. If you omit the COLOR= suboption, a color specification is searched for in this order:

1. the CTEXT= option for the procedure
2. the CTEXT= option in a GOPTIONS statement
3. the color of the default style

Alias C=

FONT=*font* | NONE
specifies the font for the text. See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349 for information about specifying fonts. If you omit the FONT= suboption, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the default style font, NONE

Alias F=

Restriction ActiveX control does not support changing the font in the middle of specifying descriptive text for a legend.

HEIGHT=*text-height <units>*
specifies the height of the text characters in the number of units. By default, HEIGHT=1 CELL. If you omit the HEIGHT= suboption, a text height specification is searched for in this order:

1. the HTEXT= option in a GOPTIONS statement
2. the height specified by the default style

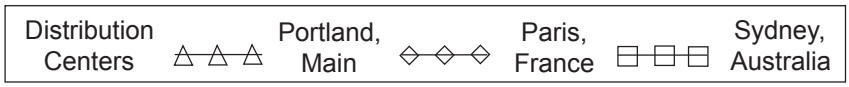
Alias H=

JUSTIFY=LEFT | CENTER | RIGHT
specifies the alignment of the text. The default for character variables is JUSTIFY=LEFT. The default for numeric variables is JUSTIFY=RIGHT. Associating a character format with a numeric variable does not change the default justification of the variable.

You can use the JUSTIFY= suboption to print multiple lines of text by repeating the suboption before the text string for each line. For example, this statement produces a legend label and value descriptions like those shown in Figure 25.38 on page 446:

```
legend label=(justify=c "Distribution"
              justify=c "Centers")
value=(tick=1 justify=c "Portland,"
              justify=c "Maine"
       tick=2 justify=c "Paris,"
              justify=c "France"
       tick=3 justify=c "Sydney,"
              justify=c "Australia");
```

Figure 25.7 *Specifying Multiple Lines of Text with the JUSTIFY= Suboption*



Place text description suboptions before the text strings that they modify.

Alias J=L | C | R

See " TICK=*n*" on page 448

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)
places the legend label in relation to the legend entries. The POSITION= suboption is used only with the LABEL= option. By default, POSITION=LEFT.

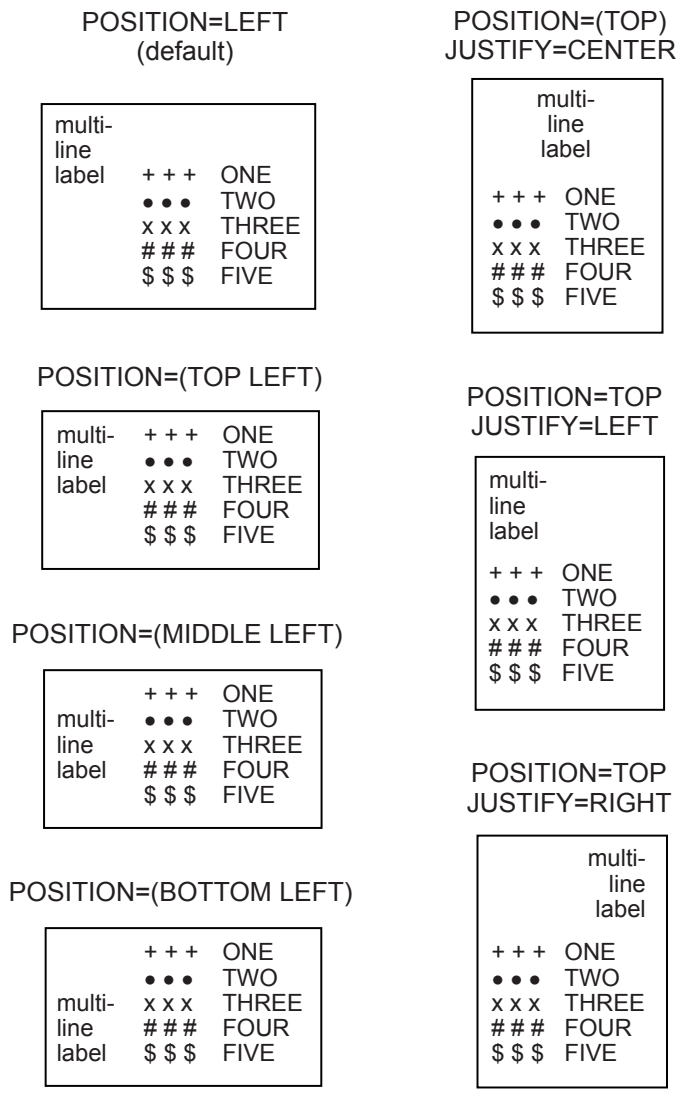
The parentheses are not required if only one value is supplied. If you specify two or three values and omit the parentheses, SAS/GRAPH accepts the first value and ignores the others.

Figure 25.39 on page 447 shows some of the ways the POSITION= suboption affects a multiple-line legend label in which the entries are stacked in a column (ACROSS=1). This figure uses a label specification such as the following:

```
label=("multi-"
        justify=left "line"
        justify=left "label"
        position=left)
```

In this specification, the POSITION= suboption specifies the default value, LEFT, which is represented by the first legend in the figure. The POSITION= value is indicated above each legend. The default justification is used unless you also use the JUSTIFY= suboption.

Figure 25.8 *Using the POSITION= Suboption with Multiple-line Legend Labels*



In addition, specifying POSITION=RIGHT mirrors the effect of POSITION=LEFT, and specifying POSITION=BOTTOM mirrors the effect of POSITION=TOP.

Restriction Partially supported by ActiveX

ROTATE=*degrees*

specifies the angle at which each character of text is rotated with respect to the *baseline* of the text string. A positive value for degree rotates the character counterclockwise; a negative value moves it clockwise. By default, ROTATE=0 (parallel to the *baseline*).

Note: Some settings of ROTATE= in the LEGEND statement might result in undesirable text positioning.

Alias R=

Restriction Not supported by ActiveX

See "ANGLE=*degrees*" on page 445

TICK=*n*

specifies the nth legend entry. The TICK= suboption is used only with the VALUE= option to designate the legend entry whose text and appearance you want to modify. For example, to change the text of the third legend entry to `Minneapolis`, specify the following code:

```
value=(tick=3 "Minneapolis")
```

The characteristics of all other value descriptions remain unchanged.

If you use the TICK= suboption when you designate text for one legend entry, you must also use it when you designate text for any additional legend entries. For example, this option changes the text of both the second and third legend entries:

```
value=(tick=2 "Paris" tick=3 "Sydney")
```

If you omitted TICK=3, the text of the second legend entry would be `ParisSydney`.

Text description suboptions that precede the TICK= suboption affect all the value descriptions for the legend unless the same suboption (with a different value) follows a TICK= specification. Text description suboptions that follow the TICK= suboption affect only the specified legend entry. For example, suppose you specify this option for a legend with three entries:

```
value=(color=red font=swiss tick=2 color=blue)
```

The text of all three entries would use the Swiss font; the first and third entries would be red and only the second entry would be blue.

Alias T=

Details

Description: LEGEND Statement

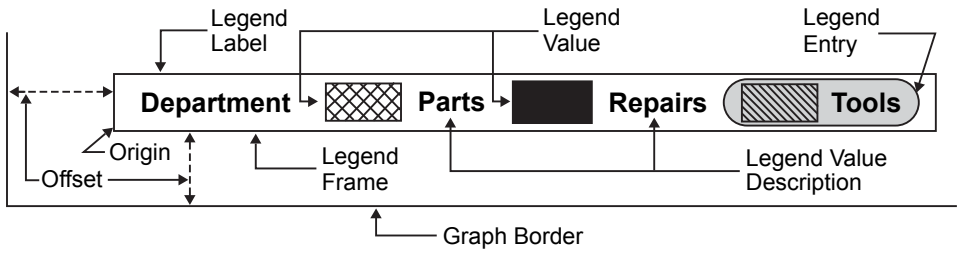
LEGEND statements specify the characteristics of a legend but do not create legends. The characteristics are as follows:

- the position and appearance of the legend box
- the text and appearance of the legend label
- the appearance of the legend entries, including the size and shape of the legend values
- the text of the labels for the legend values

LEGEND definitions are not automatically applied when a procedure generates a legend. Instead, they must be explicitly assigned with a LEGEND= option in the appropriate procedure statement.

The following figure illustrates the terms associated with the various parts of a legend.

Figure 25.9 Parts of a Legend



Using Text Description Suboptions

Text description suboptions affect all the strings that follow them unless the suboption is changed or turned off. If the value of a suboption is changed, the new value affects all the text strings that follow it. Consider this example:

```
label=(font=albany amt height=4 "Weight"
       justify=right height=3 "(in tons)")
```

FONT=ALBANY applies to both `Weight` and `(in tons)`. HEIGHT=4 affects `Weight`, but is respecified as HEIGHT=3 for `(in tons)`. JUSTIFY=RIGHT affects only `(in tons)`.

Using the LEGEND Statement

LEGEND statements can be located anywhere in your SAS program. They are global and remain in effect until canceled or until you end your SAS session. LEGEND statements are not applied automatically, and must be explicitly assigned by an option in the procedure that uses them.

You can define up to 99 different LEGEND statements. If you define two LEGEND statements of the same number, the most recently defined statement replaces the previously defined statement of the same number. A LEGEND statement without a number is treated as a LEGEND1 statement.

Cancel individual LEGEND statements by defining a LEGEND statement of the same number without options (a null statement):

```
legend4;
```

Canceling one LEGEND statement does not affect any other LEGEND definitions. To cancel all current LEGEND statements, use RESET= in a GOPTIONS statement:

```
goptions reset=legend;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current LEGEND definitions as well as other settings.

To display a list of current LEGEND definitions in the SAS LOG window, use the GOPTIONS procedure with the LEGEND option:

```
proc goptions legend nolist;
run;
```

Positioning the Legend

How to Position a Legend

By default, the legend shares the procedure output area with the procedure output, such as a map or bar chart. See "How Graphics Elements Are Placed in the Graphics Output Area" on page 87. However, several LEGEND statement options enable you to position a legend anywhere on the graphics output area and even to overlay the procedure output. This section describes these options and their effect on each other.

Positioning the Legend on the Graphics Output Area

There are two ways that you can position the legend on the graphics output area:

- Describe the general location of the legend with the POSITION= option. If necessary, fine-tune the position with the OFFSET= option.
- Position the legend explicitly with the ORIGIN=option.

Using POSITION= and OFFSET=

The values of the POSITION= option affect the legend in two ways:

- OUTSIDE and INSIDE determine whether the legend is located outside or inside the axis area.
- BOTTOM or MIDDLE or TOP (vertical position) and LEFT or CENTER or RIGHT (horizontal position) determine where the legend is located in relation to its OUTSIDE or INSIDE position.

Figure 25.41 on page 450 shows the legend positions inside the axis area.

Figure 25.10 *Legend Positions inside the Axis Area*

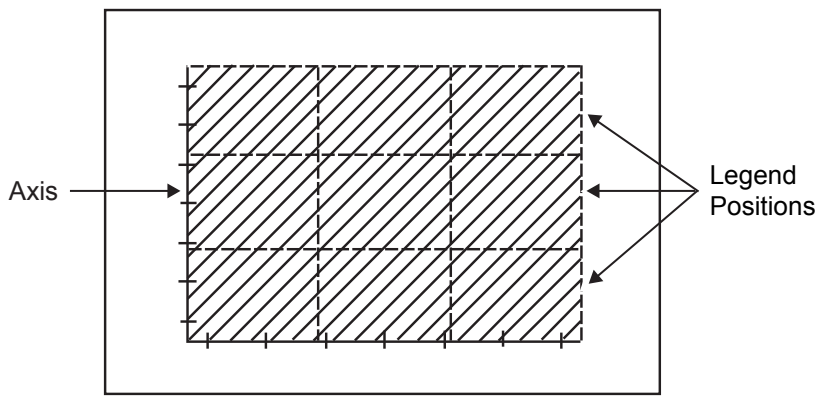
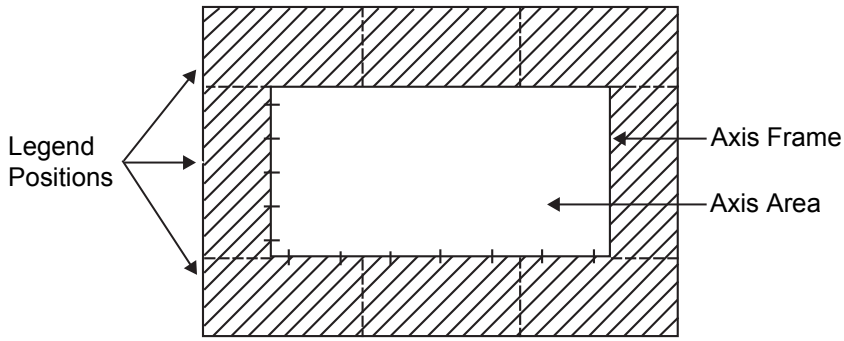


Figure 25.42 on page 451 shows legend positions outside the axis area.

Figure 25.11 *Legend Positions outside the Axis Area*



The default combination is POSITION=(BOTTOM CENTER OUTSIDE). The combination (OUTSIDE MIDDLE CENTER) is not valid.

Use OFFSET=(*x*,*y*) to adjust the position of the legend specified by the POSITION= option. The *x* value shifts the legend either left or right and the *y* value shifts the legend either up or down.

The offset values are always applied after the POSITION= request. For example, if POSITION=(TOP RIGHT OUTSIDE), the legend is located in the upper right corner of the graphics output area. If OFFSET=(0,0) is specified, the legend does not move. If OFFSET=(–5,–8)CM, the legend moves 5 centimeters to the left and 8 centimeters down.

Using ORIGIN=

Use ORIGIN=(*x*,*y*) to specify the coordinates of the exact location of the lower left corner of the legend box. Because ORIGIN=(0,0) is the lower left corner of the graphics output area, the values of *x* and *y* must be positive. If you specify negative values, a warning is issued and the default value is used.

Relating Legends to Other Graphics Elements

By default, the legend is inside the procedure output area and the space that it occupies reduces the size of the graph itself. If a choice must be made between displaying the legend or the graph, the legend is suppressed. To control how the legend relates to the other elements of the graph, use the MODE= option. These are values for the MODE= option:

- RESERVE reserves space for the legend outside the axis area and moves the graph to make room for the legend. This is the default setting and is valid only when POSITION=OUTSIDE.
- PROTECT prevents the legend from being overwritten by the procedure output. PROTECT blanks out graphics elements, allowing only legend elements to be displayed in the legend's space.
- SHARE displays both graphics elements and legend elements in the same space. This setting is usually used when the legend is positioned inside the axis area. SHARE is useful when the graph has a space that the legend can fit into.

Interactions between POSITION= and MODE=

You cannot specify both POSITION=INSIDE and MODE=RESERVE because MODE=RESERVE assumes that the legend is outside the axis area, and POSITION=INSIDE positions the legend inside the axis area. Therefore, when you specify POSITION=INSIDE, change the value of the MODE= option to SHARE or PROTECT. Otherwise, SAS/GRAPH issues a warning and automatically changes the MODE= value to PROTECT.

Creating Drop Shadows and Block Effects

You can produce a drop shadow or a three-dimensional block effect behind the legend. Use the CSHADOW= or CBLOCK= option in the LEGEND statement in conjunction with the graphics option OFFSHADOW=(*x*,*y*).

The value of *x* determines how far the shadow or block extends to the right (positive numbers) or to the left (negative numbers) of the legend. The value of *y* determines how far the shadow or block extends above (positive numbers) or below (negative numbers) the legend. If OFFSHADOW=(0,0) is specified, the shadow or block is not visible.

By default, OFFSHADOW=(0.0625, -0.0625) IN. That is, the shadow or block extends 1/16th of an inch to the right and 1/16th of an inch below the legend.

Substituting BY Line Values in a Text String

The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

- The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.
- There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the “BY Statement” on page 424.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

NOTE Statement

Writes lines of text in the output.

Type: Local

Restriction: Not supported by ActiveX, or by the SGMAP procedure

Tip: By drawing a white box in back of a note, that boxed note can appear on top of the graph instead of behind it. Here is an example of code that outputs a note on top of a map and a note on top of a chart.

```
filename odsout '.';

goptions device=png;
goptions cback=white;

ODS LISTING CLOSE;
ODS HTML path=odsout body="map01.htm" style=htmlblue;

title "Note Text On a Map";
pattern1 v=s color=tan;

proc gmap data=mapsgfk.us map=mapsgfk.us;
note move=(2,32)pct font="albany amt/bold" height=4pct
 color=white box=1 blank=yes
 color=black "Note with 'box' behind it displays on top of map";
id state;
choro state / levels=1 nolegend;
run;

title "Note Text On a Chart";
proc gchart data=maps.us;
note move=(2,50)pct font="albany amt/bold" height=4pct
 color=white box=1 blank=yes
 color=black "Note with 'box' behind it displays on top of chart";
hbar state / type=freq levels=7;
run;
quit;
ODS HTML CLOSE;
ODS LISTING;
```

See: "TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement" on page 507

Syntax

NOTE <*text-arguments(s)*>;

ODS HTML Statement

Opens or closes the HTML destination.

Used by: GANNO, GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT, GRADAR, GREPLAY, GSLIDE, G3D, G3GRID

Requirement: On mainframes, either GPATH= or PATH= is required.

Syntax

ODS HTML <(<ID=> *identifier*)> <<*action*> | <option(s)>>;

Details

Description: ODS HTML Statement

This section describes the ODS HTML statement as it relates to SAS/GRAPH procedures. For complete information about the ODS HTML statement, see the *SAS Output Delivery System: User's Guide*.

The ODS HTML statement opens or closes the HTML destination. This destination is the default output destination for the Windows and UNIX operating systems. When the destination is open, the procedure produces output that is written in Hypertext Markup Language in the form of an HTML file. If no device is specified, SAS/GRAPH by default, creates a PNG file containing the graph. The HTML file references the PNG file in order to display the graph in a web page.

SAS/GRAPH adds data tip text to some graphs depending on the device specified. These data tips are generated by default using the values of fields in a SAS data set. You can specify the DESCRIPTION= option on the SAS/GRAPH procedure to change or remove the data tip text. For more information about using data tips see "Data Tips for Web Presentations" on page 253.

The FILE= option identifies the file that contains the HTML version of the procedure output. With SAS/GRAPH, the body file contains references to the graphs. If DEVICE=PNG, the graphs are stored in separate PNG files. When you view the body file in a browser, the graphs are automatically displayed. By default with ODS processing, both the HTML and PNG files are stored in the current WORK library. To specify a destination for all the HTML and PNG files, use the PATH= option. You can store the PNG files in a different location than the HTML files. Use the GPATH= option to specify a location for the PNG files, and the PATH= option to specify the location of the HTML files. In both cases, the destination must be an aggregate storage location.

Anchors

ODS HTML automatically creates an *anchor* for every piece of output generated by the SAS procedures. An anchor specifies a particular location within an HTML file. In SAS/GRAPH, an anchor usually defines a link target such as a graph whose location is defined in an IMG element.

In order for the links from the contents, page, or frame file to work, each piece of output in the body files must have a unique anchor to link to. The anchor for the first piece of output in a body file acts as the anchor for that file. These anchors are used by the frame and contents files, if they are created, to identify the targets for the links that ODS HTML automatically generates. For more information about using anchors with the ODS HTML statement, see the *SAS Output Delivery System: User's Guide*.

You can specify a name for an HTML anchor with the following code:

```
ods html anchor="string";
```

This enables you to link directly to that identifying name.

PATTERN Statement

Defines the characteristics of patterns used in graphs.

Used by: GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT, SYMBOL statement, Annotate facility

Type: Global

Syntax

PATTERN<1 …255>
<COLOR=*pattern-color* | *_style_*>
<IMAGE=*fileref* | *"external-file"*>
<IMAGESTYLE=TILE | FIT>
<REPEAT=*number-of-times*>
<VALUE=*bar/block-pattern* | *plot-pattern* | *pie/star-pattern*>;

Optional Arguments

COLOR=*pattern-color* | *_style_*
specifies the color of the fill. *Pattern-color* is any SAS/GRAPH color name. The _STYLE_ value specifies the appropriate color based on the current style. See Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363.

Using the COLOR= option with a null value cancels the color specified in a previous PATTERN statement of the same number without affecting the values of other options.

The COLOR= option overrides the CPATTERN= graphics option.

The CFILL= option in the PIE and STAR statements overrides the COLOR= option. For details, see "Controlling Slice Patterns and Colors" on page 1074.

No color can be specified for a PATTERN statement, that is, neither the COLOR= nor the CPATTERN= option is used. In this case the PATTERN statement rotates the specified fill through each color in the color list before the next PATTERN statement is used.

Alias C=

Restriction Partially supported by ActiveX

See "Working with PATTERN Statements" on page 466

Example "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

CAUTION **Omitting the COLOR= option in a PATTERN statement can cause the PATTERN statement to generate multiple PATTERN definitions.**

IMAGE=*fileref* | "*external-file*"

specifies an image file that is used to fill one or more bars of a bar chart. The image file can be generated by the HBAR and VBAR statements of the GCHART procedure. And, in some cases, the image file can be generated by the HBAR3D and VBAR3D statements of the GCHART procedure. The format of the external file specification varies across operating environments. See the companion reference for your specific operating environment.

Restriction Partially supported by ActiveX

Notes When you specify an image file to fill a bar, the bar is not outlined. Also, the COLOR= and VALUE= options are ignored.

If an image is specified in a PATTERN statement that is used with another type of chart, then the PATTERN statement is ignored and default pattern rotation is affected. For example, you can submit a PIE statement when an image has been specified in a PATTERN statement. In this case the default fill pattern is used for the pie slices, with each slice in the pie displaying the fill pattern in the same color.

For DEVICE=ACTIVEX and DEVICE=ACTXIMG, if you do not specify a pathname to the image, then the ActiveX control searches a predefined list of locations to try to find the image. If all else fails, the ActiveX control looks for the image on the web. It is recommended that you specify the pathname to the image.

For DEVICE=ACTIVEX and DEVICE=ACTXIMG, the IMAGE= option works for the VBAR and HBAR statements as well as the VBAR3D and HBAR3D statements. For all other devices, the IMAGE= option works only for the VBAR and HBAR statements.

See "Displaying Images on Data Elements" on page 389 for related information

"IMAGESTYLE=TILE | FIT" on page 456

IMAGESTYLE=TILE | FIT

specifies how the image specified in the IMAGE= option is to be applied to fill a bar in a bar chart. The TILE value repeats the image as needed to fill the bar. The FIT value stretches a single instance of the image to fill the bar.

Default TILE

Restriction Partially supported by ActiveX

REPEAT=*number-of-times*

specifies the number of times that a PATTERN definition is applied before the next PATTERN definition is used.

The behavior of the REPEAT= option depends on the color specification:

- If you use both the COLOR= and REPEAT= options in a PATTERN statement, the pattern is repeated the specified number of times in the specified color. The fill can be either the default solid or a fill specified with the VALUE= option.
- You can use the CPATTERN= option in a GOPTIONS statement to specify a single pattern color. Use the REPEAT= option either alone or with the VALUE= option in a PATTERN statement. The resulting hatch pattern is repeated the specified number of times.
- You can omit both the COLOR= and CPATTERN= options and use the REPEAT= option. If you use the REPEAT= option alone, it generates default solids. Or you can use the REPEAT= option in conjunction with the VALUE= option in a PATTERN statement. The resulting pattern is rotated through each color in the color list. The entire group generated by this cycle is then repeated the number of times specified in the REPEAT= option. Thus, the total number of patterns produced depends on the number of colors in the current color list.

Using REPEAT= with a null value cancels the repetition specified in a previous PATTERN statement of the same number without affecting the values of other options.

Alias R=

Default REPEAT=1

Restriction Partially supported by ActiveX

See “Understanding Pattern Sequences” on page 468

VALUE=*bar/block-pattern*

Patterns are specified for the following:

- bar charts produced by the HBAR, HBAR3D, VBAR, and VBAR3D statements in the GCHART procedure including two-dimensional and three-dimensional bar shapes.
- the front surface of blocks in block charts produced by the BLOCK statement in the GCHART procedure.

Values for *bar/block-pattern* are as follows:

EMPTY

an empty pattern. The ActiveX control does not support EMPTY.

Alias E

SOLID

a solid pattern (the only valid value for three-dimensional charts).

Alias S

style<density>

a shaded pattern. *Style* specifies the direction of the lines:

L

specifies left-slanting lines.

R

specifies right-slanting lines.

X

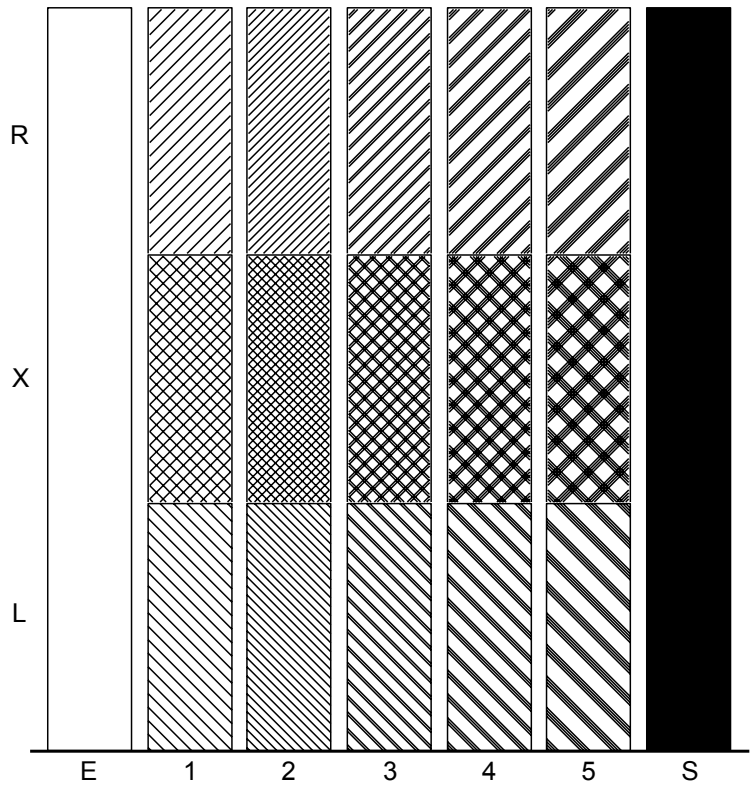
specifies crosshatched lines.

Density specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest shading.

Note: *style<density>* is not supported by the ActiveX device driver.

Figure 25.43 on page 458 shows all of the patterns available for bars and blocks.

Figure 25.12 *Bar and Block Patterns*



If no valid patterns are available, default bar and block fill patterns are selected in this order:

1. SOLID
2. X1–X5
3. L1–L5
4. R1–R5

Each fill is used once with every color in the color list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Alias V=

Restriction Partially supported by ActiveX

VALUE=*plot-pattern*

specifies patterns for the following:

- contour levels in contour plots produced by the GCONTOUR procedure
- areas under curves in plots produced by the AREAS= option in the PLOT statement in the GPLOT procedure

Values for *plot-pattern* are as follows:

MEMPTY

specifies an empty pattern.

Aliases ME

EMPTY is a valid alias for the GCONTOUR and GPLOT procedures.

E is a valid alias for the GCONTOUR and GPLOT procedures.

MSOLID

specifies a solid pattern.

Aliases MS

SOLID is also a valid alias.

S is also a valid alias.

M*density<style<angle>>*

specifies a shaded pattern.

Density specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest shading.

Style specifies the type of the pattern lines:

N

specifies parallel lines (the default).

X

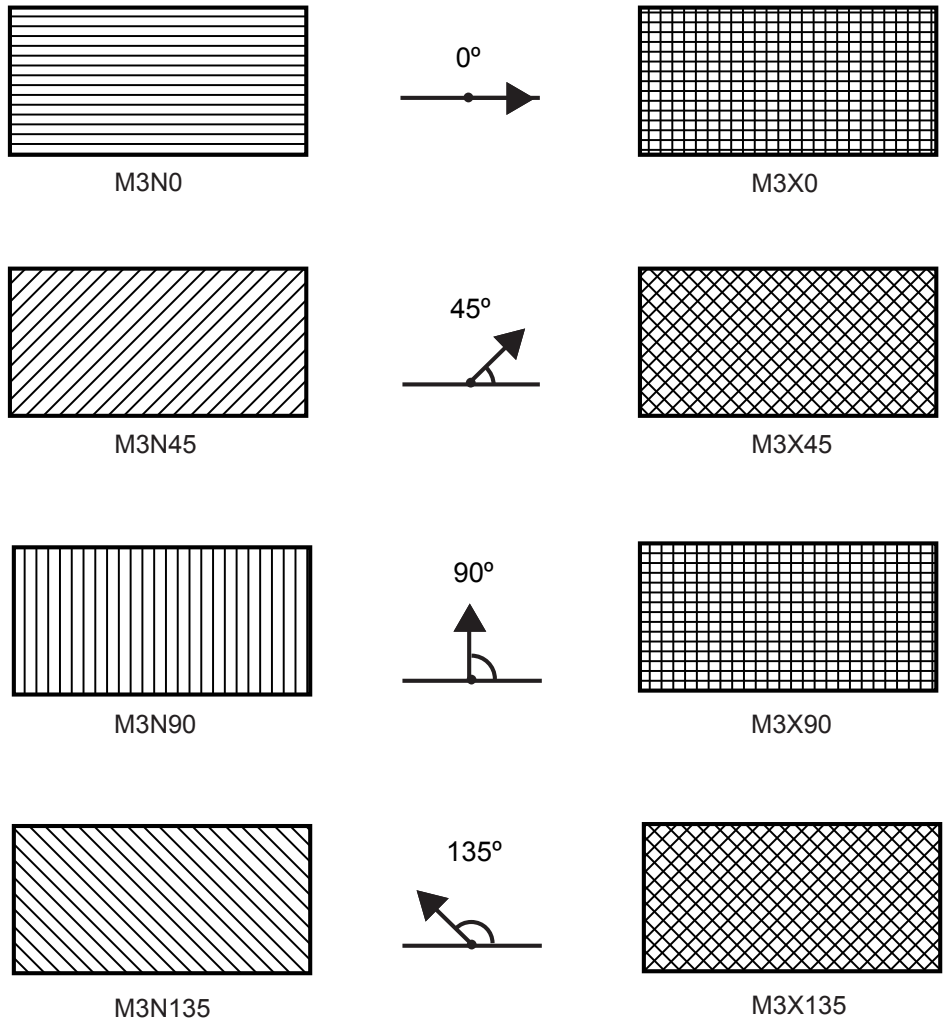
specifies crosshatched lines.

Angle specifies the angle of the pattern lines. Specify a number from 0 to 360. The number specifies the degrees at which the parallel lines are drawn, measured from the horizontal. By default, *angle* is 0 (lines are horizontal).

Note M*density<style<angle>>* is not supported by the ActiveX device driver.

Figure 25.44 on page 460 shows some typical plot patterns.Figure 25.44 on page 460 shows some typical map patterns.

Figure 25.13 Plot Patterns



If no valid patterns are available, default plot fill patterns are selected in this order:

1. MSOLID
2. M2N0
3. M2N90
4. M2X45
5. M4N0
6. M4N90
7. M4X90

Each fill is used once with every color in the color list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Alias V=

Restriction Partially supported by ActiveX

VALUE=*pie/star-pattern*

specifies patterns for pie and star charts produced by the PIE and STAR statements in the GCHART procedure. Values for *pie/star-pattern* are as follows:

PEMPTY

specifies an empty pattern. EMPTY or E are also valid aliases.

Alias PE

PSOLID

specifies a solid pattern. SOLID or S are also valid aliases.

Alias PS

P*density<style<angle>>*

specifies a shaded pattern.

Density specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest shading.

Style specifies the type of the pattern lines:

N

specifies parallel lines (the default).

X

specifies crosshatched lines.

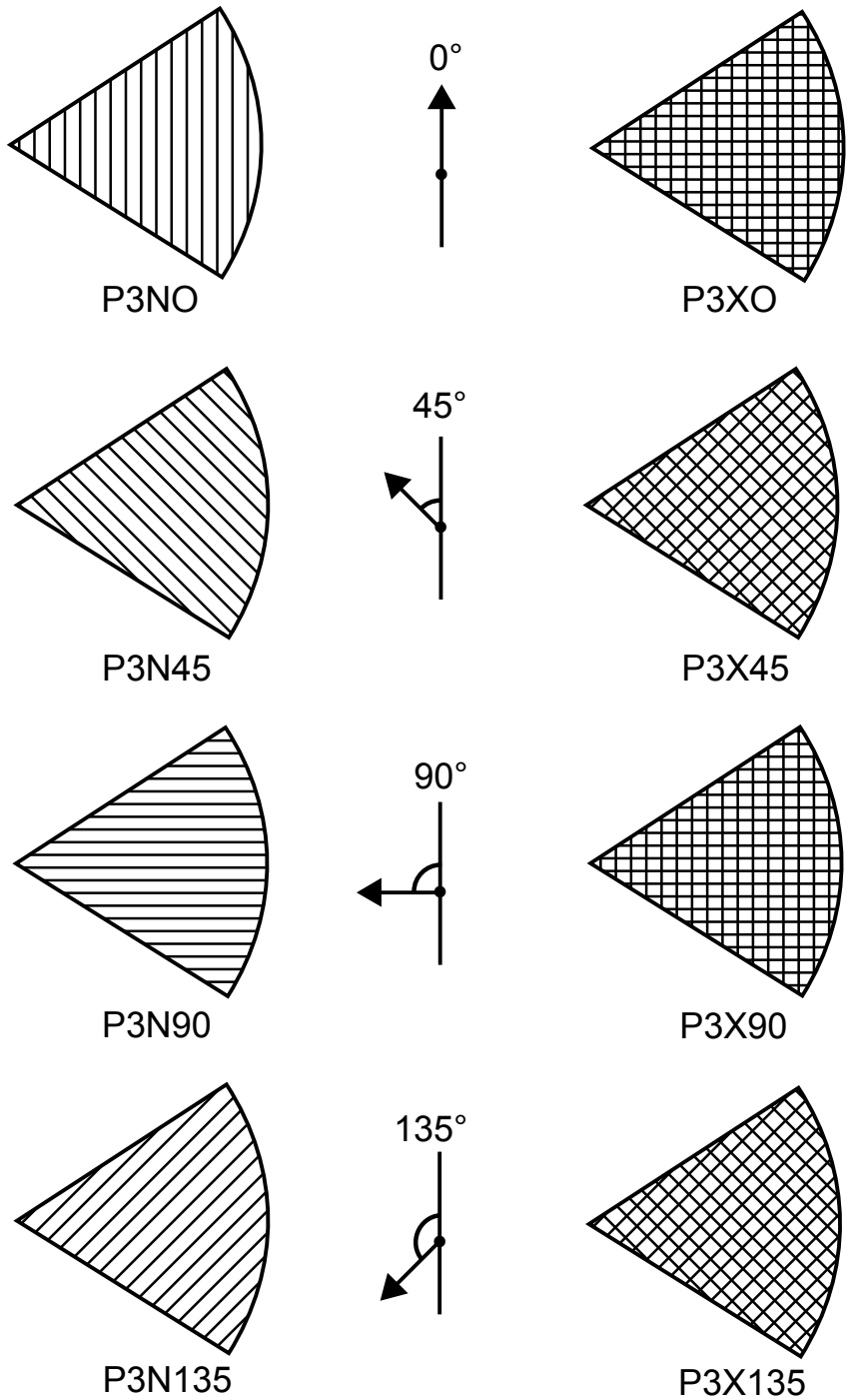
Angle specifies the angle of the pattern lines. Specify a number from 0 to 360. The number specifies the angle of the lines, measured in degrees from perpendicular to the radius of the slice. By default, *angle* is 0.

Note P*density<style<angle>>* is not supported by the ActiveX device driver.

The FILL= option in the PIE and STAR statements in the GCHART procedure overrides VALUE=.

Figure 25.45 on page 462 shows some typical pie and star patterns.

Figure 25.14 Pie and Star Patterns



If no valid patterns are available, default pie and star fill patterns are selected in this order:

1. PSOLID
2. P2N0
3. P2N90
4. P2X45
5. P4N0

6 P4N90

7 P4X90

Each fill is used once with every color in the color list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Alias V=

Restriction Partially supported by ActiveX

Note If you use hatch patterns and request a legend instead of slice labels, the patterns in the slices are oriented to be visually equivalent to the legend.

Details

Description: PATTERN Statement

PATTERN statements create PATTERN definitions that define the color and type of area fill for patterns used in graphs. These are the procedures and the graphics areas that they create that use PATTERN definitions:

GCHART
: color, fill pattern, or image for the bars in two-dimensional bar charts; color and fill pattern for the segments of three-dimensional bar charts, pie charts, and star charts.

GCONTOUR
: contour levels in contour plots

GPLOT
: areas beneath or between plotted lines

In addition, the SYMBOL statement and certain Annotate facility functions and macros can use pattern specifications. For details, see the “SYMBOL Statement” on page 470 and Chapter 28, “Using Annotate Data Sets,” on page 721.

You can use the PATTERN statement to control the fill and color of a pattern, and whether the pattern is repeated. There are various types of patterns:

- bar and block patterns
- plot patterns
- pie and star patterns

Pattern fills can be solid or empty, or composed of parallel or crosshatched lines. For two-dimensional bar charts, the PATTERN statement can specify images to fill horizontal or vertical bars. In addition, you can specify device-dependent hardware patterns for polygon, rectangle, and pie fills on devices that support hardware patterns.

If you do not create PATTERN definitions, SAS/GRAPH software generates them as needed and assigns them to your graphs by default. Generally, the default behavior is to rotate a solid pattern through the current color list. For details, see “About Default Patterns” on page 465.

Using the PATTERN Statement

How PATTERN Definitions Are Generated

PATTERN statements can be located anywhere in your SAS program. They are global and remain in effect until redefined, canceled, or until the end of your SAS session.

You can define up to 255 different PATTERN statements. A PATTERN statement without a number is treated as a PATTERN1 statement.

PATTERN statements generate one or more PATTERN definitions, depending on how the COLOR=, VALUE=, and IMAGE= options are used. For information about PATTERN definitions, see “Working with PATTERN Statements” on page 466, as well as the descriptions of “ COLOR=*pattern-color* | *_style_*” on page 455, “IMAGE=*fileref* | "*external-file*"” on page 456, and “VALUE=*plot-pattern* ” on page 459.

PATTERN definitions are generated in the order in which the statements are numbered, regardless of gaps in the numbering or the statement's position in the program. Although it is common practice, you do not have to start with PATTERN1, and you do not have to use sequential statement numbers.

PATTERN definitions are applied automatically to all areas of the graphics output that require patterns. When assigning PATTERN definitions, SAS/GRAPH starts with the lowest-numbered definition with an appropriate fill specification or with no fill specification. It continues to use the specified patterns until all valid PATTERN definitions have been used. Then, if more patterns are required, SAS/GRAPH returns to the default pattern rotation, but continues to outline the areas in the same color as the fill.

Altering or Canceling PATTERN Statements

PATTERN statements are additive. You can define a PATTERN statement and later submit another PATTERN statement with the same number. The new PATTERN statement redefines or cancels only the options that are included in the new statement. Options not included in the new statement are not changed and remain in effect. For example, assume you define PATTERN4 as follows:

```
pattern4 value=x3 color=red repeat=2;
```

This statement cancels only REPEAT= without affecting the rest of the definition:

```
pattern4 repeat=;
```

Add or change options in the same way. This statement changes the color of the pattern from red to blue:

```
pattern4 color=blue;
```

After all these modifications, PATTERN4 has these characteristics:

```
pattern4 value=x3 color=blue;
```

Cancel individual PATTERN statements by defining a PATTERN statement of the same number without options (a null statement):

```
pattern4;
```

Canceling one PATTERN statement does not affect any other PATTERN definitions. To cancel all current PATTERN statements, use the RESET= option in a GOPTIONS statement:

```
goptions reset=pattern;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current PATTERN definitions as well as other settings.

To display a list of current PATTERN definitions in the LOG window, use the GOPTIONS procedure with the PATTERN option:

```
proc goptions pattern nolist;
run;
```

About Default Patterns

How SAS/GRAPH Generates and Assigns PATTERN Definitions

When a procedure produces a graph that needs one or more patterns, SAS/GRAPH does one of the following actions:

- automatically generates the appropriate default patterns and outlines to fill the areas
- uses patterns, colors, and outlines that are defined by PATTERN statements, graphics options, and procedure options.

In order to understand how SAS/GRAPH generates and assigns patterns defined with PATTERN statements, it is helpful to understand how it generates and assigns default patterns. The following sections describe the default pattern behavior for all procedures. See "Working with PATTERN Statements" on page 466 for details about defining patterns.

How Default Patterns and Outlines Are Generated

In general, the default pattern that SAS/GRAPH uses is a solid fill. The default colors are determined by the current style and the device.

SAS/GRAPH uses default patterns when no PATTERN statements are defined. The default colors are determined by the current style and the device.

Because the system option-GSTYLE-is in effect by default, the procedure uses the style's default bar and block fill colors, plot line colors, widths, symbols, patterns, and outline colors when producing output. Specifically, SAS/GRAPH uses the default values when you do not specify any of the following:

- any PATTERN statements
- the CPATTERN= graphics option
- the COLORS= graphics options (that is, you use the device's default color list and it has more than one color)
- the COUTLINE= option in the action statement

If all of these conditions are true, then SAS/GRAPH performs the following operations:

- selects the first default fill for the appropriate pattern, which is always solid. It rotates once through the list of colors available in the current style, generating one solid pattern for each color. If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, then the procedure uses all the colors in the list to generate the patterns.
- uses the style's outline color to outline every patterned area.

If a procedure needs additional patterns, SAS/GRAPH selects the next default pattern fill appropriate to the graph and rotates it through the color list, skipping the foreground color as before. SAS/GRAPH continues in this fashion until it has generated enough patterns for the chart.

Things That Affect Default Patterns

Changing any of these conditions can change or override the default behavior:

- If you specify a color list with the COLORS= option in a GOPTIONS statement and the list contains more than one color, SAS/GRAPH rotates the default fills, beginning with SOLID, through that list. In this case, it uses every color, even if the foreground color is black (or white). The default outline color remains the foreground color.
- If you specify either COLORS=(*one-color*) or the CPATTERN= graphics option, the default fill changes from SOLID to the appropriate list of hatch patterns. SAS/GRAPH uses the specified color to generate one pattern definition for each hatch pattern in the list.

For a description of these options, see “Graphics Options and Device Parameters Dictionary” on page 577.

Working with PATTERN Statements

What You Can Specify with PATTERN Statements

With PATTERN statements, you can specify the following:

- the type of fill (VALUE=)
- the color of the fill (COLOR=)
- the images used to fill the bars in a two-dimensional chart (IMAGE=)
- how many times to apply the statement before using the next one (REPEAT=).

See “Displaying Images on Data Elements” on page 389 for information about filling the bars of two-dimensional bar charts with images using the PATTERN statement.

You can also use procedure options to specify the pattern outline color and the CPATTERN= graphics option to specify a default color for all patterns.

Whether you use PATTERN statement options alone or with each other affects the number and type of patterns your PATTERN statements generate. Depending on which options you use, you can explicitly specify every pattern used by your graphs. Or you can let the PATTERN statement generate a series of pattern definitions using either the color list or the list of default fills.

Explicitly Specifying Patterns

To explicitly specify all the patterns in your graph, you need to do one of the following for every pattern your graph requires:

- Provide a PATTERN statement that uses the COLOR= option to specify the pattern color, for example:

  ```
  pattern1 color=red;
  ```

  By default, the fill type is SOLID.

- Provide a PATTERN statement that uses both the COLOR= option and the VALUE= option to specify the fill, for example:

  ```
  pattern1 color=blue value=r3;
  ```

Including the COLOR= option in the PATTERN statement is the simplest way to ensure that you get exactly the patterns that you want. When you use the COLOR= option, the PATTERN statement generates exactly one PATTERN definition for that statement. If you also use the REPEAT= option, the PATTERN definition is repeated the specified number of times.

Generating Multiple Pattern Definitions

You can also use PATTERN statements to generate multiple PATTERN definitions. To do this use the VALUE= option to specify the type of fill you want but omit the COLOR= option. For example:

```
pattern1 value=r3;
```

In this case, the PATTERN statement rotates the R3 fill through all the colors in the color list. For more information about pattern rotation, see “Understanding Pattern Sequences” on page 468.

Selecting an Appropriate Pattern

The type of fill pattern that you specify depends on the type of graph that you are producing:

Table 25.3 *Fill Patterns for Graph Types*

Type of graph	Appropriate type of fill
Bar and block charts (PROC GCHART)	“ VALUE=*bar/block-pattern*” on page 457
Contour plots (PROC GCONTOUR)	“VALUE=*plot-pattern* ” on page 459
Pie and star charts (PROC GCHART)	“VALUE=*pie/star-pattern* ” on page 461

Note: You might specify a fill that is inappropriate for the type of graph that you are generating. In this case SAS/GRAPH ignores the PATTERN statement and

continues searching for a valid pattern. If it does not find a definition with a valid fill specification, it uses default patterns instead.

Controlling Outline Colors

Whenever you use PATTERN statements, the default outline color uses the style's outline color to outline every patterned area.

You can change the outline color of any pattern, whether the pattern is default or user-defined. Use the COUTLINE= option or the CEMPTY= option, or both, in the action statement that generates the chart.

The Effect of the CPATTERN= Graphics Option

Although the CPATTERN= graphics option is used most often with default patterns, it does affect the PATTERN statement. With default patterns (no PATTERN statements specified), it does the following:

- specifies the color for all patterns
- causes default patterns to use hatched fills instead of the default SOLID

In conjunction with the PATTERN statement that it does the following:

- With a PATTERN statement that specifies only a fill (VALUE=), the CPATTERN= option determines the color of that fill. For example, these statements produce two green, hatched patterns:

```
goptions cpattern=green;
pattern1 value=x3;
pattern2 value=x1;
```

- With a PATTERN statement that specifies only a color (COLOR=), the COLOR= option overrides the CPATTERN= color, but CPATTERN= causes the fill to be hatched, not the default SOLID. For example, these statements produce one red, hatched pattern:

```
goptions cpattern=green;
pattern1 color=red;
```

See also the description of “CPATTERN Graphics Option” on page 601.

Understanding Pattern Sequences

About Pattern Sequences

Pattern sequences are sets of PATTERN definitions that SAS/GRAPH automatically generates when a PATTERN statement specifies a fill but not a color. In this case, the specified fill is used once with every color in the color list. If the REPEAT= option is also used, the resulting PATTERN definitions are repeated the specified number of times.

Generating Pattern Sequences

SAS/GRAPH generates pattern sequences when a PATTERN statement uses VALUE= to specify a fill and all of the following conditions are also true:

- The COLOR= option is not used in the PATTERN statement.
- The CPATTERN= graphics option is not used.
- The color list, either default or user-specified, contains more than one color.

In this case, the PATTERN statement rotates the fill specified by the VALUE= option through every color in the color list. One PATTERN definition is generated for every color in the list. After every color has been used once, SAS/GRAPH goes to the next PATTERN statement. For example, suppose you specified the following color list and PATTERN statements for bar or block patterns:

```
goptions colors=(blue red green) ctext=black;
pattern1 color=red   value=x3;
pattern2 value=r3;
pattern3 color=blue  value=l3;
```

Here, `PATTERN1` generates the first PATTERN definition. `PATTERN2` omits the COLOR= option, so the specified fill is rotated through all three colors in the color list before the PATTERN3 statement is used. This table sequentially lists the color and fill of the PATTERN definitions that would be generated if nine patterns were required:

Table 25.4 *Color and Fill PATTERN Definitions for the First Nine Patterns*

Definition Number	Source	Characteristics	
		Color	Fill
1	PATTERN1	Red	x3
2	PATTERN2	Blue	r3
3	PATTERN2	Red	r3
4	PATTERN2	Green	r3
5	PATTERN3	Blue	l3
6	First default	Blue	Solid
7	First default	Red	Solid
8	First default	Green	Solid
9	Second default	Blue	x1

Notice that after all the PATTERN statements are exhausted, the procedure begins using the default bar and block patterns, beginning with SOLID. Each fill from the

default list is rotated through all three colors in the color list before the next default fill is used.

Repeating Pattern Sequences

If you use the REPEAT= option but not the COLOR= option, the sequence generated by cycling the definition through the color list is repeated the number of times specified by the REPEAT= option. For example, these statements illustrate the effect of the REPEAT= option on PATTERN statements both with and without explicit color specifications:

```
goptions colors=(red blue green);
pattern1 color=gold repeat=2;
pattern2 value=x1 repeat=2;
```

In the following table, `PATTERN1` is used twice and `PATTERN2` cycles through the list of three colors and then repeats this cycle a second time:

Table 25.5 *The Cyclical Use of PATTERN Statements*

Sequence Number	Source	Characteristics	
		Color	Fill
1	PATTERN1	Gold	Solid (first default)
2	PATTERN1	Gold	Solid (first default)
3	PATTERN2	Red	x1
4	PATTERN2	Blue	x1
5	PATTERN2	Green	x1
6	PATTERN2	Red	x1
7	PATTERN2	Blue	x1
8	PATTERN2	Green	x1

SYMBOL Statement

Defines the characteristics of symbols that display the data plotted by a PLOT statement used by the GBARLINE, GCONTOUR, and GPLOT procedures.

Used by: GBARLINE, GCONTOUR, GPLOT

Type: Global

Syntax

SYMBOL<1 …255>
<COLOR=*symbol-color* | *_style_*>
<MODE=EXCLUDE | INCLUDE>
<REPEAT=*number-of-times*>
<STEP=*distance*<*units*>>
<*appearance-options*>
<*interpolation-option*>
<SINGULAR=*n*>;

Summary of Optional Arguments

Appearance options

BWIDTH=*box-width*
: specifies the width of the box generated by either the INTERPOL=BOX or INTERPOL=HILOB option.

CI=*line-color* | *_style_*
: specifies a color for an interpolation line (GPLOT and GBARLINE) or a contour line (GCONTOUR).

CO=*color*
: specifies a color for outlines of filled areas, confidence limit lines, and staffs, boxes, and bars.

COLOR=*symbol-color* | *_style_*
: specifies a color for the entire definition, unless it is followed by a more explicit specification.

CV=*value-color* | *_style_*
: specifies a color for plot symbols, filled areas, and contour labels.

FONT="*font*"
: specifies the font for the plot symbol (GPLOT, GBARLINE) or contour labels (GCONTOUR) specified by the VALUE= option.

HEIGHT=*symbol-height*<*units*>
: specifies the height in number of units of plot symbols (GPLOT, GBARLINE) or contour labels (GCONTOUR).

LINE=*line-type*
: specifies the line type of plot or contour lines.

MODE=EXCLUDE | INCLUDE
: specifies that any interpolation method exclude or include data values that are outside the range of plot axes.

POINTLABEL<=(*label-description(s)*) | NONE>
: labels plot points.

STEP=*distance*<*units*>
: specifies the minimum distance between labels on contour lines

VALUE=*special-symbol* | *text-string* | SPECIAL | NONE
: specifies a plot symbol for the data points (GPLOT and GBARLINE).

WIDTH=*thickness-factor*
: specifies the thickness of interpolated lines (GPLOT) or contour lines (GCONTOUR).

General interpolation options

INTERPOL=JOIN
connects data points with straight lines.

INTERPOL=*map/plot-pattern*
specifies that a pattern fill the polygon that has been defined by the data points.

INTERPOL=NEEDLE
draws a vertical line from each data point to a horizontal line at the 0 value on the vertical axis or the minimum value on the vertical axis.

INTERPOL=NONE
suppresses any interpolation and, if the VALUE= option is not specified, also suppresses plot points.

INTERPOL=STEP<*placement*><J><S>
specifies that the data are plotted with a step function.

High-Low interpolation options

INTERPOL=BOX<*options*><00 …25>
produces box-and-whisker plots.

INTERPOL=HILO<C><*option*>
specifies that a solid vertical line connect the minimum and maximum Y values for each X value.

INTERPOL=STD<1 | 2 | 3><*variance*><*options*>
specifies that a solid line connect the mean Y value with ± 1, 2, or 3 standard deviations for each X.

Regression interpolation option

INTERPOL=R<*type*><0><CLM | CLI<50 – 99>>
specifies that a plot is a regression analysis.

Specification options

REPEAT=*number-of-times*
specifies the number of times that a SYMBOL definition is applied before the next SYMBOL definition is used.

SINGULAR=*n*
tunes the algorithm used to check for singularities.

Spline interpolation options

INTERPOL=L<*degree*><P><S>
specifies a Lagrange interpolation to smooth the plot line.

INTERPOL=SM<*nn*><P><S>
specifies that a smooth line is fit to data using a spline routine.

INTERPOL=SPLINE<P><S>
specifies that the interpolation for the plot line use a spline routine.

Optional Arguments

BWIDTH=*box-width*
specifies the width of the box generated by either the INTERPOL=BOX or INTERPOL=HILOB option. *Box-width* can be any number greater than 0. By default, the value of *box-width* is the same as the value of the WIDTH= option, whose default value is 1. Therefore, if you specify a WIDTH= value for the line

thickness of interpolated lines, and omit the BWIDTH= option, the width of the box changes according to the WIDTH= value.

Example "Example 4: Creating and Modifying Box Plots" on page 538

CI=*line-color* | *_style_*
specifies a color for an interpolation line (GPLOT and GBARLINE) or a contour line (GCONTOUR). The _STYLE_ value specifies the appropriate color based on the current style. If you omit the CI= option but specify the CV= option, the CI= option assumes the value of the CV= option. In this case, the CI= and CV= options specify the same color, which is the same as specifying the COLOR= option alone.

If you omit the CI= option, the color specification is searched for in this order:

1. the COLOR= option
2. the CV= option
3. the CSYMBOL= option in a GOPTIONS statement
4. each color in the color list sequentially before the next SYMBOL definition is used

See "Using Color" on page 500

Example "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

CO=*color*
specifies a color for the following:

- outlines of filled areas generated by the INTERPOL=*map/plot-pattern* option
- confidence limit lines generated by the INTERPOL=R *series* option
- staffs, boxes, and bars generated by the high-low interpolation methods: INTERPOL=HILO, INTERPOL=BOX, and INTERPOL=STD

If you omit the CO= option, the search order for a color specification depends on the interpolation method being used.

See "Using Color" on page 500

Examples "Example 5: Filling the Area between Plot Lines" on page 543

"Example 4: Creating and Modifying Box Plots" on page 538

COLOR=*symbol-color* | *_style_*
specifies a color for the entire definition, unless it is followed by a more explicit specification. For the GPLOT and GBARLINE procedures, this includes plot symbols, the plot line, confidence limit lines, and outlines. For the GCONTOUR procedure, this includes contour lines and labels. The _STYLE_ value specifies the appropriate color from the current style.

Using the COLOR= option is exactly the same as specifying the same color for both the CI= and CV= options.

If COLOR= precedes the CI= or CV= option in the same statement, the CI= or CV= option is used instead.

If you do not use the COLOR=, CI=, CV=, or CO= option, the color specification is searched for in this order:

1. the CSYMBOL= option in a GOPTIONS statement
2. each color in the color list sequentially before the next SYMBOL definition is used.

Alias C=

Style reference Color attribute of the GraphLabelText style element

Restriction Partially supported by ActiveX

Notes The ActiveX control does not support the COLOR= option when used with the GCONTOUR procedure.

You might not use a SYMBOL statement to specify a color for each symbol. Instead, you specify a color list in a GOPTIONS statement, then ActiveX assigns symbols and symbol colors differently than other devices. To ensure consistency on all devices, you should use SYMBOL statements to explicitly specify the symbols and symbol colors that you want to use in your plot.

See "Using Color" on page 500

CV=*value-color* | *_style_*
specifies a color for the following:

- plot symbols in the GPLOT procedure
- the filled areas generated by the INTERPOL=*map/plot-pattern* option
- contour labels in the GCONTOUR procedure

The _STYLE_ value specifies the appropriate color based on the current style. If you omit the CV= option but specify the CI=, the CV= option assumes the value of the CI= option. In this case, the CV= and CI= options specify the same color, which is the same as specifying the COLOR= option alone.

If you omit the CV= option, the color specification is searched for in this order:

1. the COLOR= option
2. the CI= option
3. the CSYMBOL= option in a GOPTIONS statement
4. each color in the color list sequentially before the next SYMBOL definition is used

Restriction Partially supported by ActiveX

Note The ActiveX control does not support using the CV= option with the GCONTOUR procedure.

See "Using Color" on page 500

Examples "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

"Example 4: Creating and Modifying Box Plots" on page 538

"Example 5: Filling the Area between Plot Lines" on page 543

FONT="*font*"

specifies the font for the plot symbol (GPLOT, GBARLINE) or contour labels (GCONTOUR) specified by the VALUE= option. The *font* specification must be enclosed in quotation marks and can include the `/bold` and `/italic` font modifiers.

By default, the symbol specified by the VALUE= option is taken from the special symbol table shown in Table 25.65 on page 492. To use symbols from the special symbol table, you must omit the FONT= option.

You might want to use a symbol that is not in that special symbol table. Specify the font containing the symbol and the character code or hexadecimal code of the symbol that you want to use. You can also specify text instead of special symbols. For example:

```
symbol font="Albany AMT" value="80"x;   /* hexadecimal code for the Euro symbol */
symbol font="Monotype Sorts" value="s";   /* character code for a filled triangle */
symbol font="Cumberland AMT/bo" value="F";   /* prints the letter F in bold */
```

To cancel a font specification and return to the default special symbol table, enter a null font specification:

```
symbol font= value=dot;
```

Alias F=

Restrictions Not supported by ActiveX

Only single-byte symbol characters are supported.

Note If the font is specified with the Unicode attribute,: then the font symbol values are processed as double-byte characters. Double-byte characters resolve to question marks.

See “ VALUE=*special-symbol* | *text-string* | SPECIAL | NONE” on page 491

“Specifying Plot Symbols” on page 499

“Specifying Special Characters Using Character and Hexadecimal Codes” on page 355

Example “Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1152

HEIGHT=*symbol-height<units>*

specifies the height in number of units of plot symbols (GPLOT, GBARLINE) or contour labels (GCONTOUR).

Alias H=

Restriction Partially supported by ActiveX

Notes With ActiveX, a symbol can be so small as to be invisible.

The ActiveX control does not support the HEIGHT= option when used with the GCONTOUR procedure.

The HEIGHT= option affects only the height of the symbols and labels on the plot; it does not affect the height of any symbols that might appear in a legend.

The HEIGHT option overrides the MarkerSize attribute in ODStyles. For more information about ODS styles, see the *SAS Output Delivery System: User's Guide*.

See the option " SHAPE=BAR(*width*<*units*>,*height*<*units*>) <*units*> | LINE(*length*) <*units*> | SYMBOL(*width*<*units*>,*height*<*units*>) <*units*>" on page 440 in the LEGEND statement

Examples "Example 4: Creating and Modifying Box Plots" on page 538

"Example 3: Rotating Plot Symbols through the Color List" on page 535

INTERPOL=BOX<*options*><00 …25>

produces box-and-whisker plots.

In all instances the bottom and top edges of the box are located at the sample 25th and 75th percentiles. The center horizontal line is drawn at the 50th percentile (median). Specifying options controls the appearance and the drawing of reference lines. Percentile specifications control the whiskers.

By default, INTERPOL=BOX, with no option or percentile specified. In this case the vertical lines, or whiskers, are drawn from the box to the most extreme point less than or equal to 1.5 interquartile ranges. (An interquartile range is the distance between the 25th and 75th sample percentiles.) Any value more extreme than this is marked with a plot symbol.

Values for *options* are one or more of these:

F
fills the box with the color specified by CV= and outlines the box with the color specified by CO=

J
joins the median points of the boxes with a line

T
draws tops and bottoms on the whiskers.

In addition, you can specify a percentile to control the length of the whiskers within the range 00 through 25. These are examples of percentile specifications and their effect:

00
specifies high or low extremes. INTERPOL=BOX00 is not the same as the default, INTERPOL=BOX.

01
specifies 1st percentile low, 99th high.

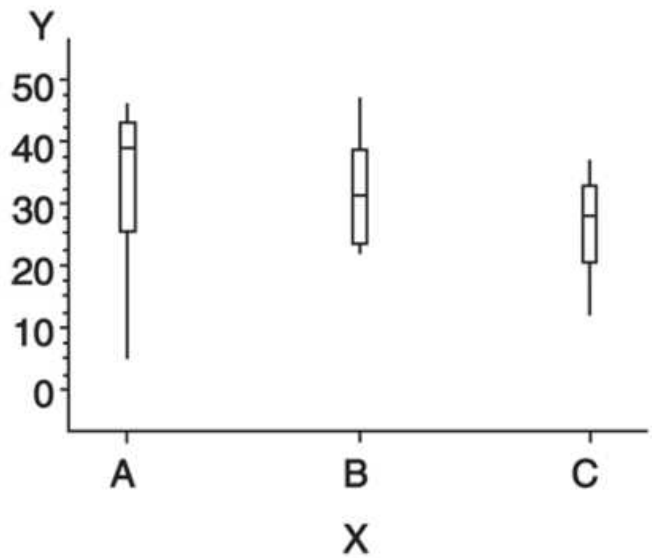
05
specifies 5th percentile low, 95th high.

10
specifies 10th percentile low, 90th high.

25
specifies 25th percentile low, 75th high. Because the box extends from the 25th to the 75th percentile, no whiskers are produced.

Figure 25.46 on page 477 shows the type of plot INTERPOL=BOX produces.

Figure 25.15 Box Plot



To increase the thickness of all box plot lines, including the box, whiskers, join line, and top and bottom ticks, use the WIDTH= option.

To increase the width of the box itself, use the BWIDTH= option. By default the value of the BWIDTH= option is the same as the value of the WIDTH= option. Therefore, if you specify a value for the WIDTH= option and omit the BWIDTH= option, the width of the box changes.

For a scatter effect with the box, use a multiple plot request, as in this example:

```
symbol1 i=none v=star color=green;
symbol2 i=box v=none color=blue;
proc gplot data=test;
   plot (y y)*x / overlay;
```

Alias I=

Default INTERPOL=BOX

Restriction You cannot use the GPLOT procedure PLOT statement option AREAS= with INTERPOL=BOX.

Notes You can use the HAXIS= or VAXIS= options in the PLOT statement. Or you can use the ORDER= option in an AXIS definition. Use any of these options to restrict the range of axis values. By default any observations that fall outside the axis range are excluded from the interpolation calculation. See the " MODE=EXCLUDE | INCLUDE" on page 487.

When using DEVICE=JAVAIMG with overlaid plots, different interpolations are supported per overlay unless any of the interpolations is BOX, HILO or STD. When any of these interpolations are encountered, the first interpolation specified becomes the only interpolation that is used for all overlays. All other interpolations are ignored.

Example "Example 4: Creating and Modifying Box Plots" on page 538

INTERPOL=HILO<C><*option*>

specifies that a solid vertical line connect the minimum and maximum Y values for each X value. The data should have at least two values of Y for every value of X. Otherwise, the single value is displayed without the vertical line.

By default, for each X value, the mean Y value is marked with a tick. This is shown in Figure 25.47 on page 478.

To specify high, low, close stock market data, include this option:

C

draws tick marks at the close value instead of at the mean value. Specifying C assumes that there are three values of Y (HIGH, LOW, and CLOSE) for every value of X. If more or fewer than three Y values are specified, the mean is ticked. The Y values can be in any order in the input data set.

In addition, you can specify one of these values for *option*:

B

connects the minimum and maximum Y values with bars instead of lines. Use the BWIDTH= option to increase the width of the bars.

J

joins the mean values or the close values (if HILOC is specified) with a line. This point is not marked with a tick mark. You cannot use the PLOT statement option AREAS= with INTERPOL=HILOJ.

T

adds tops and bottoms to each line.

BJ

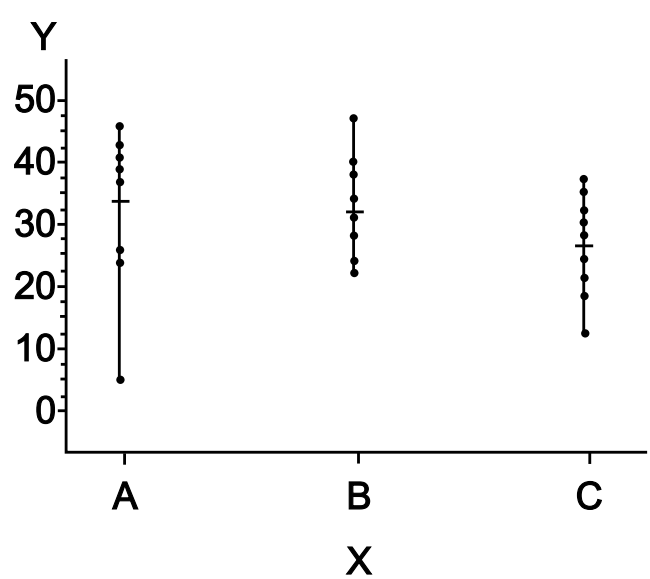
connects maximum and minimum values with a bar and joins the mean or close values.

TJ

adds tops and bottoms to the lines and joins the mean or close values.

Figure 25.47 on page 478 shows the type of plot INTERPOL=HILO produces. Plot symbols in the form of dots have been added to this figure.

Figure 25.16 *High-Low Plot*



When using DEVICE=JAVAIMG with overlaid plots, different interpolations are supported per overlay unless any of the interpolations is BOX, HILO or STD. When any of these interpolations are encountered, the first interpolation specified becomes the only interpolation that is used for all overlays. All other interpolations are ignored.

Alias I=

Note You can restrict the range of axis values by using the HAXIS= or VAXIS= option in a PLOT statement. Or use the ORDER= option in an AXIS definition. By default any observations that fall outside the axis range are excluded from the interpolation calculation. See the option " MODE=EXCLUDE | INCLUDE" on page 487.

Tip To increase the thickness of all lines generated by the INTERPOL=HILO option, use the WIDTH= option.

Example "Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

INTERPOL=JOIN

connects data points with straight lines. Points are connected in the order in which they occur in the input data set. Therefore, the data should be sorted by the independent (horizontal axis) variable.

If the data contain missing values, the observations are omitted. However, the plot line is not broken at missing values unless the SKIPMISS option is used.

Alias I=

See "SKIPMISS" on page 1329

"Missing Values" on page 1265

INTERPOL=L<*degree*><P><S>

specifies a Lagrange interpolation to smooth the plot line.

Specify one of these values for *degree*:

1
3
5

specifies the degree of the Lagrange interpolation polynomial. By default, *degree* is 1.

In addition, you can specify one or both of these:

P

specifies a parametric interpolation.

S

sorts a data set by the independent variable before plotting its data.

The Lagrange methods are useful chiefly when data consist of tabulated, precise values. A polynomial of the specified degree (1, 3, or 5) is fitted through the nearest 2, 4, or 6 points. In general, the first derivative is not continuous. If the values of the horizontal variable are not strictly increasing, the corresponding parametric method (L1P, L3P, or L5P) is used.

Specifying INTERPOL=L1P, INTERPOL=L3P, or INTERPOL=L5P results in a parametric Lagrange interpolation of degree 1, 3, or 5, respectively. Both the horizontal and vertical variables are processed with the Lagrange method and a

parametric interpolation of degree 1, 3, or 5, using the distance between points as a parameter.

Alias I=

INTERPOL=*map/plot-pattern*
specifies that a pattern fill the polygon that has been defined by the data points. Values for *map/plot-pattern* are as follows:

MEMPTY
specifies an empty pattern. EMPTY and E are valid aliases.

Alias ME

MSOLID
specifies a solid pattern. SOLID and S are valid aliases

Alias MS

M*density<style<angle>>*
specifies a shaded pattern.

Density specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest.

Style specifies the direction of pattern lines:

N
specifies parallel lines (the default)

X
specifies crosshatched lines.

Angle specifies the starting angle for parallel or crosshatched lines. Specify a number from 0 to 360. The number specifies the degree at which the parallel lines are drawn. By default, *angle* is 0 (lines are parallel to the horizontal axis).

Alias I=

Restriction The INTERPOL=*map/plot-pattern* option works only if the data are structured so that the data points and, consequently, the plot lines form an enclosed area. The plot lines should not cross each other.

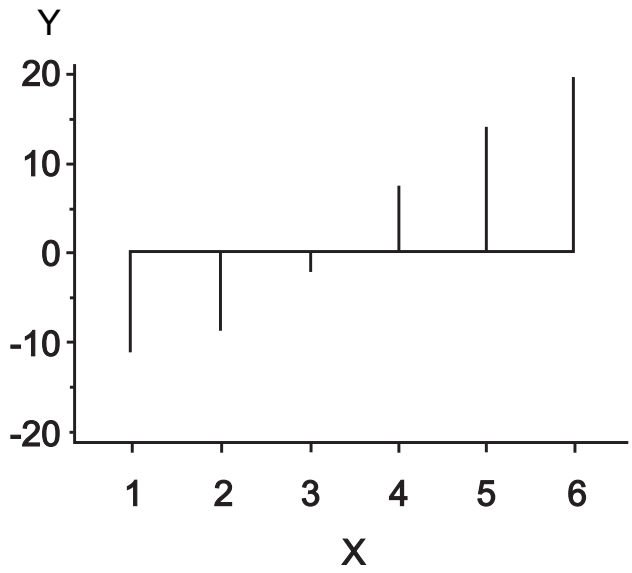
See "PATTERN Statement" on page 455

Example "Example 5: Filling the Area between Plot Lines" on page 543

INTERPOL=NEEDLE
draws a vertical line from each data point to a horizontal line at the 0 value on the vertical axis or the minimum value on the vertical axis. The horizontal line is drawn automatically.

Figure 25.48 on page 481 shows the type of plot INTERPOL=NEEDLE produces. Plot symbols are not displayed in this figure.

Figure 25.17 Needle Plot



Alias I=

Restriction You cannot use the PLOT statement option AREAS= with INTERPOL=NEEDLE

Tip Use the WIDTH= option or the HEIGHT= option in the SYMBOL statement to vary the width or the height of the plot needles, respectively. The specified width or height is also reflected in the legend entry. You can vary the width or height of the legend separately from the size of the plot needles. In this case, use the LEGEND statement option SHAPE="SYMBOL (*width*,*height*)<*units*>|(*width*<*units*>,*height*<*units*>) <*units*>" on page 441 when using the SYMBOL statement option INTERPOL=NEEDLE to modify the width or height values of the legend entry.

INTERPOL=NONE

suppresses any interpolation and, if the VALUE= option is not specified, also suppresses plot points. If no interpolation method is specified in a SYMBOL statement and if the graphics option INTERPOL= is not used, INTERPOL=NONE is the default.

Alias I=

Restriction You cannot use the PLOT statement option AREAS= with INTERPOL=NONE.

INTERPOL=R<*type*><0><CLM | CLI<50 – 99>>

specifies that a plot is a regression analysis. By default, the regression type is L, regression lines are not forced through plot origins, and confidence limits are not displayed.

The regression line is drawn in the line type specified in the LINE= option. By default, the line type of the regression line is 1.

Type indicates the type of regression. Specify one of these values for *type*:

L

requests linear regression representing the regression equation

$Y = \beta_0 + \beta_1 X$.

Q

requests quadratic regression representing the regression equation

$Y = \beta_0 + \beta_1 X + \beta_2 X^2$.

C

requests cubic regression representing the regression equation

$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$.

Note: When least-square solutions for the parameters are not unique, most devices use a quadratic equation by default for the interpolation. However, the ActiveX device driver might use a cubic solution.

CAUTION

You must specify *type* if you use 0, or CLI, or CLM, in order to achieve the expected results.

For example, the following interpolation request is correct:

```
I=RL0CLM95
```

while the following interpolation request is incorrect because type is missing although 0 is specified:

```
I=R0CLM95
```

and the following interpolation request is incorrect because type is missing although CLM is indicated:

```
I=RCLM95
```

To force the regression line through a (0,0) origin, specify the following:

0

eliminates the β_0 parameter, or intercept, from the regression equation. If the origin is at (0,0), also forces the regression line through the origin. For example, if you specify 0 for a linear regression, the plot line represents the equation

$Y = \beta_1 X$.

Note: You can force the regression line through the origin (0,0) when the data ranges do not place the origin at (0,0). Use the GPLOT procedure options HZERO and VZERO (ignored if the data contain negative values). Or use the HAXIS= and VAXIS= options to specify axes ranges from 0 to maximum data value. If the data ranges contain negative values and the HAXIS= and VAXIS= options specify ranges starting at 0, only values within the displayed range are used in the interpolation calculations.

To display confidence limits, specify one of these:

CLM

displays confidence limits for mean predicted values.

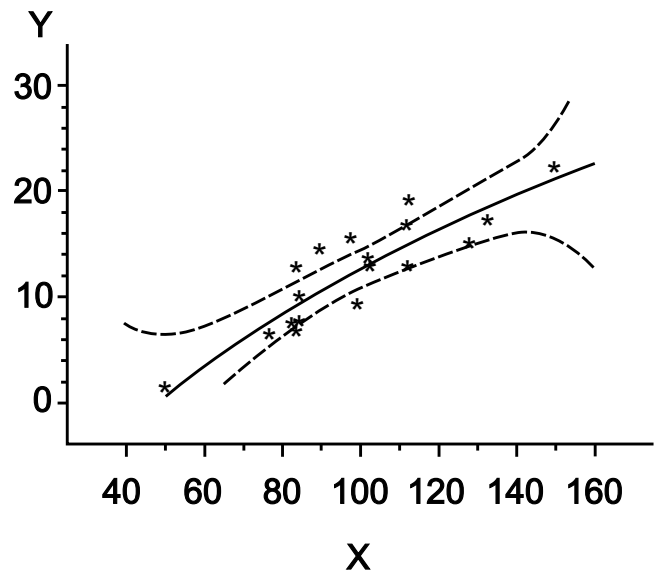
CLI
displays confidence limits for individual predicted values.

You can specify confidence levels from 50% to 99%. By default, the confidence level is 95%. Include a confidence level specification only if you use CLM or CLI.

The line type used for the confidence limit lines is determined by adding 1 to the values of LINE=. By default, the line type of confidence limit lines is 2.

Figure 25.49 on page 483 shows the type of plot INTERPOL=RCCLM95 produces (cubic regression analysis with 95% confidence limits).

Figure 25.18 *Plot of Regression Analysis and Confidence Limits*



Alias I=

Example "Example 4: Plotting Two Variables" on page 1345

INTERPOL=SM<*nn*><P><S>
specifies that a smooth line is fit to data using a spline routine. INTERPOL=SM is a method for smoothing noisy data. The points on the plot do not necessarily fall on the line.

The relative importance of plot values versus smoothness is controlled by *nn*. Values for *nn* are as follows:

0 …99
produces a cubic spline that minimizes a linear combination of the sum of squares of the residuals of fit and the integral of the square of the second derivative[1]. The greater the *nn* value, the smoother the fitted curve. By default, the value of *nn* is 0.

In addition, specify one or both of these:

P
specifies a parametric cubic spline.

S
sorts data by the independent variable before plotting.

Alias I=

1. C.H. Reinsch, "Smoothing by Spline Functions," *Numerische Mathematik* 10 (1967): 177–183.

INTERPOL=SPLINE<P><S>

specifies that the interpolation for the plot line use a spline routine. INTERPOL=SPLINE produces the smoothest line and is the most efficient of the nontrivial spline interpolation methods.

Spline interpolation smooths a plot line using a cubic spline method with continuous second derivatives[1] This method uses a piecewise third-degree polynomial for each set of two adjacent points. The polynomial passes through the plotted points and matches the first and second derivatives of neighboring segments at the points.

Specify one or both of these:

P

specifies a parametric spline interpolation method. This interpolation uses a parametric spline method with continuous second derivatives. Using the method described earlier for the spline interpolation, a parametric spline is fitted to both the horizontal and vertical values. The parameter used is the distance between points

$$t = \sqrt{(x^2 + y^2)}$$

If two points are so close together that the computations overflow, the second point is not used.

S

sorts a data set by the independent variable before plotting its data.

Alias I=

Note When points on the graph are out of range of the axis values, the curve is clipped. If an end point is out of range, no curve is drawn. Out-of-range conditions can be caused by restricting the range of axis values with the HAXIS= or VAXIS= option in the PLOT statement or the ORDER= option in an AXIS definition.

INTERPOL=STD<1 | 2 | 3><*variance*><*options*>

specifies that a solid line connect the mean Y value with ± 1, 2, or 3 standard deviations for each X. The sample variance is computed about each mean, and from it, the standard deviation s_y is computed. *Variance* can be one or both of these:

M

computes $s_{\bar{y}}$.

P

computes sample variances using a pooled estimate, as in a one-way ANOVA model.

In addition, specify one of these values for *options*:

B

connects the minimum and maximum Y values with bars instead of lines.

J

connects the means from bar to bar with a line.

T

adds tops and bottoms to each line.

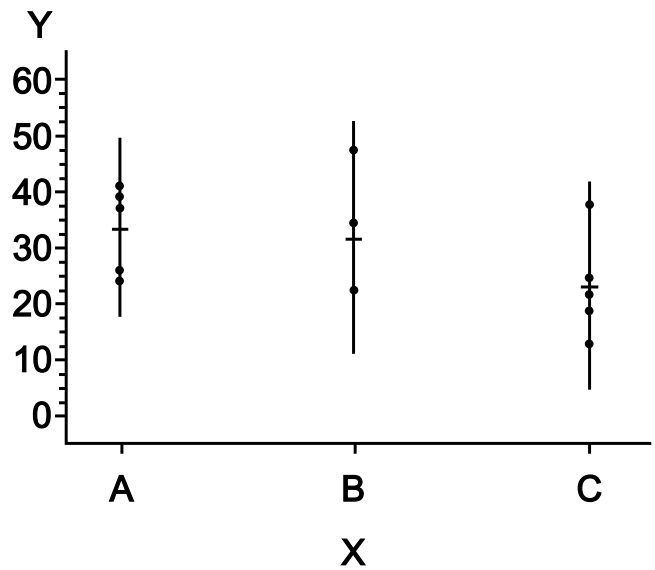
1. Stephen M. Pizer,*Numerical Computing and Mathematical Analysis*, Chicago: Science Research Associates, Inc. (1975), Chapter 4.

BJ
connects maximum and minimum values with a bar and joins the mean values.

TJ
adds tops and bottoms to the lines and joins the mean values.

Figure 25.50 on page 485 shows the type of plot INTERPOL=STD produces. A horizontal tick is drawn at the mean. Plot symbols in the form of dots have been added to this figure.

Figure 25.19 *Plot of Standard Deviations*



You can restrict the range of axis values by using the HAXIS= or VAXIS= option in a PLOT statement. Or use the ORDER= option in an AXIS definition. By default any observations that fall outside the axis range are excluded from the interpolation calculation. See the " MODE=EXCLUDE | INCLUDE" on page 487.

To increase the thickness of all lines generated by the INTERPOL=STD option, use the WIDTH= option.

You cannot use the PLOT statement option AREAS= with INTERPOL=STD.

When using DEVICE=JAVAIMG with overlaid plots, different interpolations are supported per overlay unless any of the interpolations is BOX, HILO or STD. When any of these interpolations are encountered, the first interpolation specified becomes the only interpolation that is used for all overlays. All other interpolations are ignored.

Alias I=

Notes By default, two standard deviations are used.

By default, the vertical axis ranges from the minimum to the maximum Y value in the data. The requested number of standard deviations from the mean covers a range of values that might exceed the maximum or is less than the minimum. In this case the STD lines are cut off at the minimum and maximum Y values. When this cutoff occurs, rescale the axis using VAXIS= in the PLOT statement or ORDER= in an AXIS definition so that the STD lines are shown.

INTERPOL=STEP<*placement*><J><S>

specifies that the data are plotted with a step function. By default, the data point is on the left of the step, the steps are not joined with a vertical line, and the data are not sorted before processing.

Specify one of these values for *placement*:

L
displays the data point on the left of the step.

R
displays the data point on the right of the step.

C
displays the data point in the center of the step.

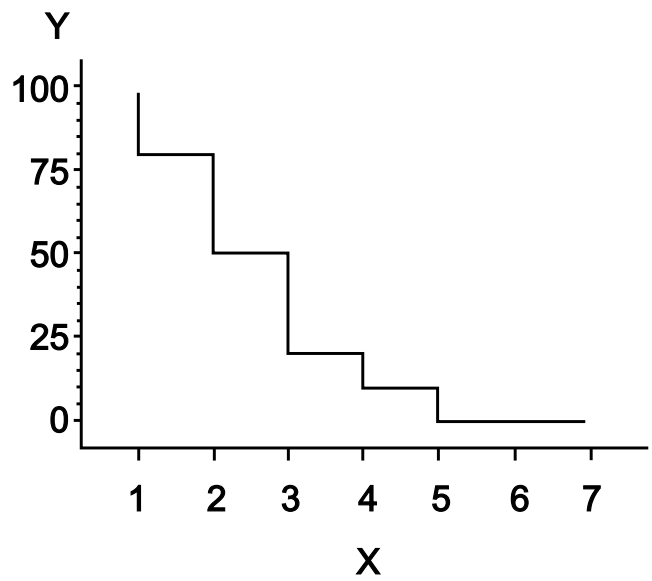
In addition, specify one or both of these:

J
produces steps joined with a vertical line.

S
sorts unordered data by the independent variable before plotting.

Figure 25.51 on page 486 shows the type of plot that INTERPOL=STEPJR produces. Plot symbols in the form of dots have been added to this figure.

Figure 25.20 *Step Plot*



Alias I=

Notes Sometimes a step is retraced in order to locate its center point. In this case the GIF, JPEG, PNG, ACTXIMG,and JAVAIMG devices treat this as effectively not drawing that part of the step at all.

The ActiveX devices does not support the S option. For this device, use the SORT procedure or the ORDER= option in an AXIS statement to sort your data by the independent variable before calling the GPLOT or GBARLINE procedure.

LINE=*line-type*

specifies the line type of the plot line in the GPLOT procedure, or the contour line in the GCONTOUR procedure:

1
specifies a solid line.

2 …46
specifies a dashed line.

Line types are shown in Figure 25.52 on page 503. By default, LINE=1.

Alias L=

Restriction Partially supported by ActiveX

Notes This option overrides the LineStyle attribute in graph styles.

The ActiveX control does not support GCONTOUR.

MODE=EXCLUDE | INCLUDE

specifies that any interpolation method exclude or include data values that are outside the range of plot axes. By default, MODE=EXCLUDE prevents values outside the axis range from being displayed.

You can restrict the range of axis values by using the HAXIS= or VAXIS= option in a PLOT statement. Or use the ORDER= option in an AXIS definition. By default any observations that fall outside the axis range are excluded from the interpolation calculation. Using these options to control value ranges has a particularly noticeable effect on the high-low interpolation methods, which include INTERPOL=HILO, INTERPOL=BOX, and INTERPOL=STD. Regression analysis also represents only part of the original data.

Restrictions Partially supported by ActiveX

When plotting multiple plot lines on the same output graph, MODE=INCLUDE is not supported by the ActiveX device driver. Specifying a PNG or GIF device driver generates the graph with values that are interpolated but not plotted when those values fall outside of the axis range.

See "Values Out of Range" on page 1265

POINTLABEL<=(*label-description(s)*) | NONE>

labels plot points. The labels always use the format that is assigned to the variable or variables whose values are used for the labels. POINTLABEL without any specified description labels results in no displayed labels for the points. NONE suppresses the point labels. *Label-description(s)* can be used to change the variable whose values are used to label points. This value can also change features of the label text, such as the color, font, or size of the text.

Label-description(s) can be one or more of these:

COLOR=*text-color*
specifies the color of the label text.

The POINTLABEL COLOR= specification searches for a color in this order:

1. the color specified by the COLOR= option of the POINTLABEL= option of the SYMBOL statement

2 the color specified by the CTEXT= option in the PLOT statement of the GPLOT procedure

3 the color specified by the CTEXT= option in the GOPTIONS statement

4 the first color in the color list of the GOPTIONS statement

Alias C=

Default The default is the color of the axis label.

Note If you do not specify a color in a SYMBOL statement, the symbol definition is rotated through the color list before the next SYMBOL statement is used. Take the case where your plot contains multiple plot lines and you want to limit your POINTLABEL specification to a single line. You must specify a color in the SYMBOL statement that contains the POINTLABEL description.

DROPCOLLISIONS
NODROPCOLLISIONS
specify DROPCOLLISIONS to drop new labels if they collide with a label already in use. Specify NODROPCOLLISIONS to retain all labels. The default is DROPCOLLISIONS.

The algorithm for the placement of markers tries to avoid placing labels such that they collide. If the algorithm is unable to avoid a collision, then the default DROPCOLLISIONS is to drop the new label, whereas NODROPCOLLISIONS retains even colliding labels.

FONT=*font* | NONE
specifies the font for the text. See Chapter 22, “Specifying Fonts in SAS/GRAPH Programs,” on page 349 for details about specifying *font*. If you omit FONT=, a font specification is searched for in this order:

1 the FTEXT= option in a GOPTIONS statement

2 the default hardware font, NONE

Alias F=

Restriction Only single-byte symbol characters are supported.

Note If the font is specified with the Unicode attribute, then the font symbol values are processed as double-byte characters. Double-byte characters resolve to question marks. POINTLABEL specifications that mix double- and single-byte characters are processed as if the entire string consists of double-byte characters.

HEIGHT=*text-height <units >*
specifies the height of the text characters in number of units. By default, HEIGHT=1 CELL. If you omit HEIGHT=, a text height specification is searched for in this order:

1 the HTEXT= option in a GOPTIONS statement

2 the default value, 1

Alias H=

JUSTIFY=CENTER | LEFT | RIGHT
specifies the horizontal alignment of the label text. The default is CENTER. The location of the point label is relative to the location of the corresponding data point.

Alias J=C | L | R

POSITION=TOP | MIDDLE | BOTTOM
specifies the vertical placement of the label text. The default is TOP. The location of the point label is relative to the location of the corresponding data point.

"#*var*1<:#*var*2 <$*char*>>"
specifies the variable or variables whose values label the plot points, and the delimiter between more than one variable. The variable specification must be enclosed in either single or double quotation marks. The first specified variable must be prefixed with a number sign (#). If a second variable is specified, it must be prefixed with a colon and a number sign (:#). When you specify two variables, you can also specify the character to display as the delimiter between variable values in the plot label.

By default, if the POINTLABEL= option is specified without naming a label variable, the Y values label the plot points. You can change the default by using "#*var*" to specify a different variable whose values should label the points. For example, you might specify the name of the X variable. The following option specifies the variable SALES as the variable whose values label plot points:

```
POINTLABEL=("#sales")
```

Alternatively, you can label the plot points with the values of two variables, in either order. The order in which you specify the variables determines the order that the values are displayed in the label. The following option specifies variables HEIGHT and WEIGHT; which in the label displays the value for HEIGHT and then the value for WEIGHT:

```
POINTLABEL=("#height:#weight")
```

By default, when you specify two variables, a colon (:) is displayed in the label to separate the variable values. To change the character that is displayed as the delimiter, use the $ syntax to specify an alternative character. The following option specifies a vertical bar (|) as the delimiter in the label:

```
POINTLABEL=("#height:#weight $|")
```

The $ syntax must be within the same quotation marks as the variable specification. The $ specification can precede or follow the variable specification, but it must be separated from the variable specification by at least one space.

Specify as many label-description suboptions as you want. Enclose them all within a single set of parentheses, and separate each suboption from the others by at least one space.

Restrictions When creating output using the ACTIVEX device, the variables that you specify must be for the plot's X and Y variables. Specifying any other variables causes unexpected labeling.

There is a 16-character length limit for each variable. A maximum character length limit of thirty-three characters is possible. This

can be composed of X and Y variables, any other valid data set variable, and a separator as required.

Notes Specifying a delimiting character with the $ changes only the character that is displayed in the label. It does not change the syntax of the variable specification, which requires a colon and number sign (:#) to precede the second variable.

Creating multiple plots that share the same or close-ranging data points, along with specifying appearance suboptions, can result in multiple data point labels.

Depending on the appearance suboptions that you use for point labels, the labels can overlap other elements on the graph. The HEIGHT= suboption is an example. When this occurs, a warning appears in the SAS log.

The algorithm for placing markers on the graph repositions point labels that are overwriting a graphics output area boundary. When this occurs, a note appears in the SAS log.

REPEAT=*number-of-times*

specifies the number of times that a SYMBOL definition is applied before the next SYMBOL definition is used. The behavior of REPEAT= depends on whether any of the SYMBOL color options (CI=, CV=, CO=, and COLOR=) or the CSYMBOL= graphics option also is used:

- If any SYMBOL color option also is used in the SYMBOL definition, that SYMBOL definition is repeated the specified number of times in the specified color.
- If no SYMBOL color option is used but the CSYMBOL= graphics option is currently in effect, the SYMBOL definition is repeated the specified number of times in the specified color.
- If no SYMBOL statement color options are used and the CSYMBOL= graphics option is not used, the SYMBOL definition is cycled through each color in the color list. The entire group generated by this cycle then repeats the number of times specified by the REPEAT= option. Thus, the total number of iterations of the SYMBOL definition depends on the number of colors in the current color list.

Alias R=

Default REPEAT=1.

Restrictions Partially supported by ActiveX

The ActiveX control does not support GCONTOUR.

See “Using the SYMBOL Statement” on page 496

SINGULAR=*n*

tunes the algorithm used to check for singularities. The default value is machine dependent but is approximately 1E-7 on most machines. This option is rarely needed.

STEP=*distance<units>*

specifies the minimum distance between labels on contour lines The value of *distance* must be greater than zero.

When you use the STEP= option, specify the minimum distance that you want between labels. The option then calculates how many labels it can fit on the contour line, taking into account the length of the labels and the minimum distance that you specified. Once it has calculated how many labels it can fit while retaining the minimum distance between them, it places the labels, evenly spaced, along the line. Consequently, the space between labels can be greater than what you specify, although it is never less.

In general, to increase the number of labels from the default, reduce the value of *distance*.

There are times when the procedure cannot write the label at a particular location on the contour. For example, when the contour line makes a sharp turn, the label might be placed farther along the line or omitted. If labels are omitted, a note appears in the log. Specifying a low value for the GCONTOUR procedure's TOLANGLE= option can also cause labels to be omitted. This specification forces the procedure to select smoother labeling locations, which might not be available on some contours.

Default STEP=65PCT

Restriction Not supported by ActiveX

Note If you specify units of PCT or CELLS, the STEP= option calculates the distance between the labels based on the width of the graphics output area, not the height. For example, if you specify STEP=50PCT and if the graphics output area is 9 inches wide, the distance specified is 4.5 inches. A value less than 10% is ignored and 10% is used instead.

Example "Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend" on page 1152

VALUE=*special-symbol* | *text-string* | SPECIAL | NONE
specifies a plot symbol for the data points (GPLOT and GBARLINE).

- If you omit the SYMBOL statement, plot points are generated using the default plot symbols. If you specify a SYMBOL statement, but do not specify the VALUE= option, the plot symbols are suppressed on the graph and in the legend.
- specifies contour-label text in a contour plot (GCONTOUR). By default with the AUTOLABEL option, GCONTOUR labels contour lines with the contour variable's value at that contour level.
- VALUE=SPECIAL enables you to define unique special symbols for up to 12 plots (GPLOT) in one SYMBOL statement. Some of the symbols include the following: CIRCLE, DIAMOND, DIAMONDFILLED, DOT, HASH, PAW, POINT, PLUS, SQUARE, SQUAREFILLED, STAR, TRIANGLE, TRIANGLEFILLED, X, Y, and Z. See Table 25.65 on page 492 for the complete list. This option is useful when the number of plots in your graph is variable. Rather than writing a SYMBOL statement for each possible plot, you can use VALUE=SPECIAL in one SYMBOL statement to define symbols for up to 12 individual plots.
- VALUE=NONE suppresses plot symbols at the data points, or labels on the contour lines. You can set the VALUE=NONE option independent of the INTERPOL= option.

Values for *special-symbol* are the names and characters shown in Table 25.65 on page 492. The special symbol table can be used only if the FONT= option is not used or a null value is specified:

```
font=,
```

This means that you cannot use VALUE=*special-symbol* or VALUE=SPECIAL if you are using the FONT= option.

Table 25.6 *Supported Plot Symbols*

Value	Symbol	Value	Symbol
PLUS	+	$	◇
X	×	%	♣
STAR	✳	&	☘
SQUARE	□	' (single quote)	⚜
SQUAREFILLED	■	=	☆
DIAMOND	◇	- (hyphen)	⊙
DIAMONDFILLED	◆	@	♁
TRIANGLE	△	* (asterisk)	♀
TRIANGLEFILLED	▲	+	⊕
HASH	#	>	♂
Y	Y	. (period)	♃
Z	Z	<	♄
PAW	⁝	, (comma)	♗
POINT	·	/	♆
DOT	●	?	ℙ
CIRCLE	○	(	☾
_ (underscore)	⌑	)	♉

Value	Symbol	Value	Symbol
" (double quote)	♤	:	✳
#	♡		

Note: The words or special characters in the VALUE= column are entered exactly as shown.

To specify a single quotation mark, you must enclose it in double quotation marks:

```
value="'"
```

To specify a double quotation mark, you must enclose it in single quotation marks:

```
value='"'
```

In some operating environments, punctuation characters might require single quotation marks.

If you use VALUE=*text-string* to specify a plot symbol, you must also use the FONT= option to specify a symbol font or a text font. If you specify a symbol font, the characters in the string are character codes for the symbols in the font. If you specify a text font, the characters in the string are displayed. If you specify a text string containing quotation marks or blanks, enclose the string in single quotation marks.

For example, if you specify this statement, the plot symbol is the word "plus" instead of the symbol +:

```
symbol font=swiss value=plus;
```

ActiveX supports the following characters from the Figure A3.5 on page 1665 for *special-symbol* and SPECIAL:

Table 25.7 *Marker-font Symbols Supported by ActiveX*

Character	Aliases
Marker	Cone, Pyramid, Default
Square	Cube
Star	
Circle	Sphere, Dot, Balloon
Plus	Cross
Flag	Y
X	

Character	Aliases
Prism	Z
Spade	“
Heart	#
Diamond	$
Club	%
Hexagon	Paw
Cylinder	Hash

Note: You might not use a SYMBOL statement to specify a color for each symbol, but you do specify a color list in a GOPTIONS statement. In this case, ActiveX assigns colors to symbols differently than do the other device drivers. To ensure consistency on all devices, for VALUE=*special-symbol* or VALUE=*text-string*, you should specify the desired color of each symbol. If you do not specify a symbol color, SAS/GRAPH uses the first default color and the first symbol. It uses each color in the list of default colors until the list is exhausted. SAS/GRAPH then selects the next symbol and begins again with the first default color. It rotates the new symbol through the list of default colors before selecting another symbol. It continues selecting new symbols and colors until no more symbols are needed.

For VALUE=SPECIAL, the way in which symbols and their colors are assigned is different from VALUE=*special-symbol* and VALUE=*text-string*. For example, you specify VALUE=SPECIAL when generating symbols for a plot. Rather than rotating each symbol through the list of colors, SAS/GRAPH rotates through the 12 special symbols and assigns each symbol a color from the color list. The symbols and the symbol colors that are used in a plot depend on the following:

- whether a symbol color is specified in the SYMBOL statement or in the GOPTIONS statement
- the number of colors in the color list when a symbol color is not specified

Alias V=

Restriction Partially supported by ActiveX

Notes For ActiveX output, the VALUE= option is not supported when INTERPOL=HILO or INTERPOL=STD. You can use the OVERLAY option with GPLOT to get symbols to appear on the data points.

The VALUE option overrides the MarkerSymbol attribute in graph styles.

See “Symbol Sequences Generated from SYMBOL Statements” on page 504

" FONT=*font* | NONE " on page 488

"Specifying Plot Symbols" on page 499

"About Special Fonts and Symbols in ActiveX" on page 202 for a description of the special symbols that the ActiveX devices support and the value that you must specify in the SYMBOL statement VALUE= option for each.

Examples

"Example 3: Rotating Plot Symbols through the Color List" on page 535

"Example 4: Creating and Modifying Box Plots" on page 538

"Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend" on page 1152

WIDTH=*thickness-factor*

specifies the thickness of interpolated lines (GPLOT) or contour lines (GCONTOUR). *thickness-factor* is a number. The thickness of the line increases directly with *thickness-factor*. By default, WIDTH=1.

WIDTH= also affects all the lines in box plots (INTERPOL=BOX), high-low plots with bars (INTERPOL=HILOB), and standard deviation plots (INTERPOL=STD). It also affects the outlines of the area generated by the AREAS= option in the PLOT statement of the GPLOT procedure.

Alias W=

Style reference LineThickness attribute of the GraphAxisLines element

Restriction Partially supported by ActiveX

Notes By default, the value specified by WIDTH= is used as the default value for the BWIDTH= option. For example, specifying WIDTH=6 also sets BWIDTH= to 6 unless you explicitly assign a value to the BWIDTH= option.

ActiveX does not provide the same measure of control for width as the other SAS/GRAPH device drivers. Measurements are translated to pixels rather than a percentage. For the JAVAIMG, ACTIVEX, and ACTXIMG devices, the maximum width is 6.

Examples

"Example 1: Ordering Axis Tick Marks with SAS Date Values" on page 527

"Example 4: Creating and Modifying Box Plots" on page 538

Details

Description: SYMBOL Statement

SYMBOL statements create SYMBOL definitions. These definitions are used by the GPLOT, GBARLINE, and GCONTOUR procedures. For GPLOT and GBARLINE procedures, SYMBOL definitions control the following:

- the appearance of plot symbols and plot lines, including bars, boxes, confidence limit lines, and area fills
- interpolation methods
- how plots handle data out of range

For the GCONTOUR procedure, SYMBOL definitions control the following:

- the appearance and text of contour labels
- he appearance of contour lines

If you create SYMBOL definitions, they are automatically applied to a graph by the procedure. If you do not create SYMBOL definitions, these procedures generate default definitions and apply them as needed to your plots.

Using the SYMBOL Statement

How SYMBOL Definitions Are Generated

A SYMBOL statement specifies one or more options that indicate the color and other attributes used by the GPLOT, GBARLINE, and GCONTOUR procedures. For GPLOT and GBARLINE, the main attributes include the plot symbol, interpolation method, and type of plot line. For GCONTOUR, the main attributes include the type of contour lines used and the text used to label those lines.

Note: SYMBOL statements can be applied only to contour plots when the AUTOLABEL option is specified on GCONTOUR.

You can define up to 255 different SYMBOL statements. A SYMBOL statement without a number is treated as a SYMBOL1 statement.

SYMBOL definitions can be defined anywhere in your SAS program. They are global and remain in effect until canceled or until you end your SAS session. Once defined, SYMBOL definitions can be used as follows:

- assigned by default by GPLOT or explicitly selected with the plot request
- used by GCONTOUR to control the labels and attributes of contour lines

SYMBOL statements generate one or more symbol definitions, depending on how color is used and whether a plot symbol or type of contour line is specified. For more information, see “Controlling Consecutive SYMBOL Statements” on page 497 and “Using Generated Symbol Sequences” on page 503.

Although it is common practice, you do not have to start with SYMBOL1, and you do not have to use sequential statement numbers. When assigning SYMBOL

definitions, SAS/GRAPH software starts with the lowest-numbered definition and works upward, ignoring gaps in the numbering.

Altering or Canceling SYMBOL Statements

SYMBOL statements are additive. You can define a SYMBOL statement and later submit another SYMBOL statement with the same number. The new SYMBOL statement defines or cancels only the options that are included in the new statement. Options that are not included in the new statement are not changed and remain in effect.

Note: An exception to this rule is presented by POINTLABEL= suboptions that are not carried over to subsequent SYMBOL statements.

Assume you define SYMBOL4 as follows:

```
symbol4 value=star cv=red height=4;
```

The following statement cancels only HEIGHT= without affecting the rest of the definition:

```
symbol4 height=;
```

Add or change options in the same way. This statement adds an interpolation method to SYMBOL4:

```
symbol4 interpol=join;
```

This statement changes the color of the plot symbol from red to blue:

```
symbol4 cv=blue;
```

After all these modifications, SYMBOL4 has these characteristics:

```
symbol4 value=star cv=blue interpol=join;
```

Cancel individual SYMBOL statements by defining a SYMBOL statement of the same number without options (a null statement):

```
symbol4;
```

Canceling one SYMBOL statement does not affect any other SYMBOL definitions. To cancel all current SYMBOL statements, use the RESET= option in a GOPTIONS statement:

```
goptions reset=symbol;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current SYMBOL definitions as well as other settings.

To display current SYMBOL definitions in the LOG window, use the GOPTIONS procedure with the SYMBOL option:

```
proc goptions symbol nolist;
run;
```

Controlling Consecutive SYMBOL Statements

You can specify consecutively numbered SYMBOL statements and you might want SAS/GRAPH to use each definition only once. Use color specifications to ensure that each SYMBOL statement generates only one symbol definition. You can do the following actions:

- specify colors in each SYMBOL statement, using the COLOR=, CI=, CV=, or CO= options. This method lets you explicitly assign colors for each definition. For example, these statements generate four definitions:

```
symbol1 value=star color=green;
symbol2 value=square color=yellow;
symbol3 value=special color=cyan;
symbol4 value=special color=orange;
```

- specify a default color for all SYMBOL statements using the CSYMBOL= option in the GOPTIONS statement. This method makes it easy to specify the same color for each definition when you do not need more explicit color specifications.
- limit the color list to a single color using the COLORS= option in the GOPTIONS statement. This method makes it easy to specify the same color for each definition when you want the color to apply to other definitions also, such as PATTERN definitions.

For more information about specifying colors for symbol definitions, see "Using Color" on page 500.

You do not have to use a color to limit a SYMBOL statement to a single symbol definition. In this case, SAS/GRAPH generates multiple symbol definitions from that statement by rotating the current definition through the color. For more details, see "Using Generated Symbol Sequences" on page 503. Note that SAS/GRAPH uses symbol definitions in the order in which they are generated. This can result in the nth symbol definition applied to a graph to not necessarily correspond to the SYMBOL*n* statement.

For example, assuming that no color is specified on the CSYMBOL= graphics option, these statements generate seven definitions:

```
goptions colors=(red blue green);
symbol1 value=star;
symbol2 value=square color=yellow;
symbol3 value=special;
```

Because no color is specified on SYMBOL1, SAS/GRAPH rotates the symbol definition through the color list, which has three colors. Thus, SYMBOL1 defines the first three applied symbol definitions. The SYMBOL2 definition specifies a color for the symbol, which means that it generates only one symbol. Thus, SYMBOL2 defines the fourth. Finally, SYMBOL3 specifies SPECIAL as its value, which means that it rotates through the list of 12 special symbols using the colors in the color list. Thus, SYMBOL3 defines the last three symbol definitions. Here is a summary of the symbols that are generated by these SYMBOL statements.

Table 25.8 *Example Output Using 3 Symbol Definitions*

Sequence Number	Source	Characteristics: Color	Symbol
1	SYMBOL1	Red	Star
2	SYMBOL1	Blue	Star
3	SYMBOL1	Green	Star
4	SYMBOL2	Yellow	Square

Sequence Number	Source	Characteristics: Color	Symbol
5	SYMBOL3	Red	Dot
6	SYMBOL3	Blue	Plus
7	SYMBOL3	Green	X

In this case, if a graph needs only three symbols, the SYMBOL2 and SYMBOL3 definitions are not used.

To make the nth applied symbol definition correspond to the SYMBOL*n* statement, limit each SYMBOL statement to a single color. You can use one of the techniques listed at the beginning of this section.

Setting Definitions for PROC GPLOT and PROC GBARLINE

Specifying Plot and Interpolation Methods

The following topics apply only for SYMBOL statements used with PROC GPLOT and PROC GBARLINE:

- specifying plot symbols
- specifying default interpolation methods
- sorting data with spline interpolation

Specifying Plot Symbols

The VALUE= option specifies the plot symbols that PROC GPLOT and PROC GBARLINE uses to mark the data points on a plot. Plot symbols can be in the following forms:

- special symbols as shown in Table 25.65 on page 492
- characters from symbol fonts
- text strings

By default, the plot symbol is the DOT symbol for the ActiveX device or the + symbol for all of the other devices. To specify a special symbol, use the VALUE= option to specify a name or a character from Table 25.65 on page 492:

```
symbol1 value=hash color=green;
symbol2 value=) color=blue;
```

This example uses color to ensure that each SYMBOL statement generates only one definition. You can omit color specifications to let SAS/GRAPH rotate symbol definitions through the color list. For details, see “Using Generated Symbol Sequences” on page 503.

To use plot symbols other than those in Table 25.65 on page 492, use the FONT= option to specify a font for the plot symbol. If the font is a symbol font, such as

Marker, the string specified with the VALUE= option is the character code for the symbol to be displayed. If the font is a text font, the string specified with the VALUE= option is displayed as the plot symbol. (See " VALUE=*special-symbol* | *text-string* | SPECIAL | NONE" on page 491 and " FONT="*font*"" on page 475.)

This table illustrates some of the ways that you can define a plot symbol:

Table 25.9 *Examples of Plot Symbol Definitions*

Definition	Plot Symbol
`symbol1 value=plus;`	+
`symbol2 value=+;`	⊕
`symbol3 font=swiss value=plus;`	plus
`symbol4 font=marker value=U;`	■
`symbol5 value="'";`	⚜

Specifying a Default Interpolation Method

The INTERPOL= option in a GOPTIONS statement specifies a default interpolation method to be used with all SYMBOL definitions. This default interpolation method is in effect unless you specify a different interpolation in a SYMBOL statement. If the GOPTIONS statement does not specify an interpolation method, the default for each SYMBOL statement is NONE.

Sorting Data with Spline Interpolation

To use the GPLOT procedure to sort by the horizontal axis variable before plotting, add the letter S to the end of any of the spline interpolation methods (INTERPOL=L, INTERPOL=SM, and INTERPOL=SPLINE). For example, suppose you want to overlay three plots (Y1*X1, Y2*X2, and Y3*X3). For each plot, you want the X variable sorted in ascending order. Use these statements:

```
symbol1 i=splines c=red;
symbol2 i=splines c=blue;
symbol3 i=splines c=green;
proc gplot;
   plot y1*x1 y2*x2 y3*x3 / overlay;
run;
```

Using Color

How to Specify Color for Symbols

Generally, there are two ways to explicitly specify color with SYMBOL statements:

- specify color in the SYMBOL statement
- specify color on the CSYMBOL= graphics option

You can also let SAS/GRAPH rotate symbol definitions through the color list. For details, see "Using Generated Symbol Sequences" on page 503.

Specifying Colors with SYMBOL Statements

The SYMBOL statement has these options for specifying color:

- The CV= option specifies color for plot symbols in GPLOT and GBARLINE, or for contour labels in GCONTOUR.
- The CO= option specifies color for confidence limit lines and area outlines in GPLOT and GBARLINE.
- The CI= option specifies color for plot lines in GPLOT and GBARLINE, or contour lines in GCONTOUR.
- The COLOR= option specifies color for the entire symbol. For GPLOT and GBARLINE, this includes plot symbols, plot lines, and outlines. For GCONTOUR, this includes contour lines and labels.

The CV= and CI= options have the same effect as using the COLOR= option when they are used in these ways:

- Only CV= or CI= option is used. (The option that is not used is assigned the value of the option used.)
- Both the CV= and CI= options specify the same color.

In general, the CI=, CV=, and CO= options color specific areas of the symbol. Use these options to produce symbols and plot lines of different colors without having to overlay multiple plot pairs. For example, if you request regression analysis with confidence limits, use the following statement. It assigns red to the plot symbol, blue to the regression lines, and green to the confidence limit lines:

```
symbol cv=red ci=blue co=green;
```

The COLOR= option colors the entire symbol or those portions of it not colored by one of the other color options. If the COLOR= option precedes the CI= or CV= options, the CI= or CV= specification is used instead. If none of the SYMBOL color options is used, color specifications are searched for in this order:

1. the CSYMBOL= option in a GOPTIONS statement
2. each color in the color list sequentially before the next SYMBOL definition is used

CAUTION

If no color options are used, the SYMBOL definition cycles through each color in the color list.

If the SYMBOL color options and the CSYMBOL= graphics option are not used, the SYMBOL definition cycles through each color in the color list before the next definition is used. For details, see "Using Generated Symbol Sequences" on page 503.

Specifying Color with CSYMBOL=

The CSYMBOL= option in the GOPTIONS statement specifies the default color to be used by all SYMBOL definitions:

```
goptions csymbol=green;
symbol1 value=star;
symbol2 value=square;
symbol3 value=special;
```

In this example, all three of the SYMBOL statements use green.

Note: Although the SYMBOL3 definition specifies VALUE=SPECIAL, because the CSYMBOL= graphics option is used, it generates only the first symbol in the 12–symbol list (DOT) using the color green.

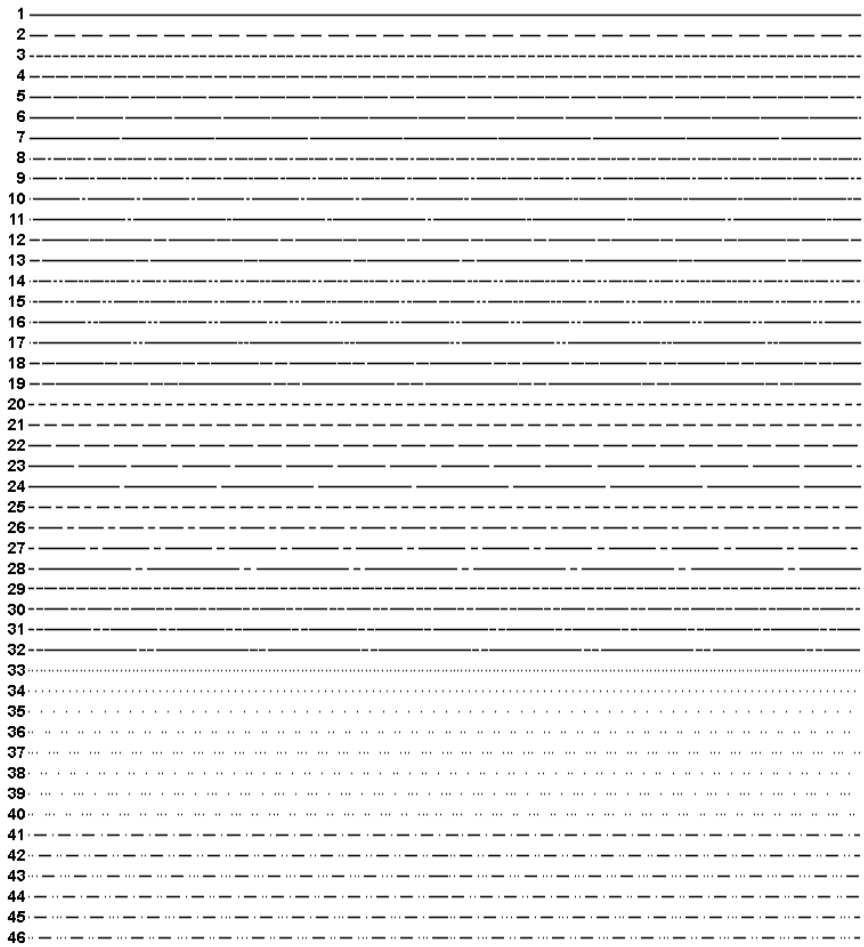
CSYMBOL= is overridden by any of the SYMBOL statement color options. See "Using Color" on page 500 for details.

If more SYMBOL definitions are needed, SAS/GRAPH returns to generating default symbol sequences.

Specifying Line Types

To specify the type of line for plot or contour lines, use the LINE= option to specify a number from 1 through 46. Figure 25.52 on page 503 shows the line types represented by these numbers. By default, the line type is 1 for plot and contour lines, and 2 for confidence limit lines.

Figure 25.21 *Line Types*



Note: These line types are also used by other statements and procedures. Some options accept a line type of 0, which produces no line.

Using Generated Symbol Sequences

How SYMBOL Definitions Are Generated

Symbol sequences are sets of SYMBOL definitions that are automatically generated by SAS/GRAPH software if any of these conditions is true:

- no valid SYMBOL definition is available. In this case, default symbol sequences are generated by rotating symbol definitions through the color specified in the GOPTIONS statement's CSYMBOL= option. If a CSYMBOL= color is not in effect, the definitions are rotated through the color list.
- a SYMBOL statement specifies color but not a plot symbol for the GPLOT procedure. Or a SYMBOL statement specifies a line type for the GCONTOUR procedure (assuming that GCONTOUR does not specify the needed line types).Either a default plot symbol or a line type is used with the specified color and only one definition is generated.

- a SYMBOL statement specifies a plot symbol for GPLOT or a line type for GCONTOUR, but no color options. In this case, the specified plot symbol or line type is used once with the color specified by the CSYMBOL= graphics option. If a CSYMBOL= color is not in effect, the specified plot symbol or line type is rotated through the color list.
- a SYMBOL statement specifies VALUE=SPECIAL for GPLOT, but no color options. In this case, the first symbol in the 12–symbol list (DOT) is used once with the color specified by the CSYMBOL= graphics option. If CSYMBOL= color is not in effect, the SAS/GRAPH software rotates through the 12 symbols and assigns each symbol a color from the color list.

If the REPEAT= option is also used, the resulting SYMBOL definition is repeated the specified number of times.

Default Symbol Sequences

Default symbol sequences are generated by rotating symbol definitions through the current color list.

- Definitions used for GPLOT rotate plot symbols through the color list; the first default plot symbol is a plus sign (+).
- Definitions used for GCONTOUR rotate line types; the first default line type is a solid line (line type 1).

Each time a default definition is required, SAS/GRAPH takes the first default plot symbol or line type and uses it with the first color in the color list. If more than one definition is required, it uses the same plot symbol or line type with the next color in the color list. It continues until all the colors have been used once. If more definitions are needed, SAS/GRAPH selects the second default plot symbol or line type and rotates it through the color list. It continues in this fashion, selecting default plot symbols or line types and cycling them through the color list until all the required definitions are generated.

A color can be specified with the CSYMBOL= option in the GOPTIONS statement. In this case, each default plot symbol or line type is used once with the specified color, and the colors in the color list are ignored.

Symbol Sequences Generated from SYMBOL Statements

You can use SYMBOL statements to generate a custom sequence of symbols and symbol colors for a plot. This is true if you do not specify the symbol color in the SYMBOL statement or with the CSYMBOL= graphics option. For symbols, you can use the default symbols or you can use the VALUE= option in the SYMBOL statement to specify a symbol by name (VALUE=*symbol-name*. Or instead use VALUE=*text-string*). Another possibility is to specify a predefined list of 12 special symbols (VALUE=SPECIAL). The predefined list of 12 symbols includes DOT, PLUS, X, TRIANGLE, SQUARE, STAR, DIAMOND, CIRCLE, HASH, Y, Z, and PAW. For the symbol colors, you can use the default color list or you can use the COLORS= graphics option to specify a custom list of colors.

Sometimes the symbol color is not specified and the VALUE=*symbol-name* or VALUE=*text-string* is used. In order to generate symbols for a plot in this case, SAS/GRAPH rotates each symbol through every color in the color list before it proceeds to the next SYMBOL definition. Sometimes the VALUE=SPECIAL option

is used to generate symbols for a plot. In this case SAS/GRAPH rotates through each symbol in the 12–symbol list and assigns each symbol a color from the color list. Before proceeding to the next SYMBOL definition, the SAS/GRAPH software continues rotating in this manner. It rotates until either all of the 12 symbols are used or all of the colors in the color list are used.

Here is an example of SYMBOL statements that use a custom color list with both default symbols and symbols that is identified by name:

```
goptions colors=(blue red green);
symbol1 cv=red i=join;
symbol2 i=spline v=dot;
symbol3 cv=green v=star;
```

In this example, the SYMBOL1 statement generates the first SYMBOL definition. The SYMBOL2 statement does not include color, so the first default plot symbol is rotated through all colors in the color list before the SYMBOL3 statement is used. This table shows the colors and symbols that would be used if nine symbol definitions were required for PROC GPLOT:

Table 25.10 *Example Output Using 9 Symbol Definitions in PROC GPLOT*

Sequence Number	Source	Characteristics : Color	Symbol	Interpolation
1	SYMBOL1	cv=red	First default	Join
2	SYMBOL2	color=blue	Dot	Spline
3	SYMBOL2	color=red	Dot	Spline
4	SYMBOL2	color=green	Dot	Spline
5	SYMBOL3	cv=green	Star	NONE
6	First default	color=blue	First default	Default
7	First default	color=red	First default	Default
8	First default	color=green	First default	Default
9	Second default	color=blue	Second default	Default

Notice that after the SYMBOL statements are exhausted, the procedure begins using the default definitions (sequences 6 through 9). Each plot symbol from the default list is rotated through all colors in the color list before the next plot symbol is used. Also, SYMBOL1 does not specify a plot symbol, so the default sequencing provides the first default symbol (a + sign). When sequencing resumes in sequence number 6, it starts at the beginning again, selecting the first default plot symbol and rotating it through the color list.

If you use the REPEAT= option but no color, the sequence generated by cycling the definition through the color list is repeated the number of times specified by the

REPEAT= option. Here is an example of statements that define a color list and illustrate the effect of the REPEAT= option on SYMBOL statements both with and without explicit color specifications:

```
goptions colors=(blue red green);
symbol1 color=gold repeat=2;
symbol2 value=star color=cyan;
symbol3 value=square repeat=2;
```

In this example, SYMBOL1 is used twice, SYMBOL2 is used once, and SYMBOL3 rotates through the list of three colors and then repeats this cycle a second time:

Table 25.11 *Example Output Using 3 SYMBOL Statements*

Sequence Number	Source	Characteristics: Color	Symbol	Interpolation
1	SYMBOL1	Gold	First default	Default
2	SYMBOL1	Gold	First default	Default
3	SYMBOL2	Cyan	Star	Default
4	SYMBOL3	Blue	Square	Default
5	SYMBOL3	Red	Square	Default
6	SYMBOL3	Green	Square	Default
7	SYMBOL3	Blue	Square	Default
8	SYMBOL3	Red	Square	Default
9	SYMBOL3	Green	Square	Default

Here is an example that uses a custom list of colors with the VALUE=SPECIAL option in the SYMBOL statement:

```
goptions reset=all colors(red green blue cyan);
symbol1 v=special;
```

In this example, assuming that four symbols are needed for a plot, the SYMBOL1 statement iterates through the 12–symbol list and assigns each symbol a color from the color list. Because four colors are specified in the color list, the SYMBOL1 statement generates only four symbols. Here is a summary of the symbols that are generated.

Table 25.12 *Example Output of 4 Symbols Generated with 1 SYMBOL Statement*

Sequence Number	Source	Characteristics: Color	Symbol	Interpolation
1	SYMBOL1	Red	Dot	Default
2	SYMBOL1	Green	Plus	Default

Sequence Number	Source	Characteristics: Color	Symbol	Interpolation
3	SYMBOL1	Blue	X	Default
4	SYMBOL1	Cyan	Triangle	Default

If more than four symbols are needed, SAS/GRAPH proceeds to the next SYMBOL definition. If no more SYMBOL definitions are provided, SAS/GRAPH returns to generating default symbol sequences for the remaining symbols.

If you want to specify the same color for all of the 12 symbols, use the CSYMBOL= graphics option. Or use a color option in the SYMBOL statement to specify the symbol color. You would then use the REPEAT= option to repeat the SYMBOL statement. Here is an example that specifies the color blue for all 12 of the symbols:

```
symbol1  value=special color=blue repeat=12;
```

The SYMBOL statement in this example repeats 12 times to generate the 12 special symbols. The symbol color is specified in the SYMBOL statement. So each time that the SYMBOL statement is repeated, the next symbol in the 12–symbol list is generated and is assigned the color blue. When a subsequent PLOT statement is executed, the symbol sequence begins again with the first symbol in the 12–symbol list (DOT).

TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement

Control the content, appearance, and placement of text.

Valid in: Anywhere

Used by: GAREABAR, GBARLINE, GCHART, GCONTOUR, GFONT, GKPI, GPLOT, GRADAR, GREPLAY, GSLIDE, GTILE, G3D

Type: TITLE and FOOTNOTE are global statements. NOTE is a local statement.

Restriction: The cumulative text from the TITLE1-10 statements must not exceed 256 characters for output using the SVG device.

Syntax

TITLE<1 …10> <*text-argument(s)*>;

FOOTNOTE<1 …10> <*text-argument(s)*>;

NOTE<*text-arguments(s)*>;

Summary of Optional Arguments

Appearance options

COLOR=*color*
: specifies the color for the following text, box, or line.

FONT=*font*
: specifies the font for the subsequent text.

HEIGHT=*text-height<units>*
: specifies the height of text characters in number of units.

Baseline angling and character rotation options

ANGLE=*degrees*
: specifies the angle of the baseline of the entire text string with respect to the horizontal.

LANGLE=*degrees*
: specifies the angle of the entire *text-string*(s).

ROTATE=*degrees*
: specifies the angle at which each character of text is rotated.

Boxing, underlining, and line drawing options

BCOLOR=*background-color*

BLANK=YES
: protects the box and its contents from being overwritten by any subsequent graphics elements.

BOX=1 …4
: draws a box around one line of text.

BSPACE=*box-space<units>*
: specifies the amount of space between the boxed text and the box.

DRAW=(*x,y* …,*x-n,y-n*)*<units>*
: draws lines anywhere on the graphics output area.

UNDERLIN=0 …3
: underlines subsequent text.

Linking option

ALT=*"text-string"*
: specifies descriptive text for a URL to which a title or footnote links, or the title or footnote itself.

LINK=*"URL"*
: specifies a uniform resource locator (URL) to which a title or footnote links.

Placement and spacing options

JUSTIFY=LEFT | CENTER | RIGHT
: specifies the alignment of the text string.

LSPACE=*line-space <units>*
: specifies the amount of spacing above and below lines of text.

MOVE=(*x,y*) *<units>*
: positions a text string.

WRAP
: wraps the text to a second line if the text does not fit on one line.

Text provision option

text-string(s)
is an element of *text-argument(s)* and specifies one or more strings up to 512 characters. Any of the following *text-options* that are used to modify a *text-string* must precede *text-string*.

Optional Arguments

text-arguments(s) can be one or more of *"text-string"* and *text-options*. Text options must precede the *"text-string"* that they modify. *text-options* can be one or more of the following optional arguments, in any order:

ALT=*"text-string"*
specifies descriptive text for a URL to which a title or footnote links, or the title or footnote itself. The *"text-string"* that you use to describe a title or a footnote, or the URL specified by the LINK= option, can contain occurrences of the variables #BYVAL, #BYVAR, and #BYLINE, as described in " *text-string(s)*" on page 521.

Supports The ALT= option can be used in conjunction with the LINK= option.

Note The title or footnote can be displayed by using an ODS markup destination (such as HTML). Also, the corresponding ODS option NOGTITLE or NOGFOOTNOTE is specified. In this case, the title or footnote is rendered in the body of the HTML file rather than in the graphic itself. And the ALT= text is not associated with the title or footnote.

See "Controlling Where Titles and Footnotes Are Rendered" on page 115

" LINK=*"URL"*" on page 518

Example "Producing a Drilldown Choropleth Map" in *SAS/GRAPH and Base SAS: Mapping Reference*

ANGLE=*degrees*
specifies the angle of the baseline of the entire text string with respect to the horizontal. A positive *degrees* value angles the *baseline* counterclockwise; a negative value angles it clockwise. By default, ANGLE=0 (horizontal).

Angled titles or footnotes might require more vertical space. Consequently, there might be an increase the size of the title area or the footnote area, thereby reducing the vertical space in the procedure output area.

Using the BOX= option with angled text does not produce angled boxes; the box is sized to accommodate the angled note.

Using the ANGLE= option after one text string and before another can reset some options to their default values. See "Using Options That Can Reset Other Options" on page 525.

The ANGLE= option has the same effect on the text as LANGLE=, except when you specify an angle of 90 degrees or –90 degrees. In these angle specifications, the procedure output area is shrunk from the left or right to accommodate the angled title or footnote. The result depends on the statement in which you use the option:

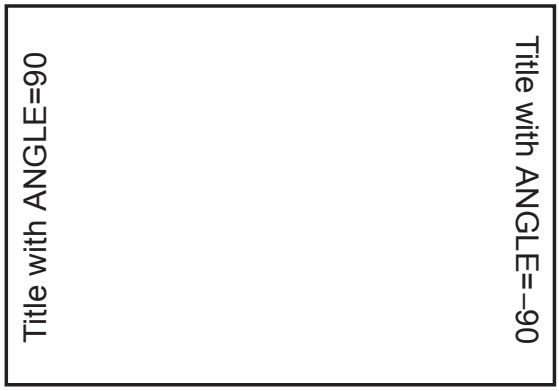
- With the TITLE statement:

  Figure 25.53 on page 510 shows how ANGLE=90-degrees or ANGLE=–90-degrees positions and rotates the title text.

ANGLE=90 positions the title at the left edge of the graphics output area, angled 90 -degrees (counterclockwise) and centered vertically.

ANGLE=–90 positions the title at the right edge of the graphics output area, angled –90-degrees (clockwise) and centered vertically.

Figure 25.22 *Positioning Titles with the ANGLE= Option*



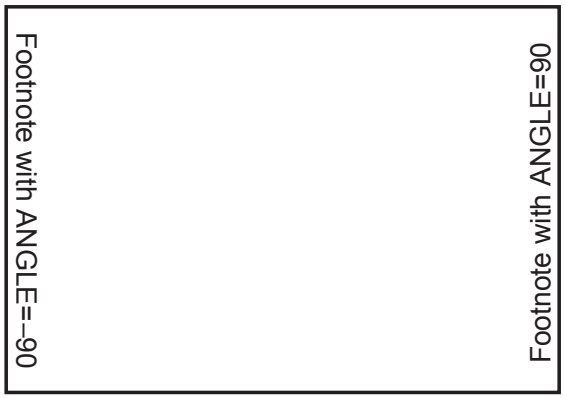
- With the FOOTNOTE statement:

  Figure 25.54 on page 510 shows how ANGLE=90 degrees or ANGLE=–90 degrees positions and rotates footnote text.

ANGLE=90 positions the footnote at the right edge of the graphics output area, angled 90 degrees (counterclockwise) and centered vertically.

ANGLE=–90 positions the footnote at the left edge of the graphics output area, angled –90 (clockwise) and centered vertically.

Figure 25.23 *Positioning Footnotes with the ANGLE=Option*



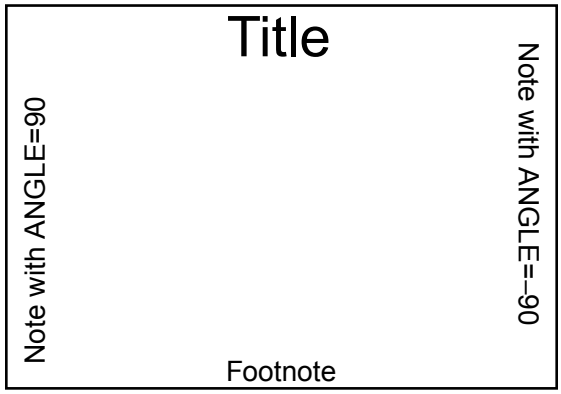
- With the NOTE statement:

  Figure 25.55 on page 511 shows how ANGLE= 90 degrees or -90 degrees positions and rotates note text.

ANGLE=90 positions the note at the bottom of the left edge of the graphics output area, angled 90 degrees (counterclockwise) and reading from bottom to top.

ANGLE=–90 positions the note at the top of the right edge of the graphics output area, angled –90 (clockwise) and reading from top to bottom.

Figure 25.24 *Positioning Notes with the ANGLE= Option*



Alias A=

Restriction Not supported by ActiveX

See " LANGLE=*degrees*" on page 517

" ROTATE=*degrees*" on page 520

Example "Example 6: Enhancing Titles" on page 547

BCOLOR=*background-color*

specifies the background color of a box produced by the BOX= option. By default, the background color of the box is the same as the background color for the entire graph. The color of the frame of the box is determined by the color specification used in BOX=.

Alias BC=

Restriction If you omit BOX=, BCOLOR= is ignored.

Note The BCOLOR= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.

See "Using Options That Can Reset Other Options" on page 525 for details

" BOX=1 …4 " on page 512

Example "Example 6: Enhancing Titles" on page 547

BLANK=YES

protects the box and its contents from being overwritten by any subsequent graphics elements. It does this by blanking out the area where the box is displayed. The BLANK= option enables you to overlay graphics elements with boxed text. It is ignored if you omit the BOX= option. Because titles and footnotes are written from the highest numbered to the lowest numbered, the BLANK= option only blanks out titles and footnotes of a lower number.

Alias BL=

Restriction Not supported by ActiveX

Note The BLANK= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.

See "Using Options That Can Reset Other Options" on page 525 for details

" BOX=1 …4 " on page 512

Example "Example 6: Enhancing Titles" on page 547

BOX=1 …4

draws a box around one line of text. A value of 1 produces the thinnest box lines; 4 produces the thickest. Boxing angled text does not produce an angled box; the box is sized to include the angled text.

The color of the box is one of the following:

- the color specified by the COLOR= option in the statement
- the default text color

The COLOR= option, preceding the BOX= option, controls the box frame color. To color the background of the box, use the BCOLOR= option. Specify the same color with both the COLOR= option and the BCOLOR= option to effect a box without a frame.

You can include more than one text string in the box as long as no text break occurs between the strings. That is, you cannot use the JUSTIFY= option to create multiple lines of text within a box.

To draw a box around multiple lines of text, you can do either of the following:

- Use the MOVE= option with relative coordinates to position an additional line of text relative to the preceding line and enclose them within the box drawn by the BOX= option. For example, this statement produces the boxed note shown in Figure 25.56 on page 513:

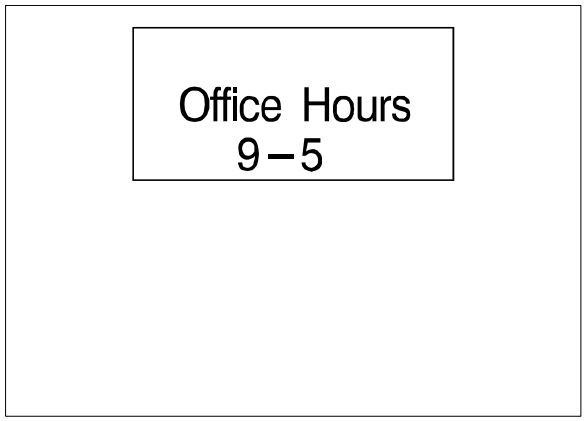
```
note font=swiss justify=center box=1
   "Office Hours"      move=(-7,-2)pct "9-5";
```

  The MOVE=option in this NOTE statement overrides the JUSTIFY= option that center-positioned the first line of text. The MOVE= option specifies the *x* and *y* coordinates in percentage units. The text string '9–5' is positioned relative to the first text string 'Office Hours'.

  Using the MOVE= option in a NOTE statement enables you to position a note on top of a graph. See the "NOTE Statement" for a code sample.

- Use the DRAW= option to draw the box and do not use the BOX= option.

Figure 25.25 *Using the BOX= Option and the MOVE= Option to Box Multiple Lines of Text*



Alias BO=

Restriction Not supported by ActiveX

Note The BOX= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= options with absolute coordinates.

See "Using Options That Can Reset Other Options" on page 525 for details

" BCOLOR=*background-color*" on page 511

" BLANK=YES " on page 511

" BSPACE=*box-space<units>*" on page 513

Example "Example 6: Enhancing Titles" on page 547

BSPACE=*box-space<units>*

specifies the amount of space between the boxed text and the box. The space above the text is measured from the font maximum, and the space below the text is measured from the font minimum. By default, BSPACE=1.

The spacing is uniform around the box. Notice that BSPACE=.5IN leaves one-half inch of space between the text and the top, bottom, and sides of the box.

Alias BS=

Restriction Not supported by ActiveX

Interaction If the BOX= option is not used, the BSPACE= option is ignored

Note The BSPACE= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.

See "Using Options That Can Reset Other Options" on page 525 for details

" BOX=1 …4 " on page 512

COLOR=*color*

specifies the color for the following text, box, or line. The COLOR= option affects all text, lines, and boxes that follow it and stays in effect until another COLOR= specification is encountered.

Change colors as often as you like. For example, this statement produces a title with red text in a box with a blue frame and a cream background:

```
title color=red "Total Sales" color=blue
  box=3 bcolor=cream;
```

Although the BCOLOR= option controls the background color of the box, the frame color is controlled with the COLOR= option that precedes the BOX= option.

If you omit the COLOR= option, a color specification is searched for in this order:

1. the CTITLE= option in a GOPTIONS statement
2. the CTEXT= option in a GOPTIONS statement
3. the default, the first color in the color list.

Alias C=

Style reference Color attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements

See " BCOLOR=*background-color*" on page 511

"Controlling Titles and Footnotes" on page 115

DRAW=(*x,y* …,*x-n,y-n*)<*units*>

draws lines anywhere on the graphics output area. Lines are drawn using *x* and *y* as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

Table 25.13 *Coordinate Specifications*

Absolute Coordinates	Relative Coordinates
x<units>	±*x<units>*
y<units>	±*y<units>*

The coordinate position (0,0) is the lower left corner of the graphics output area. Specify at least two coordinate pairs. Commas between coordinates are optional; blanks can be used instead. The DRAW= option does not affect the positioning of text.

The starting point for lines specified with relative coordinates begins at the end of the most recently drawn text or line in the current statement. If no text or line has been drawn in the current statement, a warning is issued. The relative draw is measured from where a zero-length text string would have ended, given the normal placement for the statement.

You can mix relative and absolute coordinates. For example, DRAW=(+0,+0,+0,1IN) draws a vertical line from the end of the text to one inch from the bottom of the graphics output area.

Alias D=

Restriction Not supported by ActiveX

Example "Example 6: Enhancing Titles" on page 547

FONT=*font*

specifies the font for the subsequent text. See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349 for details about specifying SAS/GRAPH fonts. If you omit this option, a font specification is searched for in this order:

- for a TITLE1 statement:

1. the FTITLE= option in a GOPTIONS statement
2. the FTEXT= option in a GOPTIONS statement
3. the default font, SWISS (COMPLEX in Release 6.06 and earlier)

- for all other TITLE statements and the FOOTNOTE and NOTE statements:

1. the FTEXT= option in a GOPTIONS statement
2. the default hardware font, NONE

Alias F=

Style reference Font attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements

Notes Font names greater than eight characters in length must be enclosed in quotation marks.

If the TITLE or FOOTNOTE is displayed using an ODS markup destination and the corresponding NOGTITLE or NOGFOOTNOTE option is specified, then the `bold` and `italic` FONT attributes are on by default. However, if you specify different attributes with the FONT= option, the `bold` and `italic` attributes are turned off.

See "Controlling Titles and Footnotes" on page 115

Example "Example 6: Enhancing Titles" on page 547

HEIGHT=*text-height<units>*

specifies the height of text characters in number of units.

Height is measured from the font minimum to the capline. Ascenders can extend above the capline, depending on the font.

If your text line is too long to be displayed in the height specified in the HEIGHT= option, the height specification is reduced so that the text can be displayed. A note in the SAS log tells you what percentage of the specified size was used.

If you omit the HEIGHT= option, a text height specification is searched for in this order:

- for a TITLE or TITLE1 statement:

1. the HTITLE= option in a GOPTIONS statement
2. the HTEXT= option in a GOPTIONS statement
3. the default value specified by the current ODS style.

- for all other TITLE statements and the FOOTNOTE and NOTE statements:

1. the HTEXT= option in a GOPTIONS statement
2. the default value, 1

Alias	H=
Default	For TITLE or TITLE1 statements, HEIGHT is determined by the current ODS style. HEIGHT=1 for subsequent TITLE statements.
Style reference	Height attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements
Restriction	Partially supported by ActiveX
Notes	The ActiveX control enables you to control the relative height of text with the HEIGHT= option, but not the absolute height in terms of specific units. A TITLE or FOOTNOTE statement without a number is treated as a TITLE1 or FOOTNOTE1 statement.
See	“Controlling Titles and Footnotes” on page 115
Examples	“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 527 “Example 6: Enhancing Titles” on page 547

JUSTIFY=LEFT | CENTER | RIGHT

specifies the alignment of the text string. The default depends on the statement with which you use the JUSTIFY= option:

- for a FOOTNOTE statement the default is CENTER
- for a NOTE statement the default is LEFT
- for a TITLE statement the default is CENTER.

All the text strings following JUSTIFY= are treated as a single string and are displayed as one line that is left-, right-, or center-aligned.

You can change the justification within a single line of text. For example, this NOTE statement displays a date on the left side of the output and the page number on the same line on the right:

```
note "June 28, 1997" justify=right "Page 3";
```

In addition, you can use the JUSTIFY= option to produce multiple lines of text by repeating the JUSTIFY= option with the same value before the text string for each line. Multiple lines of text with the same justification are blocked together. For example, this TITLE statement produces a three-line title with each line right-justified:

```
title justify=right "First Line"
      justify=right "Second Line"
      justify=right "Third Line";
```

You can get the same effect with three TITLE statements, each specifying JUSTIFY=RIGHT. You can produce a block of text by specifying the same justification for multiple text strings. You can then change the justification for an

additional text string. The result is that the text is placed on the same line as the first string specified in the statement.

Alias J=L | C | R

Note Using the JUSTIFY= option after one text string and before another can reset some options to their default values.

See "Using Options That Can Reset Other Options" on page 525 for details

Example "Example 3: Rotating Plot Symbols through the Color List" on page 535

LANGLE=*degrees*

specifies the angle of the *baseline* of the entire text string(s) with respect to the horizontal A positive value for *degrees* moves the *baseline* counterclockwise; a negative value moves it clockwise. By default, LANGLE=0 (horizontal).

Angled titles or footnotes might require more vertical space. Consequently, there might be an increase the size of the title area or the footnote area, thereby reducing the vertical space in the procedure output area.

Using the BOX= option with angled text does not produce an angled box; the box is sized to accommodate the angled note.

Unlike the ANGLE= option, the LANGLE= option does not reset any other options. Therefore, the LANGLE= option is easier to use because you do not need to repeat options after a text break.

The LANGLE= option has the same effect on the text as the ANGLE= option, except when an angle of 90 degrees or –90 degrees is specified. The result depends on the statement in which you use the option:

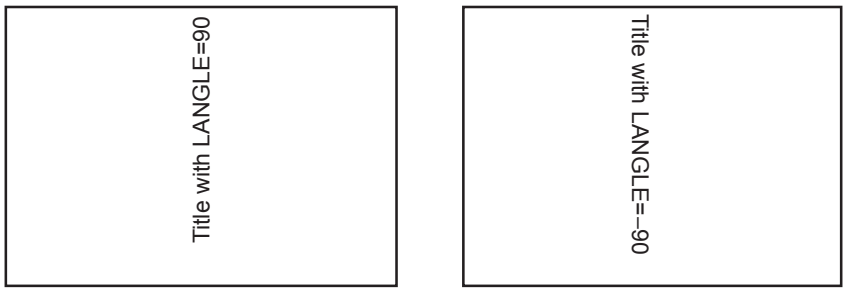
- With the TITLE statement:

  Figure 25.57 on page 517 shows how LANGLE=90 degrees and LANGLE=–90 degrees positions and rotates titles.

LANGLE=90 angles the title 90 degrees (counterclockwise) so that it reads from bottom to top. The title is centered horizontally and positioned at the top of the picture.

LANGLE=-90 angles the title –90 degrees (clockwise) so that it reads from top to bottom. The title is centered horizontally and positioned at the top of the picture.

Figure 25.26 *Positioning Titles with the LANGLE= Option*



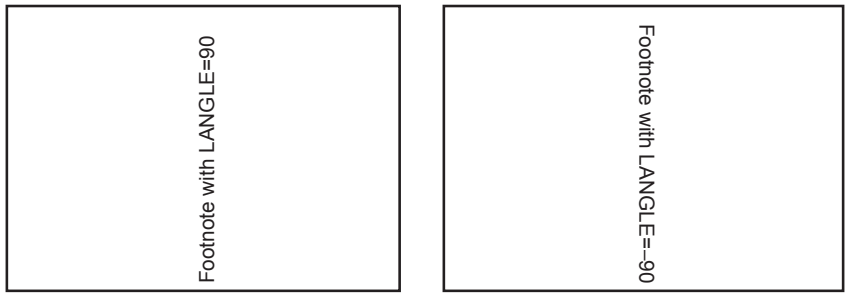
- With the FOOTNOTE statement:

Figure 25.58 on page 518 shows how LANGLE=90 degrees and LANGLE=–90 degrees positions and rotates footnotes.

LANGLE=90 angles the footnote 90 degrees (counterclockwise) so that it reads from bottom to top. The footnote is centered horizontally and positioned as the bottom of the picture.

LANGLE=–90 angles the footnote –90 degrees (clockwise) so that it reads from top to bottom. The footnote is centered horizontally and positioned at the bottom of the picture.

Figure 25.27 *Positioning Footnotes with the LANGLE= Option*



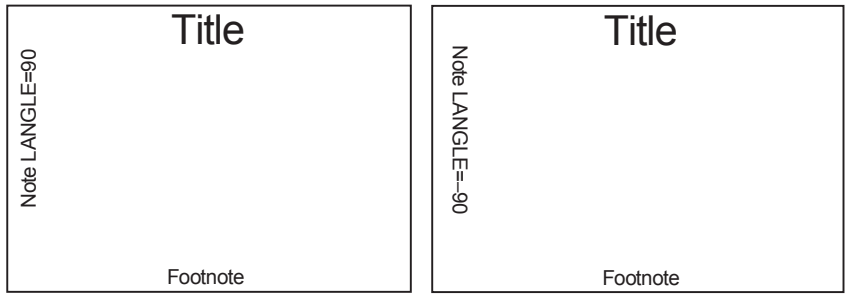
- With the NOTE statement:

  Figure 25.59 on page 518 shows how LANGLE=90 degrees and LANGLE=–90 degrees positions and rotates notes.

LANGLE=90 positions the note at the top of the left edge of the procedure output area, angled 90 degrees (counterclockwise) so that it reads from bottom to top.

LANGLE=–90 positions the note at the top of the left edge of the procedure output area, angled –90 degrees (clockwise) so that it reads from top to bottom.

Figure 25.28 *Positioning Notes with the LANGLE= Option*



Alias LA=

Restriction Not supported by ActiveX

See " ANGLE=*degrees*" on page 509

LINK=*"URL"*

specifies a uniform resource locator (URL) to which a title or footnote links. The *text-string* that you use to specify the URL can contain occurrences of the variables #BYVAL, #BYVAR, and #BYLINE, as described in " *text-string(s)*" on page 521.

Supports The LINK= option can be used in conjunction with the ALT= option.

Note The title or footnote can display using an ODS markup destination (such as HTML). And the corresponding ODS option NOGTITLE or NOGFOOTNOTE can be specified. Then the title or footnote is rendered in the body of the HTML file rather than in the graphic itself. Specifying the NOGTITLE or NOGFOOTNOTE options results in increasing the amount of space allowed for the procedure output area, which can result in increasing the size of the graph. Space that would have been used for the title or footnote is devoted instead to the graph. You might need to be aware of this possible difference if you are using annotate or map coordinates.

See "Controlling Where Titles and Footnotes Are Rendered" on page 115

" ALT=*"text-string"*" on page 509

Example "Producing a Drilldown Choropleth Map" in *SAS/GRAPH and Base SAS: Mapping Reference*

LSPACE=*line-space* <*units*>

specifies the amount of spacing above lines of note and title text and the amount of spacing below lines of footnote text. For notes and titles, the spacing is measured from the capline of the current line to the font minimum of the line above. For footnotes, the spacing is measured from the font minimum of the current line to the capline of the line below.

Alias LS=

Default LSPACE=0

Restriction Not supported by ActiveX

Note The LSPACE= option can be reset by the ANGLE= or JUSTIFY= option, or by the MOVE= option with absolute coordinates.

See "Using Options That Can Reset Other Options" on page 525 for details

MOVE=(*x*,*y*) <*units*>

positions an initial or subsequent text string, or lines of text anywhere on the graphics output area. The positioning uses *x* and *y* as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

Table 25.14 *Coordinate Specifications*

Absolute Coordinates	Relative Coordinates
x<units>	±*x<units>*
y<units>	±*y<units>*

Commas between coordinates are optional; you can use blanks instead.

The starting point for lines specified with absolute coordinates is at the bottom left corner of the graphics output area. For example, the following code fragment

uses the NOTE statement to add a simple line of text to the upper right quadrant of the graphics output area:

```
note move=(65,70)pct color=purple "My Note";
```

The starting point for lines specified with relative coordinates is at the end of the most recently drawn text or line in the current statement. If no text or line has been drawn in the current statement, a warning is issued in the SAS log. The relative move is measured from where a zero-length text string would have ended, given the normal placement for the statement.

You can mix relative and absolute coordinates.

Alias M=

Restriction Not supported by ActiveX

Interactions The MOVE= option overrides a JUSTIFY= option specified for the same text string

A NOTE, FOOTNOTE, or TITLE statement can use the MOVE= option to position the text so that the statement does not use its default position. In this case, the text of the next NOTE, FOOTNOTE, or TITLE statement occupies the unused position and no blank lines are displayed

Notes You can specify the MOVE= option with at least one absolute coordinate. If the option follows one text string and precedes another, some options can be reset to their default values.

If you specify the GUNIT graphics option, then that unit is the default unit. If you do not specify the GUNIT= graphics option, then the default unit is CELLS.

See “Using Options That Can Reset Other Options” on page 525 for details

“ BOX=1 …4 ” on page 512 for illustrative code that draws a box around lines of text positioned with relative coordinates

Examples “Example 2: Specifying Logarithmic Axes” on page 531

“Example 6: Enhancing Titles” on page 547

ROTATE=*degrees*

specifies the angle at which each character of text is rotated with respect to the *baseline* of the text string. The angle is measured from the current text *baseline* angle, which is specified by the ANGLE= or LANGLE= options. By default, the *baseline* is horizontal. A positive value for *degrees* rotates the character counterclockwise; a negative value rotates it clockwise. By default, ROTATE=0 (parallel to the *baseline*).

Figure 25.60 on page 520 shows how characters are positioned when ROTATE=90 is used with the default (horizontal) *baseline*.

Figure 25.29 *Tilting Characters with the ROTATE= Option*

Alias R=

Restriction Not supported by ActiveX

See " ANGLE=*degrees*" on page 509

Example "Example 6: Enhancing Titles" on page 547

text-string(s)

is an element of *text-argument(s)* and specifies one or more strings up to 512 characters. Any of the following *text-options* that are used to modify a *text-string* must precede *text-string*.

You must enclose text strings in single or double quotation marks. The text appears exactly as you enter it in the statement, including uppercase and lowercase characters and blanks.

To use single quotation marks or apostrophes within the title, you can do either of the following:

- use a pair of single quotation marks together:

```
footnote 'All''s Well That Ends Well';
```

- enclose the text in double quotation marks:

```
footnote "All's Well That Ends Well";
```

You can specify multiple text strings in FOOTNOTE and TITLE statements. If you want each string to be separated by a blank, add a blank to the beginning of the second and subsequent text strings rather than at the end the preceding string, as in this example:

```
note color=red "Sales:" color=blue " 2000";
```

With fonts that support Unicode, you can specify special characters by specifying a hexadecimal constant in quotation marks. A trailing `x` identifies a string as a hexadecimal constant. You must designate the hexadecimal constant as a Unicode specification and escape it: `(*ESC*){Unicode '`*`hexadecimal-value`*`'x}`. For example:

```
title "Regression with Confidence Limits ((*ESC*){unicode '03B1'x}=.05)";
```

This statement produces the title, "Regression with Confidence Limits (α = .05)" because '03B1'x is the hexadecimal value for the lowercase Greek letter alpha in all Unicode fonts.

> **TIP** You can instead specify lowercase Greek letters by name, and uppercase Greek letters by name_u: `(*ESC*){Unicode alpha}=.05` and `(*ESC*){Unicode alpha_u}=.05`.

For more information see "Specifying Special Characters Using Character and Hexadecimal Codes" on page 355.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYLINE

substitutes the entire BY line without leading or trailing blanks for #BYLINE in the text string. It also displays the BY line in the footnote, note, or title produced by the statement.

#BYVAL*n*
#BYVAL(*BY-variable-name*)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

#BYVAR*n*
#BYVAR(*BY-variable-name*)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 526

Restriction #BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

UNDERLIN=0 …3
underlines subsequent text. Values of 1, 2 and 3 underline with an increasingly thicker line. UNDERLIN=0 halts underlining for subsequent text.

Underlines follow the text *baseline*. If you use an LANGLE= or ANGLE= option for the line of text, the underline is drawn at the same angle as the text. Underlines do not break up to follow rotated characters.

To make the text and the underline the same color, specify a COLOR= option before the UNDERLIN= option that precedes the text string. To make the text a different color, specify the COLOR= option after the UNDERLIN= option.

Aliases U=

UNDERLINE=

Restriction Partially supported by ActiveX

Notes The UNDERLIN= option can be reset by the ANGLE= or JUSTIFY= option, or by the MOVE= option with absolute coordinates. See "Using Options That Can Reset Other Options" on page 525 for details.

The ActiveX control underlines text when the UNDERLIN= option is specified, but it does not vary the thickness of the line.

See " ROTATE=*degrees*" on page 520

Example "Example 6: Enhancing Titles" on page 547

WRAP

wraps the text to a second line if the text does not fit on one line. If the WRAP option is omitted, the text font-size is reduced until the text fits on one line. Wrapping occurs at the last blank before the text meets the end of the window. If there are no blanks in the text string, then there is no wrapping.

Restriction The WRAP option does not work with the BOX, BLANK, UNDERLINE, and MOVE options.

Details

Using TITLE and FOOTNOTE Statements

You can define TITLE and FOOTNOTE statements anywhere in your SAS program. They are global and remain in effect until you cancel them or until you end your SAS session. All currently defined FOOTNOTE and TITLE statements are automatically displayed.

You can define up to ten TITLE statements and ten FOOTNOTE statements in your SAS session. A TITLE or FOOTNOTE statement without a number is treated as a TITLE1 or FOOTNOTE1 statement. You do not have to start with TITLE1 and you do not have to use sequential statement numbers. Skipping a number in the sequence leaves a blank line.

You can use as many text strings and options as you want, but place the options before the text strings that they modify. See "Using Multiple Options" on page 524.

The most recently specified TITLE or FOOTNOTE statement of any number completely replaces any other TITLE or FOOTNOTE statement of that number. In addition, it cancels all TITLE or FOOTNOTE statements of a higher number. For example, if you define TITLE1, TITLE2, and TITLE3, then resubmitting the TITLE2 statement cancels TITLE3.

To cancel individual TITLE or FOOTNOTE statements, define a TITLE or FOOTNOTE statement of the same number without options (a null statement):

```
title4;
```

But remember that this cancels all other existing statements of a higher number.

To cancel all current TITLE or FOOTNOTE statements, use the RESET= graphics option in a GOPTIONS statement:

```
goptions reset=footnote;
```

Specifying RESET=GLOBAL or RESET=ALL also cancels all current TITLE and FOOTNOTE statements as well as other settings.

Note: The cumulative text from the TITLE1-10 statements must not exceed 256 characters for output using the SVG device.

Using the NOTE Statement

NOTE statements are local, not global, and they must be defined within a procedure or RUN-group with which they are used. They remain in effect for the duration of the procedure that includes NOTE statements in any of its RUN-groups or until you end your SAS session. All notes defined in the current RUN group, as well as those defined in previous RUN-groups, are displayed in the output as long as the procedure remains active.

You can use as many text strings and options as you want, but place the options before the text strings that they modify. See "Using Multiple Options" on page 524.

Using Multiple Options

In each statement, you can use as many text strings and options as you want, but you must place the options before the text strings that they modify. Most options affect all text strings that follow them in the same statement, unless the option is explicitly reset to another value. In general, TITLE, FOOTNOTE, and NOTE statement options stay in effect until one of these events occurs:

- The end of the statement is reached.
- A new specification is made for that option.

For example, this statement specifies that one part of the note is red and another part is blue, but the height for all of the text is 4:

```
note height=4 color=red "Red Tide"
     color=blue " Effects on Coastal Fishing";
```

Controlling Placement of Text

The following options enable the positioning of text on the graphics output area:

- " JUSTIFY=LEFT | CENTER | RIGHT" on page 516 aligns the text string
- " MOVE=(*x*,*y*) <*units*>" on page 519 positions a line or lines of text anywhere on the graphics output area using *x* and *y* as absolute or relative coordinates.

  Figure 25.56 on page 513 illustrates using the MOVE= option with relative coordinates to position a subsequent line of text
- "WRAP" on page 523 wraps the text to a second line

See the description of each option for usage details and restrictions.

Setting Defaults

You can set default characteristics for titles (including TITLE1 definitions), footnotes, and notes by using the following graphics options in a GOPTIONS statement:

CTITLE=*color*
: sets the default color for all titles, footnotes, and notes; overridden by the COLOR= option in a TITLE, FOOTNOTE, or NOTE statement.

CTEXT=*text-color*
: sets the default color for all text; overridden by the CTITLE= option for titles, footnotes, and notes.

FTITLE=*title-font*
: sets the default font for TITLE1 definitions; overridden by the FONT= option in the TITLE1 statement.

FTEXT=*text-font*
: sets the default font for all text, including the TITLE1 statement if the FTITLE= option is not used; overridden by the FONT= option a TITLE, FOOTNOTE, or NOTE statement.

HTITLE=*height<units>*
: sets the default height for TITLE1 definitions; overridden by the HEIGHT= option in the TITLE1 statement.

HTEXT=*n<units>*
: sets the default height for all text, including the TITLE1 statement if the HTITLE= option is not used; overridden by the HEIGHT= option a TITLE, FOOTNOTE, or NOTE statement.

See “Graphics Options and Device Parameters Dictionary” on page 577 for a complete description of each option.

Using Options That Can Reset Other Options

The ANGLE=, MOVE=, and JUSTIFY= options affect the position of the text and cause text breaks. (To cause a text break, the MOVE= option must have at least one absolute coordinate.) When a statement contains multiple text strings, the resulting text break can cause the following options to reset to their default values:

- BCOLOR=
- BLANK=
- BOX=
- BSPACE=
- LSPACE=
- UNDERLIN=.

Note: The LANGLE= option does not cause a text break.

In a TITLE, FOOTNOTE, or NOTE statement, before the first text string, you can use an option that can be reset (such as the UNDERLIN= option). Before the second string, if you use an option that resets it (such as the JUSTIFY= option), the first option does not affect the second string. In order for the first option to affect the second string, repeat the option and position it after the resetting option and before the text string.

For example, this statement produces a two-line title in which only the first line is underlined:

```
title underlin=2 "Line 1" justify=left "Line 2";
```

To underline Line 2, repeat the UNDERLIN= option before the second text string and after the JUSTIFY= option:

```
title underlin=2 "Line 1" justify=left
      underlin=2 "Line 2";
```

Substituting BY Line Values in a Text String

The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

- The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.
- There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the “BY Statement” on page 424.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

Examples

Example 1: Ordering Axis Tick Marks with SAS Date Values

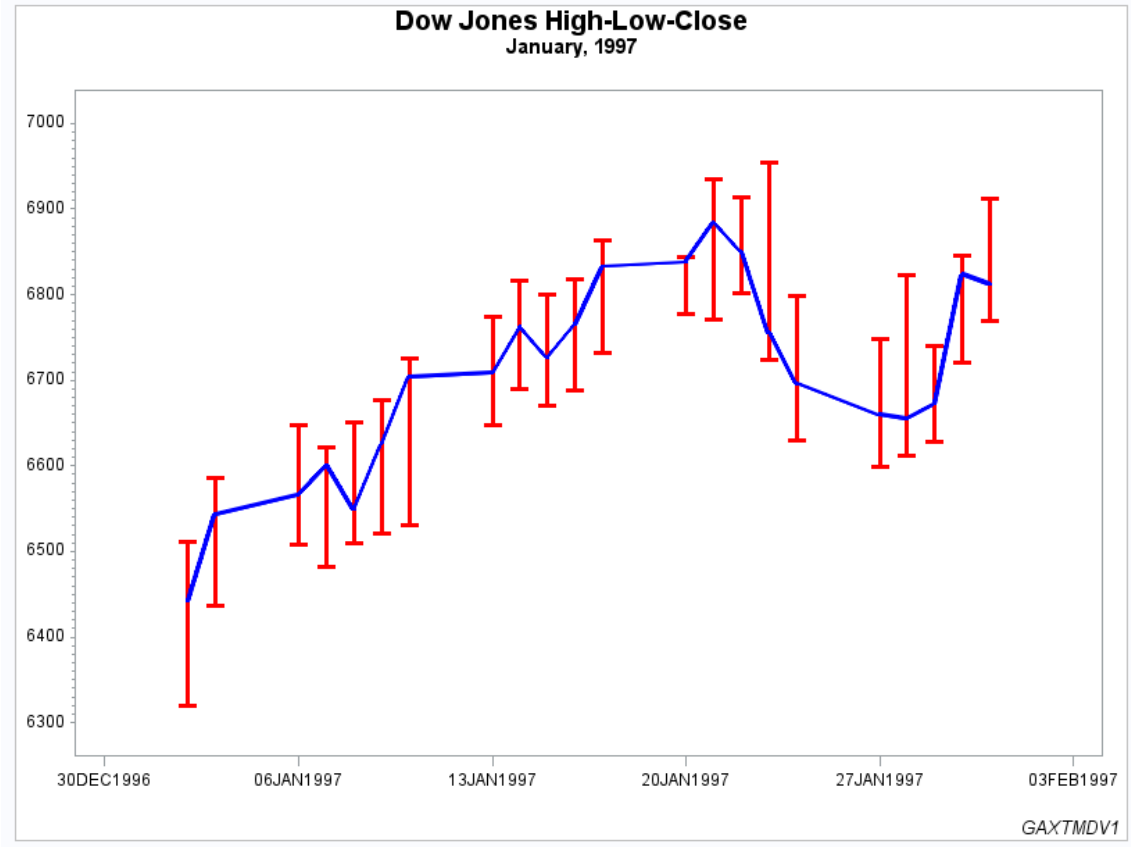
Features:	AXIS statement options LABEL= OFFSET= ORDER= FOOTNOTE statement option JUSTIFY= SYMBOL statement options INTERPOL= WIDTH= GOPTIONS statement options BORDER
Sample library member:	GAXTMDV1

Details

The code for this example is in SAS Sample Library member GAXTMDV1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses SAS datetime values with an AXIS statement's ORDER= option to set the major tick marks on the horizontal axis. It adjusts the position of the first and last major tick marks. The example also uses HILOCTJ interpolation in a SYMBOL statement to join minimum and maximum values.

Output



Program

```
goptions reset=all border;

data dowhlc;
     input date date9. high low close;
     format date date9.;
     datalines;
02JAN1997   6511.38  6318.96  6442.49
03JAN1997   6586.42  6437.10  6544.09
06JAN1997   6647.22  6508.30  6567.18
07JAN1997   6621.82  6481.75  6600.66
08JAN1997   6650.30  6509.84  6549.48
09JAN1997   6677.24  6520.23  6625.67
10JAN1997   6725.35  6530.62  6703.79
13JAN1997   6773.45  6647.99  6709.18
14JAN1997   6816.17  6689.94  6762.29
15JAN1997   6800.77  6669.93  6726.88
16JAN1997   6818.47  6688.40  6765.37
17JAN1997   6863.88  6732.66  6833.10
20JAN1997   6839.13  6777.30  6843.87
21JAN1997   6934.69  6771.14  6883.90
22JAN1997   6913.14  6801.16  6850.03
```

```
23JAN1997   6953.55  6724.19  6755.75
24JAN1997   6798.08  6629.91  6696.48
27JAN1997   6748.82  6598.73  6660.69
28JAN1997   6823.48  6612.20  6656.08
29JAN1997   6673.39  6627.98  6740.74
30JAN1997   6845.03  6719.96  6823.86
31JAN1997   6912.37  6769.99  6813.09
;

data dowhlc2;
   set dowhlc;
   drop high low close;
   dow=high; output;
   dow=low; output;
   dow=close; output;
run;

title1 "Dow Jones High-Low-Close";
title2 "January, 1997";
footnote justify=right "GAXTMDV1 ";

symbol interpol=hiloctj
       cv=red
       ci=blue
       width=2;

axis1 order=("30DEC1996"d to "03FEB1997"d by week)
      offset=(3,3)
      label=none ;

axis2
      label=none
      offset=(2,2);

proc gplot data=dowhlc2;
   plot dow*date / haxis=axis1
                   vaxis=axis2;
run;
quit;
```

Program Description

Set the graphics environment. BORDER draws a border around the graph.

```
goptions reset=all border;
```

Create the data set. DOWHLC contains the high, low, and close values of the Dow Jones Industrial index for each business day for a month.

```
data dowhlc;
     input date date9. high low close;
     format date date9.;
     datalines;
02JAN1997   6511.38  6318.96  6442.49
03JAN1997   6586.42  6437.10  6544.09
06JAN1997   6647.22  6508.30  6567.18
07JAN1997   6621.82  6481.75  6600.66
08JAN1997   6650.30  6509.84  6549.48
09JAN1997   6677.24  6520.23  6625.67
10JAN1997   6725.35  6530.62  6703.79
```

```
13JAN1997   6773.45  6647.99  6709.18
14JAN1997   6816.17  6689.94  6762.29
15JAN1997   6800.77  6669.93  6726.88
16JAN1997   6818.47  6688.40  6765.37
17JAN1997   6863.88  6732.66  6833.10
20JAN1997   6839.13  6777.30  6843.87
21JAN1997   6934.69  6771.14  6883.90
22JAN1997   6913.14  6801.16  6850.03
23JAN1997   6953.55  6724.19  6755.75
24JAN1997   6798.08  6629.91  6696.48
27JAN1997   6748.82  6598.73  6660.69
28JAN1997   6823.48  6612.20  6656.08
29JAN1997   6673.39  6627.98  6740.74
30JAN1997   6845.03  6719.96  6823.86
31JAN1997   6912.37  6769.99  6813.09
;
```

Prepare the data for a high-low plot. DOWHLC2 generates three records for each date, storing each date's high, low, and close values in variable DOW.

```
data dowhlc2;
   set dowhlc;
   drop high low close;
   dow=high; output;
   dow=low; output;
   dow=close; output;
run;
```

Define titles and footnote. JUSTIFY=RIGHT in the FOOTNOTE statement causes the footnote to be displayed in the bottom right.

```
title1 "Dow Jones High-Low-Close";
title2 "January, 1997";
footnote justify=right "GAXTMDV1 ";
```

Define symbol characteristics. INTERPOL=HILOCTJ specifies that the minimum and maximum values of DOW are joined by a vertical line with a horizontal tick mark at each end. The close values are joined by straight lines. The CV= option controls the color of the symbol. The CI= and WIDTH= options control the color and the thickness of the line that joins the close points.

```
symbol interpol=hiloctj
       cv=red
       ci=blue
       width=2;
```

Define characteristics of the horizontal axis. The ORDER= option uses a SAS date value to set the major tick marks. The OFFSET= option moves the first and last tick marks to make room for the tick mark value.

```
axis1 order=("30DEC1996"d to "03FEB1997"d by week)
      offset=(3,3)
      label=none ;
```

Define characteristics of the vertical axis. LABEL=NONE suppresses the AXIS label.

```
axis2
      label=none
      offset=(2,2);
```

Generate the plot and assign AXIS definitions. The HAXIS= option assigns AXIS1 to the horizontal axis, and the VAXIS= option assigns AXIS2 to the vertical axis.

```
proc gplot data=dowhlc2;
   plot dow*date / haxis=axis1
                   vaxis=axis2;
run;
quit;
```

Example 2: Specifying Logarithmic Axes

Features:

AXIS statement options
- LABEL=
- LENGTH=
- LOGBASE=
- LOGSTYLE=
- MAJOR=
- MINOR=
- VALUE=

TITLE statement options
- MOVE=

GOPTIONS statement options
- GUNIT

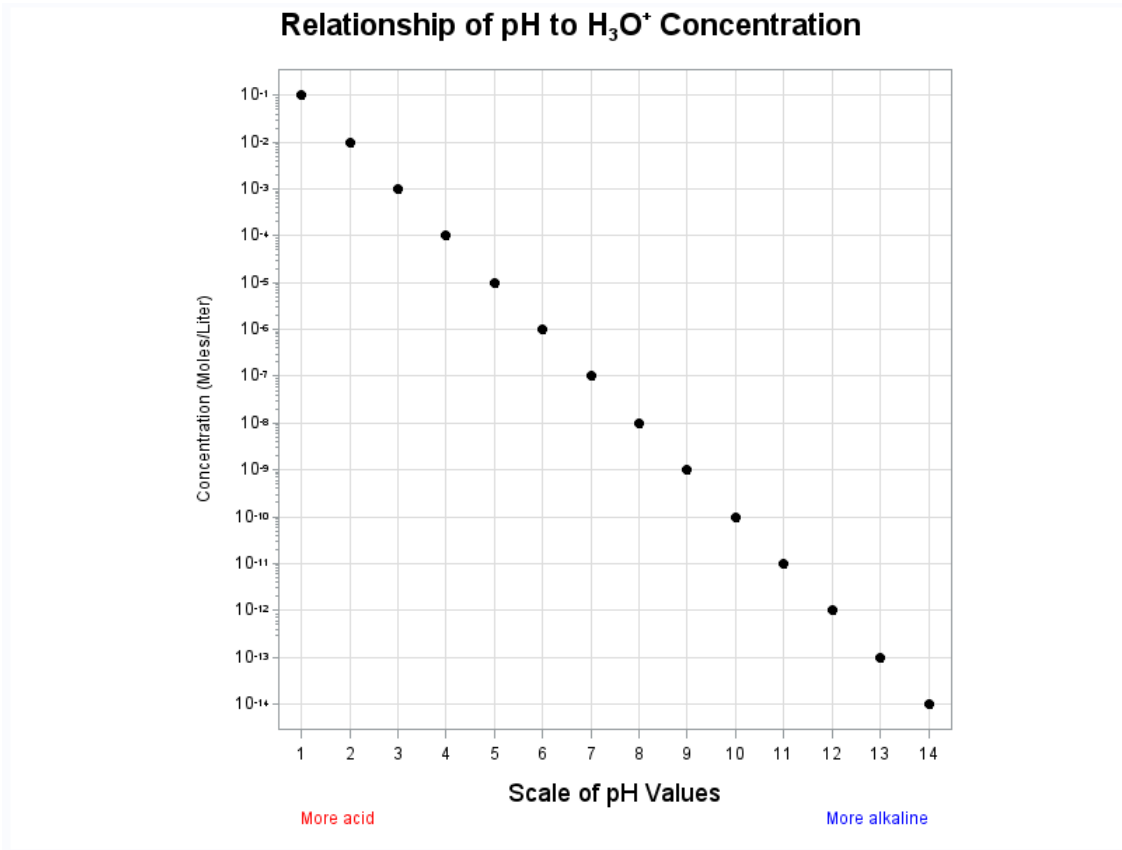
Sample library member: GAXSPLA1

Details

The code for this example is in SAS Sample Library member GAXSPLA1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example illustrates the AXIS statement options LOGBASE= and LOGSTYLE=. The horizontal axis represents pH level. The vertical axis, which represents the concentration of the hydroxide ion expressed as moles per liter, is scaled logarithmically, using a base of 10. In addition, this example shows how the TICK= parameter of the VALUE= option modifies individual tick marks. It also uses the MOVE= option in a TITLE statement to position the title's subscript and superscript text.

Output



Program

```
goptions reset=all  gunit=pct;

data concentr;
   input pH conc;
   datalines;
1  1E-1
2  1E-2
3  1E-3
4  1E-4
5  1E-5
6  1E-6
7  1E-7
8  1E-8
9  1E-9
10 1E-10
11 1E-11
12 1E-12
13 1E-13
14 1E-14
;
run;
```

```
title1 h=3.7 "Relationship of pH to H"
       move=(-0,-.75) h=2 "3"
       move=(+0,+.75) h=3.7 "O"
       move=(+0,+1.5) h=2 "+"
       move=(-0,-1.5) h=3.7  " Concentration";


symbol value=dot color=black height=2;

axis1 label=(h=3 "Scale of pH Values"
             justify=left color=red h=2 "More acid"
             justify=right color=blue "More alkaline")
      minor=none
      length=60
      offset=(2,2);

axis2 logbase=10
      logstyle=expand
      label=(angle=90 h=2 color=black
            "Concentration (Moles/Liter)" )
      value=(tick=1 "10" height=1.2 "-14"
             tick=2 "10" height=1.2 "-13"
             tick=3 "10" height=1.2 "-12"
             tick=4 "10" height=1.2 "-11"
             tick=5 "10" height=1.2 "-10"
             tick=6 "10" height=1.2 "-9"
             tick=7 "10" height=1.2 "-8"
             tick=8 "10" height=1.2 "-7"
             tick=9 "10" height=1.2 "-6"
             tick=10 "10" height=1.2 "-5"
             tick=11 "10" height=1.2 "-4"
             tick=12 "10" height=1.2 "-3"
             tick=13 "10" height=1.2 "-2"
             tick=14 "10" height=1.2 "-1")
      offset=(3,3);

proc gplot data= concentr;
   plot conc*pH / haxis=axis1
                  vaxis=axis2
                  autohref chref=graydd
                  autovref cvref=graydd;
run;
quit;
```

Program Description

Set the graphics environment. The GUNIT option specifies the default unit of measure to use with height specifications.

```
goptions reset=all  gunit=pct;
```

Create the data set. The data set is named CONCENTR and it contains the pH values and the concentration amount.

```
data concentr;
   input pH conc;
   datalines;
1  1E-1
2  1E-2
```

```
3  1E-3
4  1E-4
5  1E-5
6  1E-6
7  1E-7
8  1E-8
9  1E-9
10 1E-10
11 1E-11
12 1E-12
13 1E-13
14 1E-14
;
run;
```

Define the title. The MOVE= option positions subscript 3 and superscript +. Each new position is relative to the last position specified by the MOVE= option.

```
title1 h=3.7 "Relationship of pH to H"
       move=(-0,-.75) h=2 "3"
       move=(+0,+.75) h=3.7 "O"
       move=(+0,+1.5) h=2 "+"
       move=(-0,-1.5) h=3.7  " Concentration";
```

Define the symbol characteristics.

```
symbol value=dot color=black height=2;
```

Define the horizontal axis characteristics. The LABEL= option uses the JUSTIFY= suboption to create a descriptive two-line label that replaces the variable name pH. MINOR=NONE removes all minor tick marks. The LENGTH= option controls the length of the horizontal axis. The OFFSET= option specifies the distance from the first and last major tick marks to the ends of the axis line.

```
axis1 label=(h=3 "Scale of pH Values"
             justify=left color=red h=2 "More acid"
             justify=right color=blue "More alkaline")
      minor=none
      length=60
      offset=(2,2);
```

Define the vertical axis characteristics. LOGBASE=10 scales the vertical axis logarithmically, using a base of 10. Each major tick mark represents a power of 10. LOGSTYLE=EXPAND displays minor tick marks in logarithmic progression. The LABEL= option uses the ANGLE= suboption to place the label parallel to the vertical axis. The VALUE= option displays the major tick mark values as 10 plus an exponent. The HEIGHT= suboption for each TICK= specification affects only the text following it.

```
axis2 logbase=10
      logstyle=expand
      label=(angle=90 h=2 color=black
            "Concentration (Moles/Liter)" )
      value=(tick=1 "10" height=1.2 "-14"
             tick=2 "10" height=1.2 "-13"
             tick=3 "10" height=1.2 "-12"
             tick=4 "10" height=1.2 "-11"
             tick=5 "10" height=1.2 "-10"
             tick=6 "10" height=1.2 "-9"
```

```
                tick=7 "10" height=1.2 "-8"
                tick=8 "10" height=1.2 "-7"
                tick=9 "10" height=1.2 "-6"
                tick=10 "10" height=1.2 "-5"
                tick=11 "10" height=1.2 "-4"
                tick=12 "10" height=1.2 "-3"
                tick=13 "10" height=1.2 "-2"
                tick=14 "10" height=1.2 "-1")
        offset=(3,3);
```

Generate the plot and assign AXIS definitions. AXIS1 modifies the horizontal axis and AXIS2 modifies the vertical axis. The AUTOHREF and AUTOVREF options draw reference lines at all major tick marks on both axes. The CHREF and CVREF options specify the color for these reference lines.

```
proc gplot data= concentr;
   plot conc*pH / haxis=axis1
                  vaxis=axis2
                  autohref chref=graydd
                  autovref cvref=graydd;
run;
quit;
```

Example 3: Rotating Plot Symbols through the Color List

Features:

GOTIONS statement options
- COLORS=
- BORDER

LEGEND statement options
- LABEL=
- JUSTIFY=

SYMBOL statement options
- VALUE=

TITLE statement options
- JUSTIFY=
- HEIGHT=

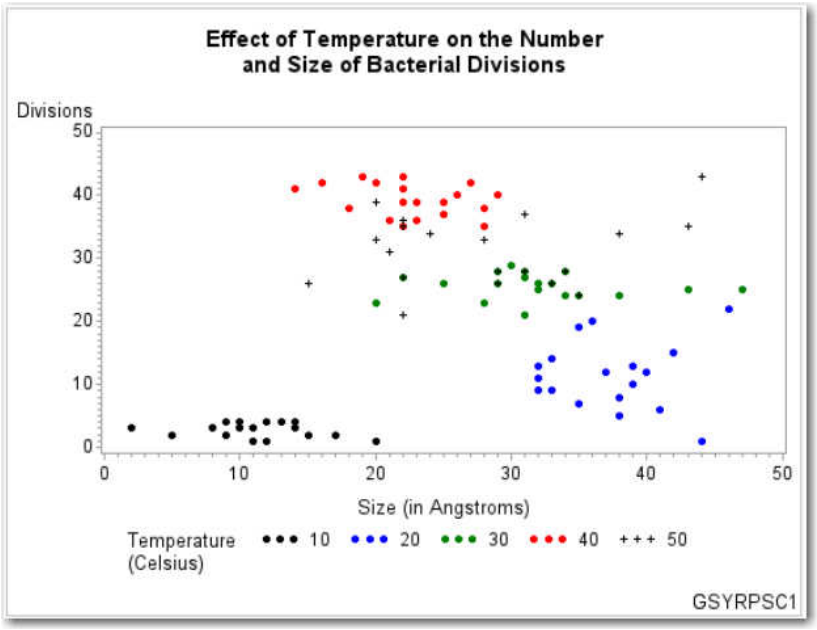
Sample library member: GSYRPSC1

Details

The code for this example is in SAS Sample Library member GSYRPSC1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example specifies a plot symbol in a SYMBOL statement and rotates the symbol through the specified color list. Temperature values in the data are represented by the same plot symbol in a different color. The example also shows how default symbol sequencing provides a default plot symbol if a plot needs more plot symbols than are defined. It also uses a LEGEND statement to specify a two-line legend label and to align the label with the legend values.

Output



Program

```
goptions reset=all  border
         colors=(black blue green red)
          ;

data bacteria;
   input temp div mass life @@;
   datalines;
10 3 10 1  20 22 46 0  30 23 20 9   40 42 16 16  50 33 20 6
10 1 11 2  20 01 44 2  30 21 31 10  40 41 14 12  50 31 21 7
10 4 14 3  20 13 32 4  30 24 34 9   40 43 22 14  50 34 24 2
10 2 09 2  20 12 40 6  30 26 29 8   40 42 20 16  50 26 29 4
10 3 08 3  20 09 33 8  30 24 38 11  40 39 23 18  50 34 38 2
10 2 09 1  20 08 38 1  30 25 47 14  40 38 18 12  50 43 44 1
10 4 10 3  20 15 42 3  30 29 30 14  40 35 22 14  50 39 20 8
10 3 11 2  20 20 36 5  30 28 31 9   40 40 26 15  50 28 31 0
10 2 15 3  20 19 35 7  30 26 25 11  40 39 25 17  50 26 15 4
10 4 12 3  20 14 33 2  30 27 22 8   40 36 23 12  50 27 22 3
10 4 13 3  20 12 37 4  30 26 33 9   40 42 27 14  50 26 33 5
10 2 17 1  20 10 39 6  30 25 43 13  40 40 29 16  50 35 43 7
10 3 14 1  20 08 38 4  30 28 34 8   40 38 28 14  50 28 34 4
```

```
10 1 12 1  20 06 41 2  30 26 32 14  40 36 21 12  50 21 22 2
10 1 11 4  20 09 32 2  30 27 31 8   40 39 22 12  50 37 31 2
10 1 20 2  20 11 32 5  30 25 32 16  40 41 22 15  50 35 22 5
10 4 09 2  20 13 39 1  30 28 29 12  40 43 19 15  50 28 29 1
10 3 02 2  20 09 32 5  30 26 32 9   40 39 22 15  50 36 22 5
10 2 05 3  20 07 35 4  30 24 35 15  40 37 25 14  50 24 35 4
10 3 08 1  20 05 38 6  30 23 28 9   40 35 28 16  50 33 28 6
;
proc sort data=bacteria;
   by temp;
run;

title1 ls=1.5 h=2 "Effect of Temperature on the Number"
  j=c "and Size of Bacterial Divisions";
footnote1  j=r "GSYRPSC1";

symbol1  value=dot;

axis1 label=("Size (in Angstroms)") ;
axis2 label=("Divisions");

legend1 label=(position=(top left)
               "Temperature" j=l "(Celsius)")
        ;

proc gplot data= bacteria;
     plot div*mass=temp / haxis=axis1
                          vaxis=axis2
                          legend=legend1;
run;
quit;
```

Program Description

Set the graphics environment. The COLORS= graphics option specifies the color list. This list is used by the SYMBOL statement if a color option is not specified on that statement.

```
goptions reset=all  border
         colors=(black blue green red)
          ;
```

Create the data set. BACTERIA contains information about the number and size of bacterial divisions at various temperatures.

```
data bacteria;
   input temp div mass life @@;
   datalines;
10 3 10 1  20 22 46 0  30 23 20 9   40 42 16 16  50 33 20 6
10 1 11 2  20 01 44 2  30 21 31 10  40 41 14 12  50 31 21 7
10 4 14 3  20 13 32 4  30 24 34 9   40 43 22 14  50 34 24 2
10 2 09 2  20 12 40 6  30 26 29 8   40 42 20 16  50 26 29 4
10 3 08 3  20 09 33 8  30 24 38 11  40 39 23 18  50 34 38 2
10 2 09 1  20 08 38 1  30 25 47 14  40 38 18 12  50 43 44 1
10 4 10 3  20 15 42 3  30 29 30 14  40 35 22 14  50 39 20 8
10 3 11 2  20 20 36 5  30 28 31 9   40 40 26 15  50 28 31 0
10 2 15 3  20 19 35 7  30 26 25 11  40 39 25 17  50 26 15 4
10 4 12 3  20 14 33 2  30 27 22 8   40 36 23 12  50 27 22 3
10 4 13 3  20 12 37 4  30 26 33 9   40 42 27 14  50 26 33 5
```

```
10 2 17 1  20 10 39 6  30 25 43 13  40 40 29 16  50 35 43 7
10 3 14 1  20 08 38 4  30 28 34 8   40 38 28 14  50 28 34 4
10 1 12 1  20 06 41 2  30 26 32 14  40 36 21 12  50 21 22 2
10 1 11 4  20 09 32 2  30 27 31 8   40 39 22 12  50 37 31 2
10 1 20 2  20 11 32 5  30 25 32 16  40 41 22 15  50 35 22 5
10 4 09 2  20 13 39 1  30 28 29 12  40 43 19 15  50 28 29 1
10 3 02 2  20 09 32 5  30 26 32 9   40 39 22 15  50 36 22 5
10 2 05 3  20 07 35 4  30 24 35 15  40 37 25 14  50 24 35 4
10 3 08 1  20 05 38 6  30 23 28 9   40 35 28 16  50 33 28 6
;
proc sort data=bacteria;
   by temp;
run;
```

Define title and footnote. LS= adds some space between the title and the border of the graphics output area. Positioning J= in the middle of the title breaks the title into two lines. H= specifies the size of the entire title.

```
title1 ls=1.5 h=2 "Effect of Temperature on the Number"
  j=c "and Size of Bacterial Divisions";
footnote1  j=r "GSYRPSC1";
```

Define symbol shape. The VALUE= option specifies a dot for the plot symbol. Because no color is specified, the symbol is rotated through the graphics option color list. Because the plot needs a fifth symbol, the default plus sign is rotated into the color list to provide that symbol.

```
symbol1  value=dot;
```

Define axis characteristics.

```
axis1 label=("Size (in Angstroms)") ;
axis2 label=("Divisions");
```

Define legend characteristics. The LABEL= option specifies text for the legend label. J=L specifies a new line and left-justifies the second string under the first. The POSITION= option aligns the top label line with the first (and in this case only) value row.

```
legend1 label=(position=(top left)
               "Temperature" j=l "(Celsius)")
        ;
```

Generate the plot.

```
proc gplot data= bacteria;
     plot div*mass=temp / haxis=axis1
                          vaxis=axis2
                          legend=legend1;
run;
quit;
```

Example 4: Creating and Modifying Box Plots

Features: GOTIONS statement options
BORDER

AXIS statement options
LABEL=

LENGTH=
OFFSET=
VALUE=

SYMBOL statement options
BWIDTH=
CO=
CV=
HEIGHT=
INTERPOL=
VALUE=
WIDTH=

Sample library member: GSYCMBP1

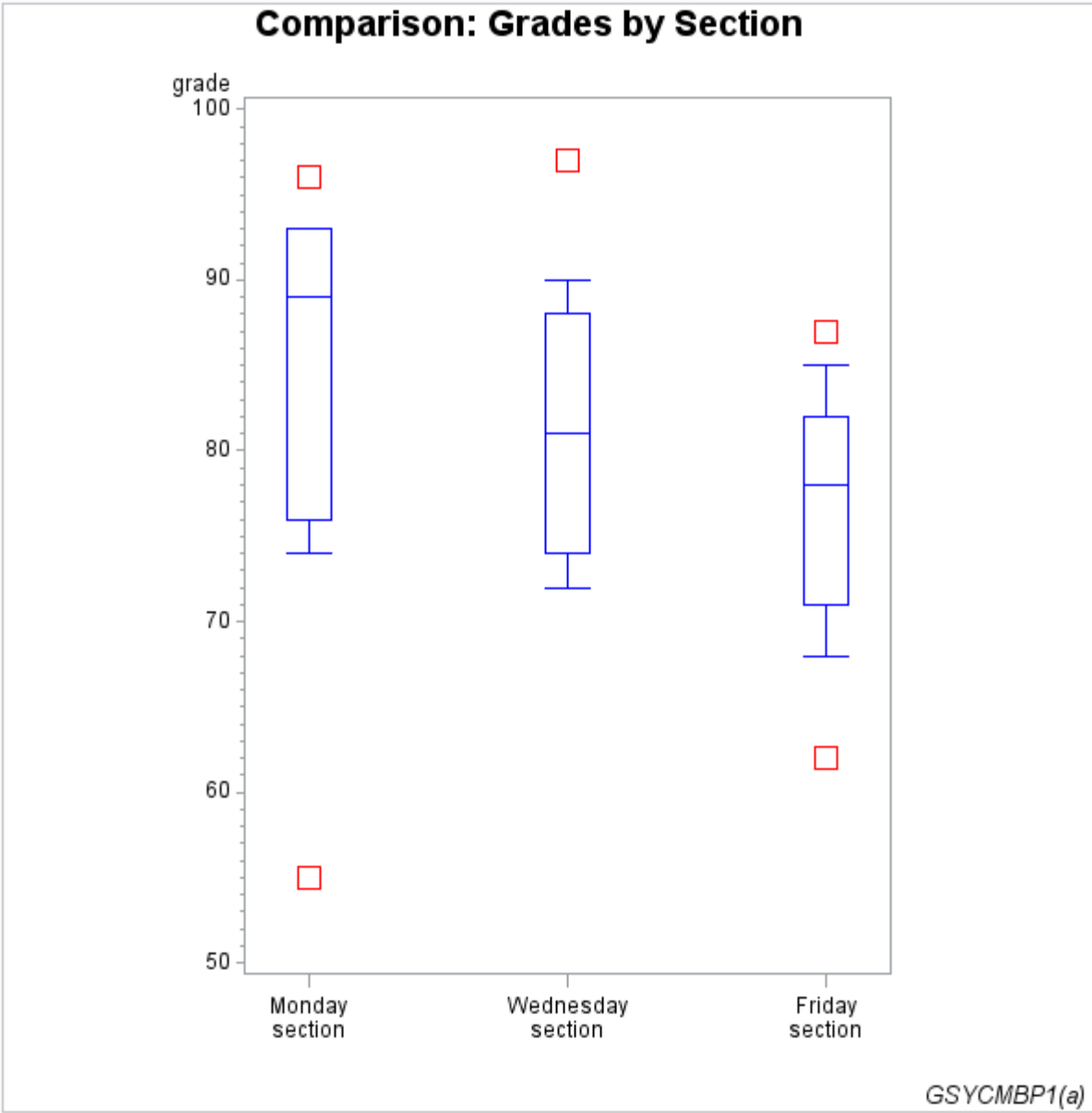
Details

The code for this example is in SAS Sample Library member GSYCMBP1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

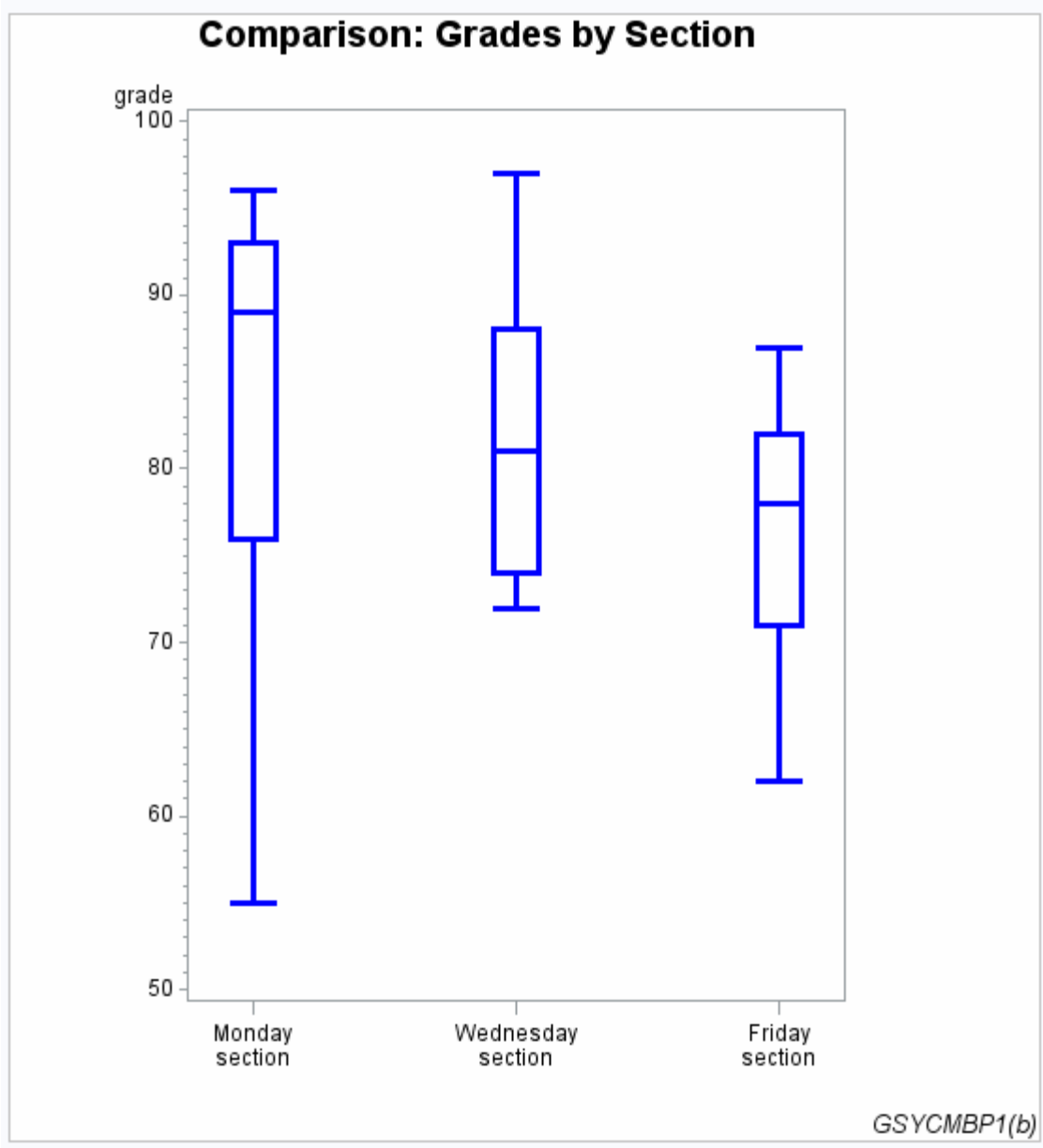
This example shows how to create box plots and how to specify SYMBOL definitions so that data outside the box-plot range can be represented with data points. It also shows how to change a box plot's percentile range to see whether the new range encompasses the data.

Output

The first plot in the example uses a SYMBOL definition with INTERPOL=BOXT20. to specify a box plot with whisker tops at the 80th percentile. It also specifies whisker bottoms at the 20th percentile. Data points that are outside this percentile range are represented with squares.



As illustrated in the following output, the example then changes the SYMBOL definition to INTERPOL=BOXT10. This definition expands the whisker range to the 90th percentile for tops and the 10th percentile for bottoms. There are no data points outside the new percentile range.



Program

```
goptions reset=all border;

data grades;
     input section $ grade @@;
     datalines;
A 74 A 89 A 91 A 76 A 87 A 93 A 93 A 96 A 55
B 72 B 72 B 84 B 81 B 97 B 78 B 88 B 90 B 74
C 62 C 74 C 71 C 87 C 68 C 78 C 80 C 85 C 82
;

title1 "Comparison: Grades by Section";
footnote1 j=r  "GSYCMBP1(a) ";

symbol interpol=boxt20 /* box plot              */
```

```
       co=blue         /* box and whisker color */
       bwidth=4        /* box width             */
       value=square    /* plot symbol           */
       cv=red          /* plot symbol color     */
       height=2;       /* symbol height         */

axis1 label=none
      value=(t=1 "Monday" j=c "section"
             t=2 "Wednesday" j=c "section"
             t=3 "Friday" j=c "section")
      offset=(5,5)
      length=50;

proc gplot data= grades;
   plot grade*section / haxis=axis1
                        vaxis=50 to 100 by 10;
run;

footnote j=r GSYCMBP1(b);

symbol interpol=boxt10 width=2;

plot grade*section / haxis=axis1
                        vaxis=50 to 100 by 10;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set. GRADES contains codes to identify each class section, and the grades scored by students in each section.

```
data grades;
     input section $ grade @@;
     datalines;
A 74 A 89 A 91 A 76 A 87 A 93 A 93 A 96 A 55
B 72 B 72 B 84 B 81 B 97 B 78 B 88 B 90 B 74
C 62 C 74 C 71 C 87 C 68 C 78 C 80 C 85 C 82
;
```

Define title and footnote.

```
title1 "Comparison: Grades by Section";
footnote1 j=r  "GSYCMBP1(a) ";
```

Define symbol characteristics. INTERPOL=BOXT20 specifies a box plot with tops and bottoms on its whiskers, and the high and low bounds at the 80th and 20th percentiles. The CO= option colors the boxes and whiskers. The BWIDTH= option affects the width of the boxes. The VALUE= option specifies the plot symbol that marks the data points outside the range of the box plot. The CV= option colors the plot symbols. The HEIGHT= option specifies a symbol size.

```
symbol interpol=boxt20 /* box plot              */
       co=blue         /* box and whisker color */
       bwidth=4        /* box width             */
       value=square    /* plot symbol           */
       cv=red          /* plot symbol color     */
```

```
         height=2;       /* symbol height         */
```

Define axis characteristics.

```
axis1 label=none
      value=(t=1 "Monday" j=c "section"
             t=2 "Wednesday" j=c "section"
             t=3 "Friday" j=c "section")
      offset=(5,5)
      length=50;
```

Generate the first plot.

```
proc gplot data= grades;
   plot grade*section / haxis=axis1
                        vaxis=50 to 100 by 10;
run;
```

Define the footnote for the second plot.

```
footnote j=r GSYCMBP1(b);
```

Change symbol characteristics. INTERPOL=BOXT10 changes the high and low bounds to the 90th percentile at the top and the 10th percentile on the bottom. All other symbol characteristics remain unchanged.

```
symbol interpol=boxt10 width=2;
```

Generate the second plot.

```
plot grade*section / haxis=axis1
                     vaxis=50 to 100 by 10;
run;
quit;
```

Example 5: Filling the Area between Plot Lines

Features:

GOTIONS statement options
- BORDER

AXIS statement options
- LABEL=
- MAJOR=
- MINOR=
- OFFSET=
- ORDER=
- WIDTH=

SYMBOL statement options
- CO=
- CV=
- INTERPOL=

Sample library member: GSYFAPL1

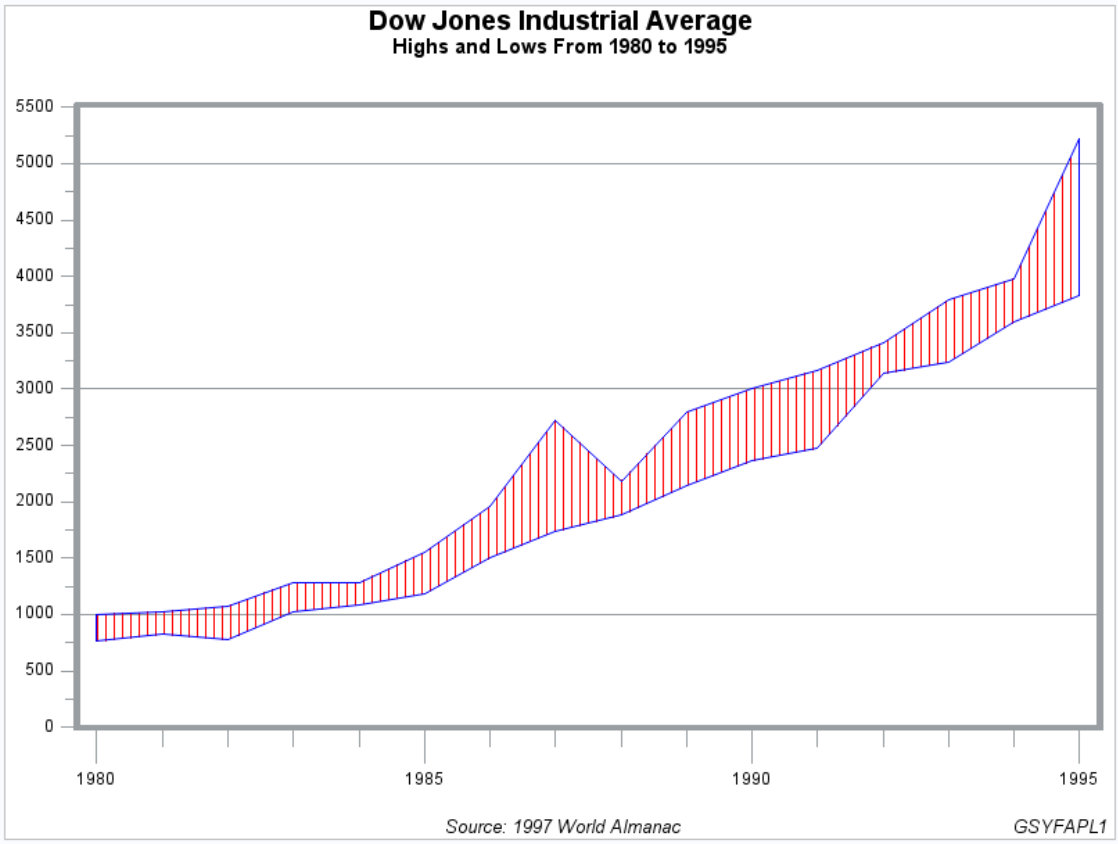
Details

The code for this example is in SAS Sample Library member GSYFAPL1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example shows how to fill the area between two plot lines by concatenating two data sets into one to form a polygon with the data points. It uses a SYMBOL statement to specify a pattern to fill the polygon and to determine the color of the area fill and the outline around the area.

The example plots yearly highs and lows for the Dow Jones Industrial Average. It separates the dependent variables HIGH and LOW to produce an upper plot line and a lower plot line. The dependent variable is named VALUE and the independent variable is named YEAR. When concatenated into one data set, AREA, the data sets form the polygon.

Output



Program

```
goptions reset=all border;

data stocks;
   input year @7  hdate date9. @17 high
              @26 ldate date9. @36 low;
   format hdate ldate date9.;
   datalines;

1980  20NOV1980 1000.17  21APR1980  759.13
1981  27APR1981 1024.05  25SEP1981  824.01
1982  27DEC1982 1070.55  12AUG1982  776.92
1983  29NOV1983 1287.20  03JAN1983 1027.04
1984  06JAN1984 1286.64  24JUL1984 1086.57
1985  16DEC1985 1553.10  04JAN1985 1184.96
1986  02DEC1986 1955.57  22JAN1986 1502.29
1987  25AUG1987 2722.42  19OCT1987 1738.74
1988  21OCT1988 2183.50  20JAN1988 1879.14
1989  09OCT1989 2791.41  03JAN1989 2144.64
1990  16JUL1990 2999.75  11OCT1990 2365.10
1991  31DEC1991 3168.83  09JAN1991 2470.30
1992  01JUN1992 3413.21  09OCT1992 3136.58
1993  29DEC1993 3794.33  20JAN1993 3241.95
1994  31JAN1994 3978.36  04APR1994 3593.35
1995  13DEC1995 5216.47  30JAN1995 3832.08
;

data high(keep=year value)
     low(keep=year value);
   set stocks;
   value=high; output high;
   value=low; output low;
run;

proc sort data=low;
   by descending year;
run;

data area;
   set high low;
run;

title1 "Dow Jones Industrial Average";
title2  "Highs and Lows From 1980 to 1995";
footnote " Source: 1997 World Almanac"
         j=r "GSYFAPL1 ";

symbol interpol=m3n90
       cv=red
       co=blue;

axis1 order=(1980 to 1995 by 5)
      label=none
      major=(height=2)
      minor=(number=4 height=1)
      offset=(2,2)
      width=3;
axis2 order=(0 to 5500 by 500)
```

```
      label=none
      major=(height=1.5)  offset=(0,0)
      minor=(number=1 height=1);

proc gplot data=area;
   plot value*year / haxis=axis1
                     vaxis=axis2
                     vref=(1000 3000 5000);
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set. STOCKS contains yearly highs and lows for the Dow Jones Industrial Average, and the dates of the high and low values each year.

```
data stocks;
   input year @7  hdate date9. @17 high
              @26 ldate date9. @36 low;
   format hdate ldate date9.;
   datalines;

1980  20NOV1980 1000.17  21APR1980  759.13
1981  27APR1981 1024.05  25SEP1981  824.01
1982  27DEC1982 1070.55  12AUG1982  776.92
1983  29NOV1983 1287.20  03JAN1983 1027.04
1984  06JAN1984 1286.64  24JUL1984 1086.57
1985  16DEC1985 1553.10  04JAN1985 1184.96
1986  02DEC1986 1955.57  22JAN1986 1502.29
1987  25AUG1987 2722.42  19OCT1987 1738.74
1988  21OCT1988 2183.50  20JAN1988 1879.14
1989  09OCT1989 2791.41  03JAN1989 2144.64
1990  16JUL1990 2999.75  11OCT1990 2365.10
1991  31DEC1991 3168.83  09JAN1991 2470.30
1992  01JUN1992 3413.21  09OCT1992 3136.58
1993  29DEC1993 3794.33  20JAN1993 3241.95
1994  31JAN1994 3978.36  04APR1994 3593.35
1995  13DEC1995 5216.47  30JAN1995 3832.08
;
```

Restructure the data so that it defines a closed area. Create the temporary data sets HIGH and LOW.

```
data high(keep=year value)
     low(keep=year value);
   set stocks;
   value=high; output high;
   value=low; output low;
run;
```

Reverse order of the observations in LOW.

```
proc sort data=low;
   by descending year;
run;
```

Concatenate HIGH and LOW to create data set AREA.

```
data area;
   set high low;
run;
```

Define titles and footnote.

```
title1 "Dow Jones Industrial Average";
title2  "Highs and Lows From 1980 to 1995";
footnote " Source: 1997 World Almanac"
         j=r "GSYFAPL1 ";
```

Define symbol characteristics. The INTERPOL= option specifies a map pattern or a plot pattern to fill the polygon formed by the data points. The pattern consists of medium-density parallel lines at 90 degrees. The CV= option colors the pattern fill. The CO= option colors the outline of the area. (If the CO= option is not used, the outline is the color of the area.)

```
symbol interpol=m3n90
       cv=red
       co=blue;
```

Define axis characteristics. The ORDER= option places the major tick marks at 5-year intervals.

```
axis1 order=(1980 to 1995 by 5)
      label=none
      major=(height=2)
      minor=(number=4 height=1)
      offset=(2,2)
      width=3;
axis2 order=(0 to 5500 by 500)
      label=none
      major=(height=1.5)  offset=(0,0)
      minor=(number=1 height=1);
```

Generate the plot using data set AREA.

```
proc gplot data=area;
   plot value*year / haxis=axis1
                     vaxis=axis2
                     vref=(1000 3000 5000);
run;
quit;
```

Example 6: Enhancing Titles

Features:

GOPTIONS statement options
- BORDER

TITLE statement options
- ANGLE=
- BCOLOR=
- BLANK=
- BOX=
- COLOR=
- DRAW=
- FONT=

HEIGHT=
MOVE=
ROTATE=
"*text-string*"
UNDERLIN=

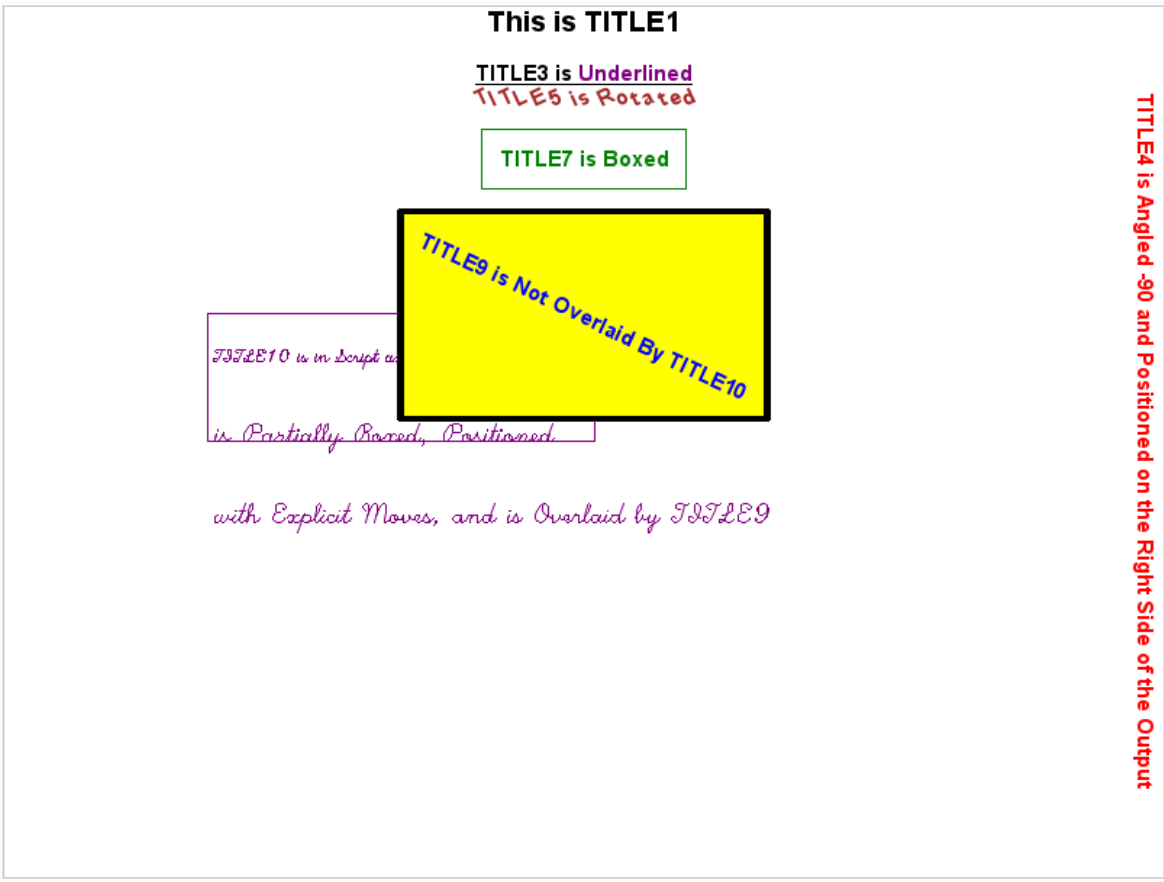
Sample library member: GTIENTI1

Details

The code for this example is in SAS Sample Library member GTIENTI1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example illustrates some ways that you can format title text. The same options can be used to format footnotes.

Output



Program

```
goptions reset=all border;

title1 "This is TITLE1"
height=4;

title3 underlin=1
       "TITLE3 is"
       color=purple
       " Underlined";

title4 color=red
       angle=-90
       "TITLE4 is Angled -90 and Positioned on the Right Side of the
Output";

title5
       color=brown
       rotate=25
       "TITLE5 is Rotated";

title7 color=green
       box=1
       "TITLE7 is Boxed";

title9 color=black
       box=3
       blank=yes
       bcolor=yellow
       color=blue
       angle=-25
       "TITLE9 is Not Overlaid By TITLE10";

title10 color=purple
        draw=(20,35 20,27 58,27 58,35 20,35)cells
        move=(20,32)
        font=script
        "TITLE10 is in Script and "
        move=(20,27)
        height=2
        "is Partially Boxed, Positioned"
        move=(20,22)
        height=2
        "with Explicit Moves, and is Overlaid by TITLE9";

proc gslide;
run;
quit;
```

Program Description

Set the graphics environment. BORDER draws a border around the graph.

```
goptions reset=all border;
```

Define TITLE1. TITLE1 uses the default font and height defined in the default style. The HEIGHT= option sets the height of the text.

```
title1 "This is TITLE1"
height=4;
```

Define TITLE3. The UNDERLIN= option underlines both text strings.

```
title3 underlin=1
       "TITLE3 is"
       color=purple
       " Underlined";
```

Define TITLE4. The ANGLE= option tilts the line of text clockwise 90 degrees and places it at the right edge of the output, centered vertically.

```
title4 color=red
       angle=-90
       "TITLE4 is Angled -90 and Positioned on the Right Side of the
Output";
```

Define TITLE5. The ROTATE= option rotates each character in the text string at the specified angle.

```
title5
       color=brown
       rotate=25
       "TITLE5 is Rotated";
```

Define TITLE7. The BOX= option draws a green box around the text with the thinnest of the 4 available box lines.

```
title7 color=green
       box=1
       "TITLE7 is Boxed";
```

Define TITLE9. The BLANK= option prevents the boxed title from being overwritten by TITLE10. The first COLOR= option specifies the color of the box border, and the BCOLOR= option specifies the background color of the box. The second COLOR= option specifies the text color.

```
title9 color=black
       box=3
       blank=yes
       bcolor=yellow
       color=blue
       angle=-25
       "TITLE9 is Not Overlaid By TITLE10";
```

Define TITLE10. In this statement, the DRAW= option draws a box around the first two text strings. The BOX= option is turned off by the MOVE= option that uses absolute coordinates and causes a text break.

```
title10 color=purple
        draw=(20,35 20,27 58,27 58,35 20,35)cells
        move=(20,32)
        font=script
        "TITLE10 is in Script and "
        move=(20,27)
        height=2
        "is Partially Boxed, Positioned"
        move=(20,22)
        height=2
        "with Explicit Moves, and is Overlaid by TITLE9";
```

Display titles. All existing titles are automatically displayed by the procedure.

```
proc gslide;
run;
quit;
```

Example 7: Using BY-group Processing to Generate a Series of Charts

Features:

AXIS statement options
- LABEL=
- MAJOR=
- MINOR=
- NOPLANE
- ORDER=
- SPLIT=
- STYLE=
- VALUE=

BY statement

OPTIONS statement options
- NOBYLINE

PATTERN statement options
- COLOR=

TITLE statement
- #BYVAL

Sample library member: GBYGMSC1

Details

The code for this example is in SAS Sample Library member GBYGMSC1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

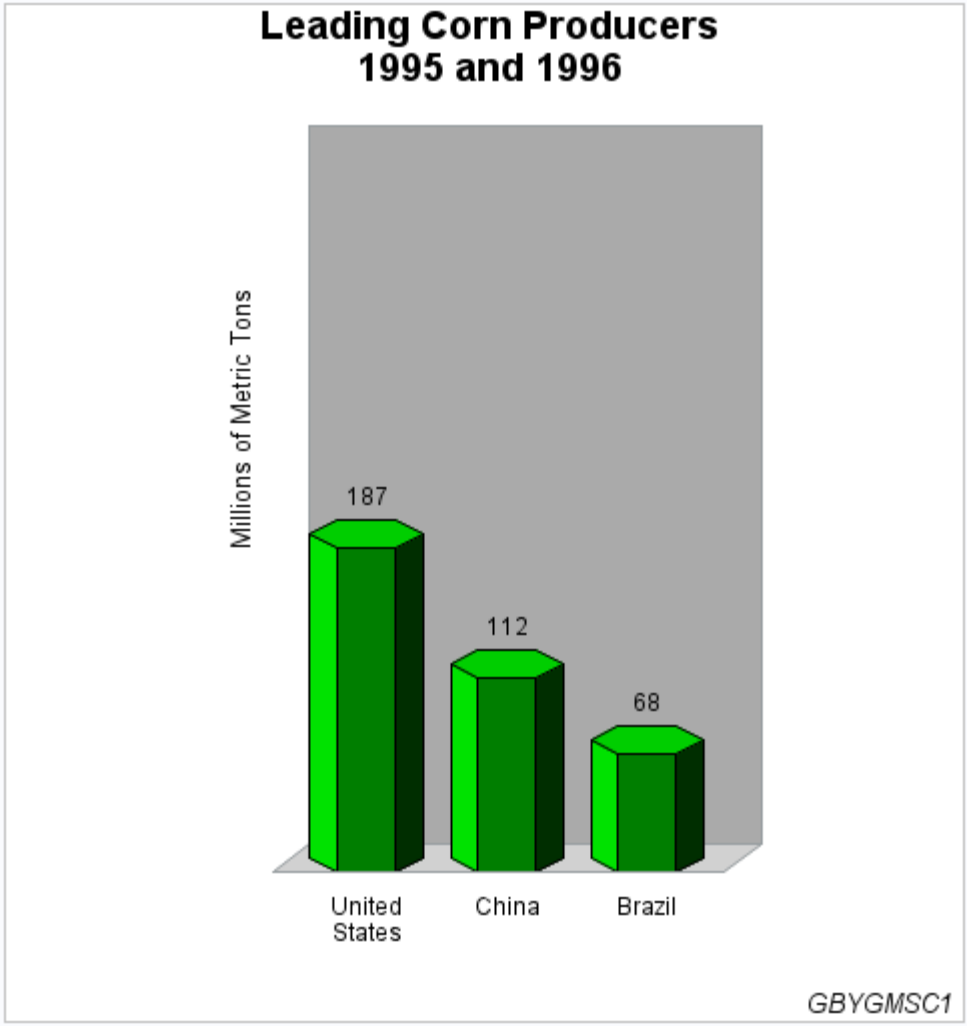
This example uses a BY statement with the GCHART procedure to produce a separate three-dimensional vertical bar chart for each value of the BY variable TYPE. The three charts, which are shown in Output 25.4 on page 552, Output 25.5 on page 553, and Output 25.6 on page 554, show leading grain producers for 1995 and 1996.

The program suppresses the default BY lines and instead uses #BYVAL in the TITLE statement text string to include the BY variable value in the title for each chart. The AXIS1 statement that is assigned to the vertical (response) axis is automatically applied to all three graphs generated by the BY statement. This AXIS statement removes all the elements of the response axis except the label. The same AXIS statement also includes an ORDER= option. Because this option is applied to all the graphs, it ensures that they all use the same scale of response values.

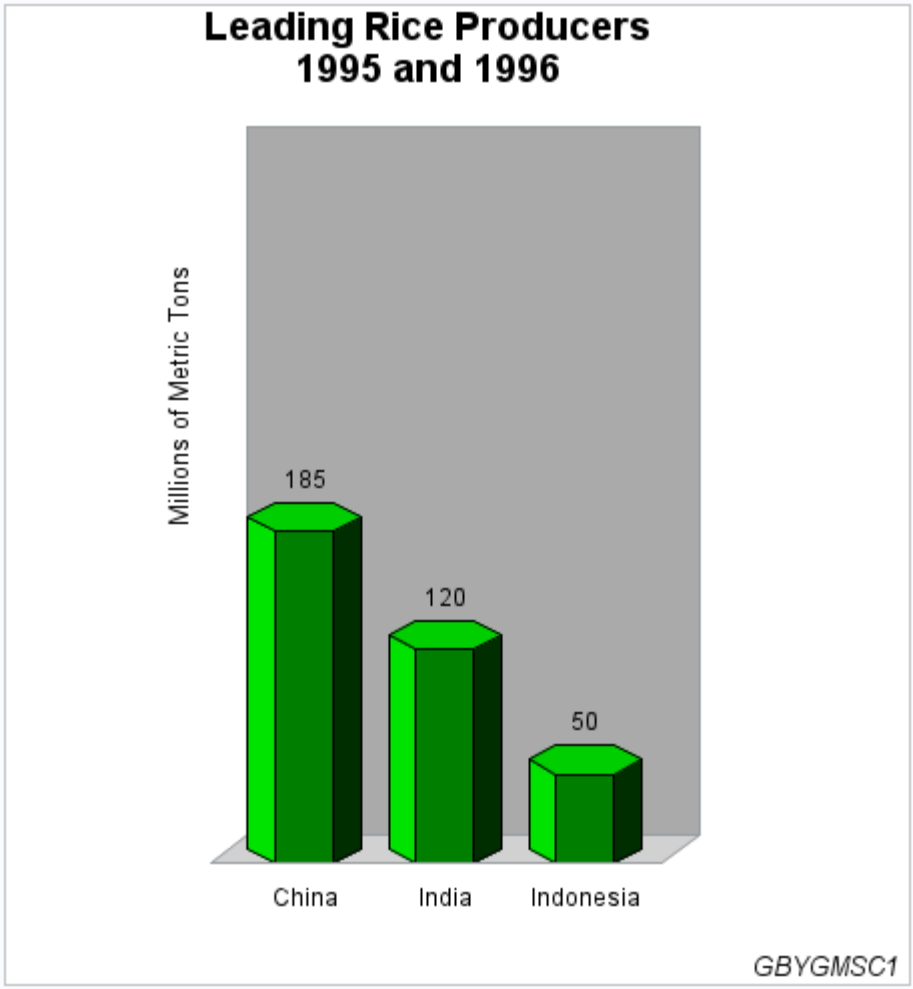
Because no subgroups are specified and the PATTERNID= option is omitted, the color specified in the single PATTERN statement is used by all of the bars.

Output

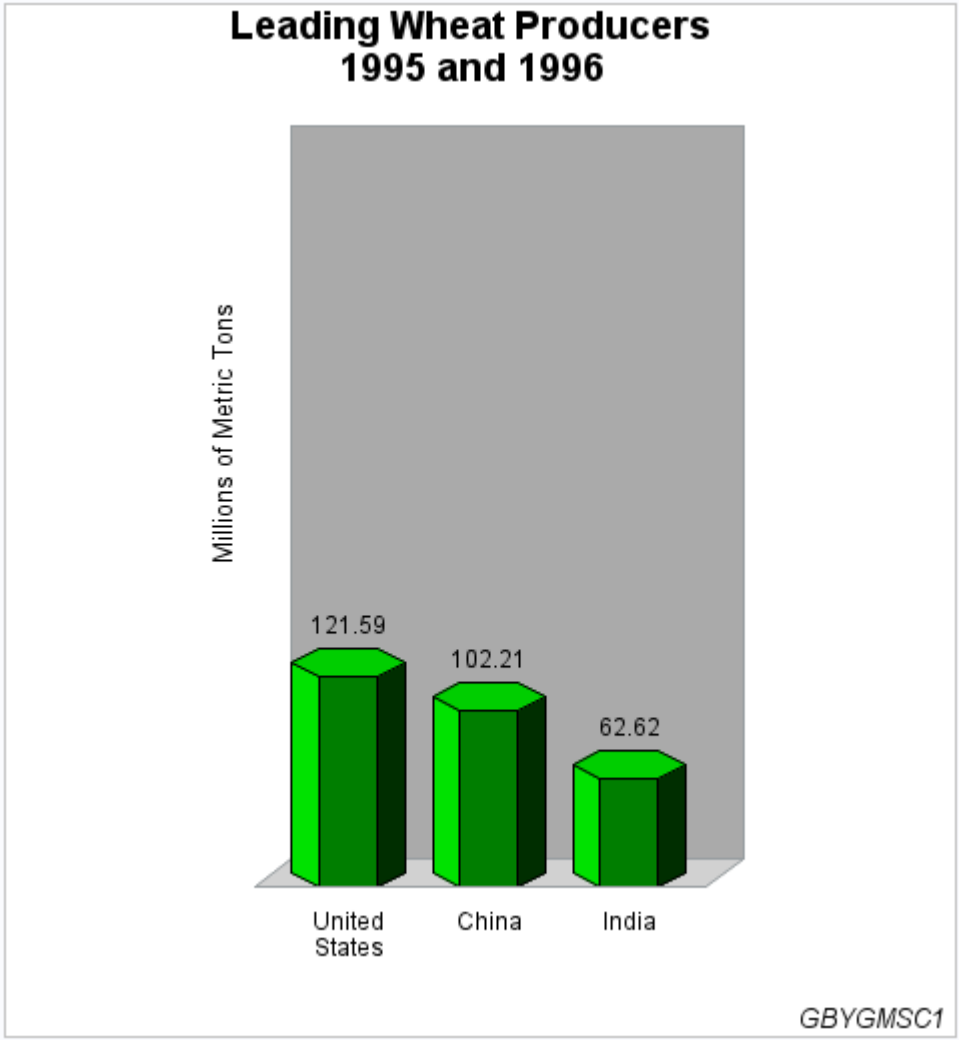
Output 25.1 *Output for BY Value Corn*



Output 25.2 *Output for BY Value Rice*



Output 25.3 *Output for BY Value Wheat*



Program

```
goptions reset=all border;

data grainldr;
   length country $ 3 type $ 5;
   input year country $ type $ amount;
   megtons=amount/1000;
   datalines;
1995 BRZ  Wheat    1516
1995 BRZ  Rice     11236
1995 BRZ  Corn     36276
1995 CHN  Wheat    102207
1995 CHN  Rice     185226
1995 CHN  Corn     112331
1995 INS  Wheat    .
1995 INS  Rice     49860
1995 INS  Corn     8223
1995 USA  Wheat    59494
```

```
1995 USA  Rice     7888
1995 USA  Corn     187300
1996 BRZ  Wheat    3302
1996 BRZ  Rice     10035
1996 BRZ  Corn     31975
1996 IND  Wheat    62620
1996 IND  Rice     120012
1996 IND  Corn     8660
1996 USA  Wheat    62099
1996 USA  Rice     7771
;

proc format;
  value $country "BRZ" = "Brazil"
                 "CHN" = "China"
                 "IND" = "India"
                 "INS" = "Indonesia"
                 "USA" = "United States";
run;

options nobyline;
title1 "Leading #byval(type) Producers"
        j=c "1995 and 1996";
footnote1 j=r  "GBYGMSC1 ";

pattern1 color=green;

axis1 order=(0 to 550 by 100)
      label=(angle=90 "Millions of Metric Tons")
      major=none
      minor=none
      value=none
      style=0
      noplane;

axis2 label=none
      split=" ";

proc sort data=grainldr out=temp;
     by type;
run;

proc gchart data=temp (where=(megtons gt 31));
   by type;
   format country $country.;
   vbar3d country / sumvar=megtons
                    outside=sum
                    descending
                    shape=hexagon
                    width=8
                    coutline=black
                    cframe=grayaa
                    maxis=axis2
                    raxis=axis1 name="GBYGMSC1";
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set GRAINLDR. GRAINLDR contains data about grain production in five countries for 1995 and 1996. The quantities in AMOUNT are in thousands of metric tons. MEGTONS converts these quantities to millions of metric tons.

```
data grainldr;
   length country $ 3 type $ 5;
   input year country $ type $ amount;
   megtons=amount/1000;
   datalines;
1995 BRZ  Wheat    1516
1995 BRZ  Rice     11236
1995 BRZ  Corn     36276
1995 CHN  Wheat    102207
1995 CHN  Rice     185226
1995 CHN  Corn     112331
1995 INS  Wheat    .
1995 INS  Rice     49860
1995 INS  Corn     8223
1995 USA  Wheat    59494
1995 USA  Rice     7888
1995 USA  Corn     187300
1996 BRZ  Wheat    3302
1996 BRZ  Rice     10035
1996 BRZ  Corn     31975
1996 IND  Wheat    62620
1996 IND  Rice     120012
1996 IND  Corn     8660
1996 USA  Wheat    62099
1996 USA  Rice     7771
;
```

Create a format for the values of COUNTRY.

```
proc format;
  value $country "BRZ" = "Brazil"
                 "CHN" = "China"
                 "IND" = "India"
                 "INS" = "Indonesia"
                 "USA" = "United States";
run;
```

Suppress the default BY line and define a title that includes the BY-value. #BYVAL inserts the value of the BY variable COUNTRY into the title of each report.

```
options nobyline;
title1 "Leading #byval(type) Producers"
        j=c "1995 and 1996";
footnote1 j=r  "GBYGMSC1 ";
```

Specify a color for the bars.

```
pattern1 color=green;
```

Define the axis characteristics for the response axes. The ORDER= option specifies the range of values for the response axes. ANGLE=90 in the LABEL=

option rotates the label 90 degrees. All the other options remove axis elements. The MAJOR=, MINOR=, and VALUE= options remove the tick marks and values. STYLE=0 removes the line. The NOPLANE option removes the three-dimensional plane.

```
axis1 order=(0 to 550 by 100)
      label=(angle=90 "Millions of Metric Tons")
      major=none
      minor=none
      value=none
      style=0
      noplane;
```

Define midpoint axis characteristics. The SPLIT= option defines one or more characters, such as a blank, that the AXIS statement uses to break axis values into multiple lines.

```
axis2 label=none
      split=" ";
```

Sort data according to values of BY variable. The data must be sorted before running PROC GCHART with the BY statement.

```
proc sort data=grainldr out=temp;
     by type;
run;
```

Generate the vertical bar charts using a BY statement. The BY statement produces a chart for each value of SITE. The FORMAT statement assigns the $COUNTRY. format to the chart variable. Assigning AXIS1 to the RAXIS= option causes all three charts to have the same response axis.

```
proc gchart data=temp (where=(megtons gt 31));
   by type;
   format country $country.;
   vbar3d country / sumvar=megtons
                    outside=sum
                    descending
                    shape=hexagon
                    width=8
                    coutline=black
                    cframe=grayaa
                    maxis=axis2
                    raxis=axis1 name="GBYGMSC1";
run;
quit;
```

Example 8: Combining Graphs and Reports in a Web Page

Features: AXIS statement options
- LENGTH=
- MINOR=
- ORDER=
- VALUE=

BY statement

GOPTIONS statement options
- BORDER
- DEVICE=
- TRANSPARENCY

LEGEND statement options
- LABEL=
- SHAPE=

ODS HTML statement options
- BODY=
- CONTENTS=
- FRAME=
- PATH=
- NOGTITLE

OPTIONS statement option
- NOBYLINE

TITLE statement option
- #BYVAL

Sample library member: GONCGRW1

Details

The code for this example is in SAS Sample Library member GONCGRW1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

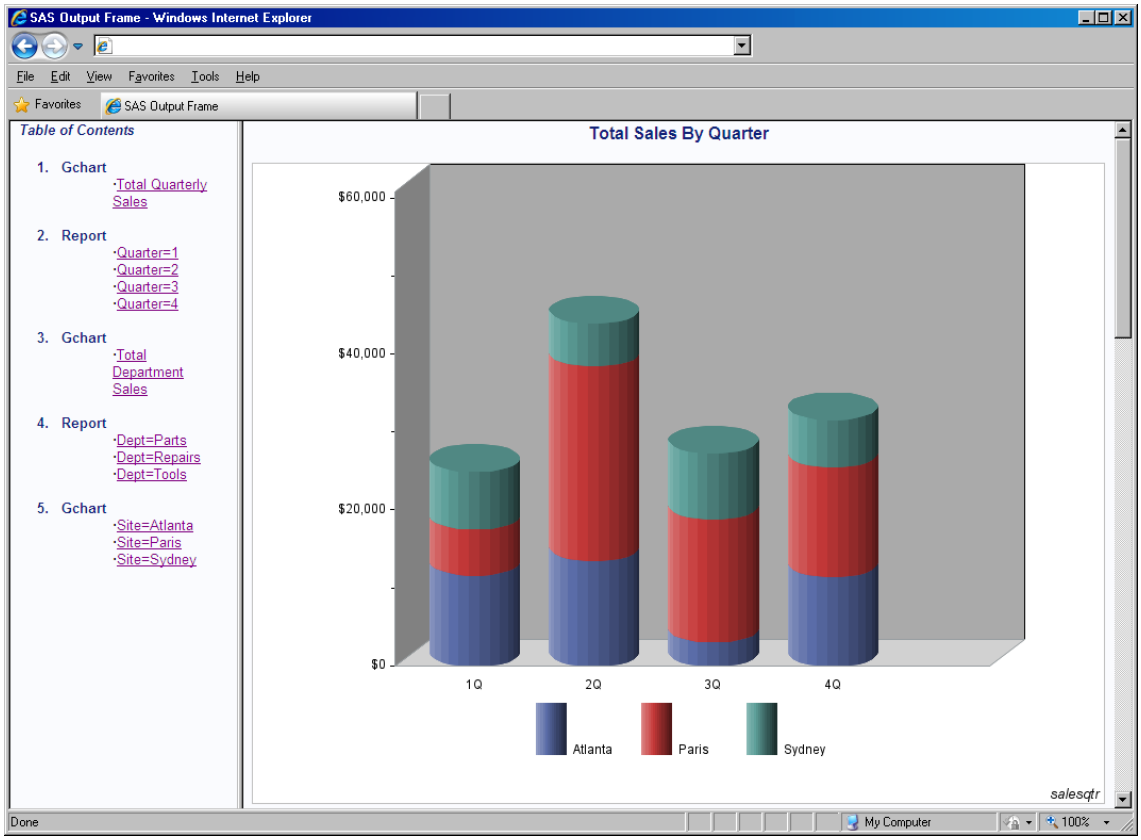
This example generates several graphs of sales data that can be accessed from a single web page. The graphs are two bar charts of summary sales data and three pie charts that break the data down by site. Each bar chart and an accompanying report is stored in a separate body file.

The three pie charts are generated with BY-group processing and are stored in one body file. The program suppresses the default BY lines and instead includes the BY variable value in the title for each chart. The SAS/GRAPH titles are displayed in the HTML output instead of in the graphics output.

Output

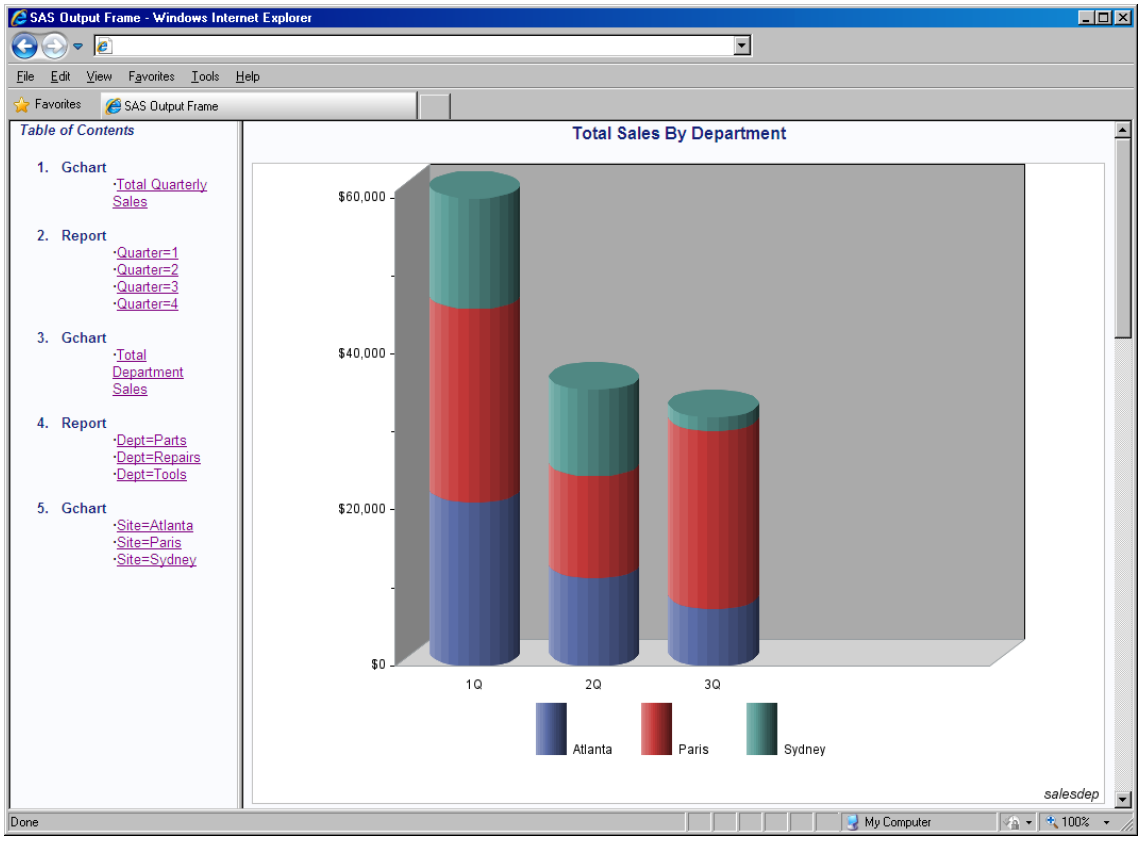
The web page contains two frames, one that displays a Table of Contents for all the graphs, and one that serves as the display area. Links to each piece of output appear in the table of contents, which is displayed in the left frame. Initially the frame file displays the first body file, which contains a bar chart and a report, as shown in the following figure.

Output 25.4 *Browser View of Bar Chart and Quarterly Sales Report*



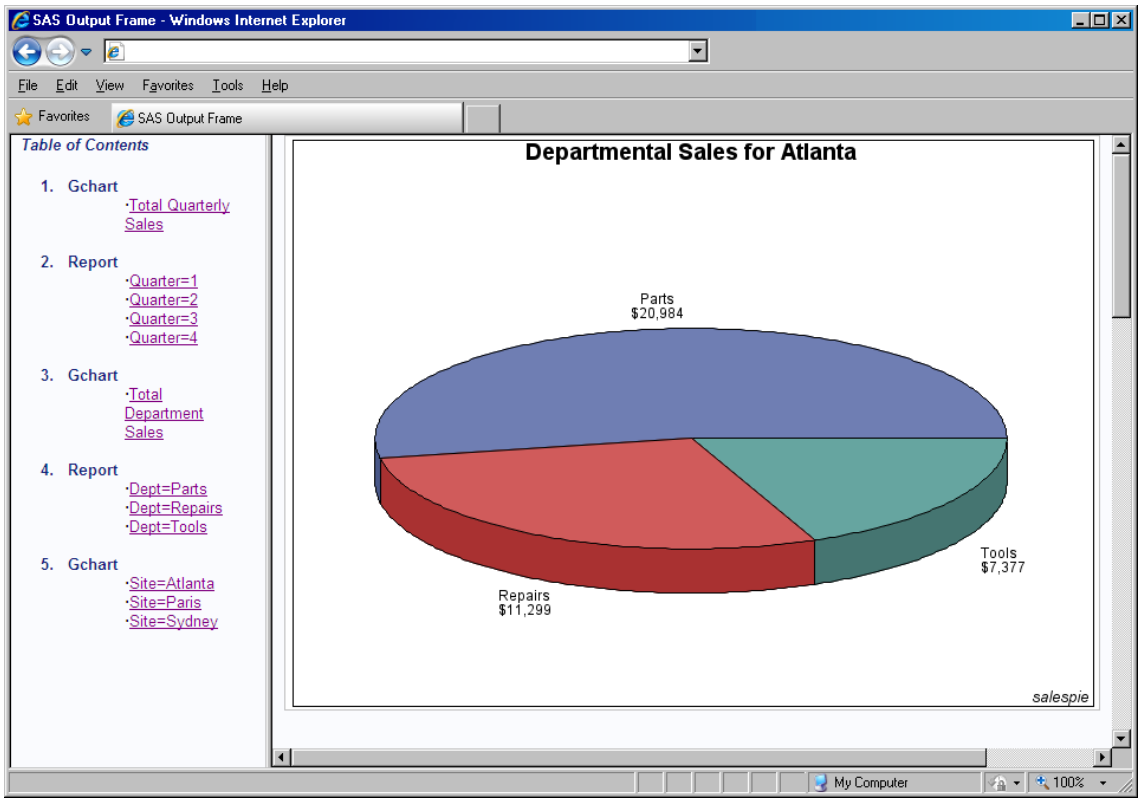
Notice that the chart title is displayed outside the graph as part of the HTML file. Select the link to **Total Department Sales** to display the second bar chart, as shown in the following figure.

Output 25.5 *Browser View of Bar Chart and Department Sales Report*



Selecting any link for **Department Sales** displays the corresponding pie chart as shown in the following figure. Because the pie charts are stored in one file, you can easily see all three by scrolling through the file.

Output 25.6 *Browser View of Pie Charts of Site Sales*



Program

```
filename odsout ".";

goptions reset=all border;

data totals;
   length Dept $ 7 Site $ 8;
   input Dept Site Quarter Sales;
   datalines;

Repairs Sydney  1 5592.82
Repairs Atlanta 1 9210.21
Tools   Sydney  1 1775.74
Tools   Atlanta 1 2424.19
Tools   Paris   1 5914.25
Parts   Atlanta 2 11595.07
Parts   Paris   2 9558.29
Repairs Sydney  2 5505.31
Repairs Paris   2 7538.56
Tools   Atlanta 2 1903.99
Tools   Paris   2 7868.34
Parts   Sydney  3 8437.96
Parts   Paris   3 6789.85
Tools   Atlanta 3 3048.52
Tools   Paris   3 9017.96
Parts   Sydney  4 6065.57
```

```
Parts   Atlanta 4 9388.51
Parts   Paris   4 8509.08
Repairs Atlanta 4 2088.30
Repairs Paris   4 5530.37
;

ods _all_ close;

ods html path=odsout frame="sales_frame.html"
   contents="sales_contents.html"
   body="sales_body1.html"
    nogtitle;

title1 "Total Sales By Quarter";
footnote j=r  "salesqtr ";

axis1 order=(0 to 60000 by 20000)
      minor=(number=1)
      label=none;
axis2 label=none length=70pct
      value=("1Q" "2Q" "3Q" "4Q");

legend1 label=none shape=bar(4,4);

proc gchart data=totals;
   format sales dollar8.;
   vbar3d quarter / discrete
                    sumvar=sales
                    shape=cylinder
                    subgroup=site
                    cframe=grayaa
                    caxis=black
                    width=12
                    space=4
                    legend=legend1
                    maxis=axis2
                    raxis=axis1
                    des="Total Quarterly Sales"
                    name="salesqtr";
run;
quit;

proc sort data=totals out=qtrsort;
   by quarter site;
run;

footnote1;
options nobyline;

title1 "Sales for Quarter #byval(quarter)";
proc report data=qtrsort nowindows;
  by quarter;
  column quarter site dept sales;
  define quarter / noprint group;
  define site    / display group;
  define dept    / display group;
  define sales   / display sum format=dollar8.;
  compute after quarter;
          site="Total";
  endcomp;
  break after site    / summarize style=rowheader;
```

```
  break after quarter / summarize style=rowheader;
run;

ods html body="sales_body2.html";

title1 "Total Sales By Department";
footnote1 j=r  "salesdep ";

axis1 label=none
      minor=(number=1)
      order=(0 to 100000 by 20000);
axis2 label=none length=70pct;

proc gchart data=totals;
   format sales dollar8.;
   vbar3d dept / shape=cylinder
                 subgroup=site
                 cframe=grayaa
                 width=12
                 space=4
                 sumvar=sales
                 legend=legend1
                 maxis=axis2
                 raxis=axis1
                 caxis=black
                 des="Total Department Sales"
                 name="salesdep";
run;
quit;

proc sort data=totals out=deptsort;
   by dept site;
run;

footnote1;
title1 "Sales for #byval(dept)";
proc report data=deptsort nowindows;
  by dept;
  column dept site quarter sales;
  define dept    / noprint group;
  define site    / display group;
  define quarter / display group;
  define sales   / display sum format=dollar8.;
  compute after dept;
          site="Total";
  endcomp;
  break after site / summarize style=rowheader;
  break after dept / summarize style=rowheader;
run;

ods html body="sales_body3.html"  gtitle;

proc sort data=totals out=sitesort;
   by site;
run;

title1 "Departmental Sales for #byval(site)";
footnote j=r "salespie ";

proc gchart data=sitesort;
     format sales dollar8.;
```

```
      by site;
      pie3d dept / noheading
                   coutline=black
                   sumvar=sales
                   des="Department Sales"
                   name="salespie";
run;
quit;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Create a fileref for the output. The current working directory is specified in this example.

```
filename odsout ".";
```

By default the ODS HTML statement to generate the graphics output as PNG files. The BORDER option is used so that the border around the graphics output area is compatible with the borders that are created for non-graphics output. Set the graphics environment.

```
goptions reset=all border;
```

Create the data set TOTALS. The data set contains quarterly sales data for three manufacturing sites for one year.

```
data totals;
   length Dept $ 7 Site $ 8;
   input Dept Site Quarter Sales;
   datalines;

Repairs Sydney  1 5592.82
Repairs Atlanta 1 9210.21
Tools   Sydney  1 1775.74
Tools   Atlanta 1 2424.19
Tools   Paris   1 5914.25
Parts   Atlanta 2 11595.07
Parts   Paris   2 9558.29
Repairs Sydney  2 5505.31
Repairs Paris   2 7538.56
Tools   Atlanta 2 1903.99
Tools   Paris   2 7868.34
Parts   Sydney  3 8437.96
Parts   Paris   3 6789.85
Tools   Atlanta 3 3048.52
Tools   Paris   3 9017.96
Parts   Sydney  4 6065.57
Parts   Atlanta 4 9388.51
Parts   Paris   4 8509.08
Repairs Atlanta 4 2088.30
Repairs Paris   4 5530.37
;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Set the ODS HTML destination for file handling options. The PATH= option specifies the fileref defined previously for the output. The FRAME= option names the HTML file that integrates the contents and body files. The CONTENTS= option names the HTML file that contains the table of contents to the HTML procedure output. The BODY= option names the file for storing the HTML output. The contents file links to each of the body files written to the HTML destination. The NOGTITLE option suppresses the graphics titles from the SAS/GRAPH output and displays them through the HTML page.

```
ods html path=odsout frame="sales_frame.html"
   contents="sales_contents.html"
   body="sales_body1.html"
    nogtitle;
```

Define title and footnote.

```
title1 "Total Sales By Quarter";
footnote j=r  "salesqtr ";
```

Define axis characteristics for the first bar chart. In AXIS2, the LENGTH= option specifies the length of the midpoint axis.

```
axis1 order=(0 to 60000 by 20000)
      minor=(number=1)
      label=none;
axis2 label=none length=70pct
      value=("1Q" "2Q" "3Q" "4Q");
```

Suppress the legend label and define the size of the legend values.

```
legend1 label=none shape=bar(4,4);
```

Generate the vertical bar chart of quarterly sales. The NAME= option specifies the name of the catalog entry.

```
proc gchart data=totals;
   format sales dollar8.;
   vbar3d quarter / discrete
                    sumvar=sales
                    shape=cylinder
                    subgroup=site
                    cframe=grayaa
                    caxis=black
                    width=12
                    space=4
                    legend=legend1
                    maxis=axis2
                    raxis=axis1
                    des="Total Quarterly Sales"
                    name="salesqtr";
run;
quit;
```

Sort the data set for the report of quarterly sales. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

```
proc sort data=totals out=qtrsort;
   by quarter site;
run;
```

Reset the footnote and suppress the BY line. We suppress the BY line because otherwise #BYVAL inserts the value of the BY variable into the title of each report.

```
footnote1;
options nobyline;
```

Generate a report of quarterly sales. Because the HTML body file that references the GCHART procedure output is still open, the report is stored in that file. The chart and report are shown in Output 25.7 on page 559.

```
title1 "Sales for Quarter #byval(quarter)";
proc report data=qtrsort nowindows;
  by quarter;
  column quarter site dept sales;
  define quarter / noprint group;
  define site    / display group;
  define dept    / display group;
  define sales   / display sum format=dollar8.;
  compute after quarter;
          site="Total";
  endcomp;
  break after site    / summarize style=rowheader;
  break after quarter / summarize style=rowheader;
run;
```

Assign a new body file for the second bar chart and report. Assigning a new body file closes SALES_BODY1.HTML. The contents and frame files, which remain open, contains links to all body files.

```
ods html body="sales_body2.html";
```

Define title and footnote for second bar chart.

```
title1 "Total Sales By Department";
footnote1 j=r  "salesdep ";
```

Define axis characteristics. These axis statements replace the ones defined earlier. As before, the LENGTH= option defines the length of the midpoint axis.

```
axis1 label=none
      minor=(number=1)
      order=(0 to 100000 by 20000);
axis2 label=none length=70pct;
```

Generate the vertical bar chart of departmental sales.

```
proc gchart data=totals;
   format sales dollar8.;
   vbar3d dept / shape=cylinder
                 subgroup=site
                 cframe=grayaa
                 width=12
                 space=4
                 sumvar=sales
                 legend=legend1
                 maxis=axis2
                 raxis=axis1
                 caxis=black
                 des="Total Department Sales"
                 name="salesdep";
run;
quit;
```

Sort the data set for the report of department sales. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

```
proc sort data=totals out=deptsort;
   by dept site;
run;
```

Reset the footnote, define a report title, and generate the report of department sales. #BYVAL inserts the value of the BY variable into the title of each report. The chart and report are shown in Output 25.7 on page 559.

```
footnote1;
title1 "Sales for #byval(dept)";
proc report data=deptsort nowindows;
  by dept;
  column dept site quarter sales;
  define dept    / noprint group;
  define site    / display group;
  define quarter / display group;
  define sales   / display sum format=dollar8.;
  compute after dept;
          site="Total";
  endcomp;
  break after site / summarize style=rowheader;
  break after dept / summarize style=rowheader;
run;
```

Assign a new body file for the pie charts. Assigning a new file as the body file closes SALES_BODY2.HTML. The contents and frame files remain open. GTITLE displays the titles in the graph.

```
ods html body="sales_body3.html"  gtitle;
```

Sort data set in order of the BY variable before running the GCHART procedure with BY-group processing.

```
proc sort data=totals out=sitesort;
   by site;
run;
```

Define title and footnote. #BYVAL inserts the value of the BY variable SITE into the title for each output.

```
title1 "Departmental Sales for #byval(site)";
footnote j=r "salespie ";
```

Generate a pie chart for each site. All the procedure output is stored in one body file. Because BY-group processing generates multiple graphs from one PIE3D statement, the name assigned by the NAME= option is incremented to provide a unique name for each piece of output.

```
proc gchart data=sitesort;
     format sales dollar8.;
     by site;
     pie3d dept / noheading
                  coutline=black
                  sumvar=sales
                  des="Department Sales"
                  name="salespie";
run;
quit;
```

Close the ODS HTML destination. If you do not close the destination, then you are not able to view the HTML file specified by the FRAME option until you close your SAS session.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```
ods html; /* Not required in SAS Studio */
```

Example 9: Creating a Web Page with Drill-Down Functionality Using the ODS HTML Statement

Features:	GOPTIONS statement options RESET= TRANSPARENCY= DEVICE= ODS HTML statement options BODY= NOGTITLE PATH=
Data set:	TOTALS
Sample library member:	GONDDCW1

Details

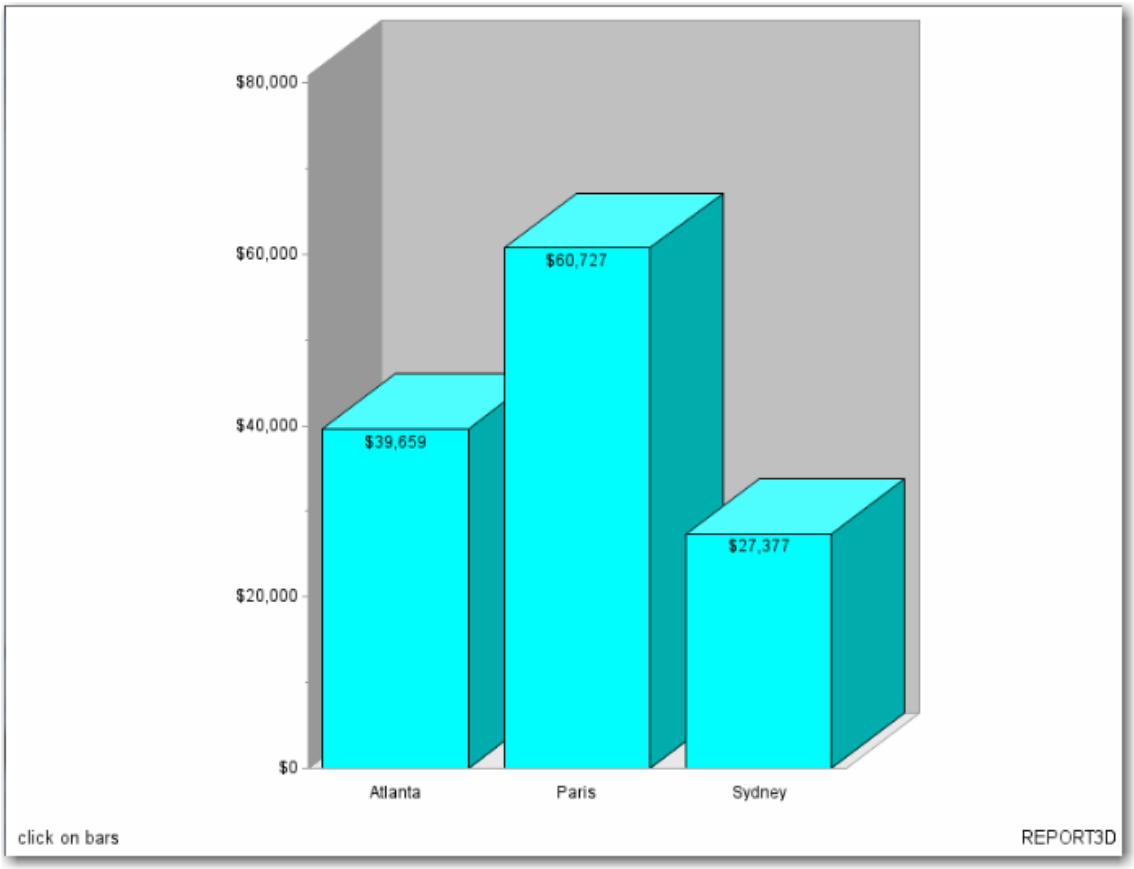
The code for this example is in SAS Sample Library member GONDDCW1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example shows you how to create a drill-down graph in which the user can select an area of the graph in order to display additional information about the data. The program creates one vertical bar chart of total sales for each site and three reports that break down the sales figures for each site by department and quarter.

Output

The following figure shows the bar chart of sales.

Output 25.7 *Vertical Bar Chart of Total Sales*



Output 25.11 on page 569 shows the PROC REPORT output that appears when you click on the bar for Atlanta.

Output 25.8 *PROC REPORT Output Displayed in a Web Browser*

Sales Report for Atlanta

Site=Atlanta

Dept	Quarter	Sales
Parts	2	$11,595
	4	$9,389
Repairs	1	$9,210
	4	$2,088
Tools	1	$2,424
	2	$1,904
	3	$3,049
Total		**$39,659**

Program

```
filename odsout ".";

goptions reset=all device=gif transparency noborder;

data totals;
   length Dept $ 7 Site $ 8;
   input Dept Site Quarter Sales;
   datalines;

Repairs Sydney  1 5592.82
Repairs Atlanta 1 9210.21
Tools   Sydney  1 1775.74
Tools   Atlanta 1 2424.19
Tools   Paris   1 5914.25
Parts   Atlanta 2 11595.07
Parts   Paris   2 9558.29
Repairs Sydney  2 5505.31
Repairs Paris   2 7538.56
Tools   Atlanta 2 1903.99
Tools   Paris   2 7868.34
Parts   Sydney  3 8437.96
Parts   Paris   3 6789.85
Tools   Atlanta 3 3048.52
Tools   Paris   3 9017.96
Parts   Sydney  4 6065.57
Parts   Atlanta 4 9388.51
Parts   Paris   4 8509.08
Repairs Atlanta 4 2088.30
Repairs Paris   4 5530.37
;

data newtotal;
   set totals;
   length sitedrill $40;
if site="Atlanta" then
   sitedrill="HREF='report_deptsales.html#IDX'";

else if site="Paris" then
   sitedrill="HREF='report_deptsales.html#IDX1'";

if site="Sydney" then
   sitedrill="HREF='report_deptsales.html#IDX2'";
run;

ods _all_ close;

ods html
    path=odsout body="report_body.html"
    nogtitle;

title1 "Total Sales for All Sites";
footnote1  j=l "click on bars" j=r "REPORT3D ";

pattern color=cyan;

axis1 order=(0 to 80000 by 20000)
      minor=(number=1)
```

```
      label=none;
axis2 label=none offset=(9,9);

proc gchart data=newtotal;
   format sales dollar8.;
   vbar3d site / discrete
                 width=15
                 sumvar=sales
                 inside=sum
                 html=sitedrill
                 coutline=black
                 cframe=ltgray
                 maxis=axis2
                 raxis=axis1
                 name="report3d ";
run;
quit;

proc sort data=newtotal;
   by site dept quarter;
run;

ods html body="report_deptsales.html" ;

goptions reset=footnote;

options nobyline;
title1 "Sales Report for #byval(site)";

proc report data=newtotal nowindows;
  by site;
  column site dept quarter sales;
  define site    / noprint group;
  define dept    / display group;
  define quarter / display group;
  define sales   / display sum format=dollar8.;
  compute after site;
          dept="Total";
  endcomp;
  break after site / summarize style=rowheader page;
run;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Create a fileref for the output. The current working directory is specified in this example.

```
filename odsout ".";
```

Set the graphics environment. In the GOPTIONS statement, DEVICE=GIF causes the ODS HTML statement to generate the graphics output as GIF files. The TRANSPARENCY option causes the graphics output to use the web-page background as the background of the graph.

```
goptions reset=all device=gif transparency noborder;
```

Create the data set TOTALS. The data set contains quarterly sales data for three manufacturing sites for one year.

```
data totals;
   length Dept $ 7 Site $ 8;
   input Dept Site Quarter Sales;
   datalines;

Repairs Sydney  1 5592.82
Repairs Atlanta 1 9210.21
Tools   Sydney  1 1775.74
Tools   Atlanta 1 2424.19
Tools   Paris   1 5914.25
Parts   Atlanta 2 11595.07
Parts   Paris   2 9558.29
Repairs Sydney  2 5505.31
Repairs Paris   2 7538.56
Tools   Atlanta 2 1903.99
Tools   Paris   2 7868.34
Parts   Sydney  3 8437.96
Parts   Paris   3 6789.85
Tools   Atlanta 3 3048.52
Tools   Paris   3 9017.96
Parts   Sydney  4 6065.57
Parts   Atlanta 4 9388.51
Parts   Paris   4 8509.08
Repairs Atlanta 4 2088.30
Repairs Paris   4 5530.37
;
```

Add the HTML variable to the TOTALS data set and create the NEWTOTAL data set. The HTML variable SITEDRILL contains the targets for the values of the variable SITE. Each HREF value specifies the HTML body file and the name of the anchor within the body file that identifies the target graph.

```
data newtotal;
   set totals;
   length sitedrill $40;
if site="Atlanta" then
   sitedrill="HREF='report_deptsales.html#IDX'";

else if site="Paris" then
   sitedrill="HREF='report_deptsales.html#IDX1'";

if site="Sydney" then
   sitedrill="HREF='report_deptsales.html#IDX2'";
run;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Set the ODS HTML destination for file handling options. The PATH= option specifies the fileref that was defined previously for the output. The BODY= option names the file for storing HTML output. The NOGTITLE option suppresses the graph titles from the SAS/GRAPH output and displays them in the HTML.

```
ods html
    path=odsout body="report_body.html"
    nogtitle;
```

Define title and footnote.

```
title1 "Total Sales for All Sites";
footnote1  j=l "click on bars" j=r "REPORT3D ";
```

Assign a pattern color for the bars. Each bar in the graph uses the same PATTERN definition.

```
pattern color=cyan;
```

Define axis characteristics. The VBAR3D statement to follow assigns axis1 to the response axis and axis2 to the midpoint axis.

```
axis1 order=(0 to 80000 by 20000)
      minor=(number=1)
      label=none;
axis2 label=none offset=(9,9);
```

Generate the vertical bar chart of total sales for each site. The HTML= option specifies SITEDRILL as the variable that contains the name of the target. Specifying the HTML= option causes SAS/GRAPH to add an image map to the HTML body file. The NAME= option specifies the name of the catalog entry.

```
proc gchart data=newtotal;
   format sales dollar8.;
   vbar3d site / discrete
                 width=15
                 sumvar=sales
                 inside=sum
                 html=sitedrill
                 coutline=black
                 cframe=ltgray
                 maxis=axis2
                 raxis=axis1
                 name="report3d ";
run;
quit;
```

Sort the data set NEWTOTAL. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

```
proc sort data=newtotal;
   by site dept quarter;
run;
```

Open the file for the PROC REPORT output. Assigning a new body file closes REPORT_BODY.HTML.

```
ods html body="report_deptsales.html" ;
```

Clear the footnote.

```
goptions reset=footnote;
```

Suppress the default BY line and define a title that includes the BY-value. #BYVAL inserts the value of the BY variable SITE into the title of each report.

```
options nobyline;
title1 "Sales Report for #byval(site)";
```

Print a report of departmental sales for each site.

```
proc report data=newtotal nowindows;
  by site;
  column site dept quarter sales;
```

```
    define site    / noprint group;
    define dept    / display group;
    define quarter / display group;
    define sales   / display sum format=dollar8.;
    compute after site;
            dept="Total";
    endcomp;
    break after site / summarize style=rowheader page;
  run;
```

Close ODS HTML.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```
ods html; /* Not required in SAS Studio */
```

Details

Putting It All Together

This section provides additional information about the pieces of this program and how they work together to generate SAS/GRAPH output with drill-down functionality. It describes

- how an HREF value is built
- how the HTML= option creates an image map in the HTML file
- how the HTML file references the SAS/GRAPH output.

Building an HREF Value

In the DATA step, the variable SITEDRILL is assigned a string that defines the link target for a data value. An example follows.

```
if site="Atlanta" then
   sitedrill="HREF='report_deptsales.html#IDX1'";
```

The link target is specified by the HTML HREF attribute. The HREF value tells the web page where to link to when a user selects the region associated with the value `Atlanta`.

For example, clicking on the first bar in the chart links to the target defined by `report_deptsales.html#IDX1`. This target consists of a filename and an anchor. The file, `report_deptsales.html`, is generated by the PROC REPORT step. `IDX1` is the anchor that identifies the section of the file that contains the report for the first BY group, `Atlanta`.

Because anchor names increment, in order to assign them accurately you must know how many pieces of output your program generates and in what order. For example, this table lists in order the pieces of output generated by this example and their default anchor names:

Procedure	Output	Anchor name
GCHART	report3d.gif	IDX
REPORT	Atlanta report	IDX1
REPORT	Paris report	IDX2
REPORT	Sydney report	IDX3

Creating an Image Map

The HTML= option in the GCHART procedure is assigned the variable with the target information – in this case, SITEDRILL.

```
html=sitedrill
```

This option causes SAS/GRAPH to generate (in the HTML body file) the MAP and AREA elements that compose the image map. It loads the HREF attribute value from SITEDRILL into the AREA element. This image map is named `gqcke00k_map`. The map is stored in `report_body.html`. (ODS generates unique map names each time you run the program, so the next time this program runs, the map name will be different):

```
<MAP NAME="gqcke00k_map">
   <AREA SHAPE="POLY"
       HREF="report_deptsales.html#IDX3"
       COORDS="423,409,423,242,510,242,510,409" >
   <AREA SHAPE="POLY"
       HREF="report_deptsales.html#IDX2"
       COORDS="314,409,314,139,401,139,401,409" >
   <AREA SHAPE="POLY"
       HREF="report_deptsales.html#IDX1"
       COORDS="205,409,205,199,292,199,292,409" >
</MAP>
```

The AREA element defines the regions within the graph that you can select to link to other locations. It includes attributes that define the shape of the region (the SHAPE= option) and position of the region (the COORDS= option) as well as the link target (the HREF= option).

The value assigned to the HREF= attribute is contained in the variable assigned to the HTML= option, in this case SITEDRILL.

Referencing SAS/GRAPH Output

In the GOPTIONS statement, DEVICE=GIF causes SAS/GRAPH to create GIF files from the SAS/GRAPH output. It also adds to the open body file an IMG element that points to the GIF file. In this case, SAS/GRAPH adds the following IMG element to `report_body.html`:

```
<IMG SRC="report3d.gif" USEMAP="#gqcke00k_map">
```

The IMG element tells the web page to get the image from the file `report3d.gif`. It also tells the web page to use the image map `#report3d_map` to define the hotspots of the bar chart.

26

Graphics Options and Device Parameters Dictionary

Introduction to Graphics Options and Device Parameters

This chapter provides a detailed description of all of the graphics options and device parameters used with SAS/GRAPH software. These include

- all graphics options used by the GOPTIONS statement
- all device parameters that can be specified as options in the ADD and MODIFY statements in the GDEVICE procedure
- all device parameters that appear as fields in the GDEVICE windows.

The descriptions provide the syntax, defaults, and required information for each option and parameter.

The graphics options and device parameters are intermixed and listed alphabetically. When the graphics option and device parameter have the same name, they are discussed in the same dictionary entry and the description uses only that name and does not distinguish between the option and the parameter except where the distinction is necessary.

For a list of all the graphics options, see “GOPTIONS Statement” on page 430. For a list of all the device parameters, see “ADD Statement” on page 1167.

If the syntax for the graphics option and the device parameter is different, both forms are shown. If the syntax is the same, one form is shown.

Specifying Graphics Options and Device Parameters

Using the GOPTIONS Statement and the GDEVICE Procedure

Use a GOPTIONS statement to specify the graphics options. Some graphics options can also be specified in an OPTIONS statement. Use the GDEVICE procedure to specify the device parameters. (See "GOPTIONS Statement" on page 430 and Chapter 38, "GDEVICE Procedure," on page 1163 for details.)

Note: The syntax for device parameters is the syntax for specifying parameters when using the GDEVICE procedure statements. With the GDEVICE windows, simply enter values into fields in the windows.

Note: The values that you specify for any option or parameter must be valid for the device. If you specify a value that exceeds the device's capabilities, SAS/GRAPH software reverts to values that can be used with the device.

Specifying Units of Measurement

When the syntax of an option includes *units*, use one of these unless the syntax specifies otherwise:

CELLS	character cells
CM	centimeters
IN	inches
PCT	percentage of the graphics output area
PT	points (there are approximately 72 points in an inch)

If you omit *units*, a unit specification is searched for in this order:

1. the value of GUNIT= in a GOPTIONS statement
2. the default unit, CELLS

Note: The IN and CM units of measurement are not supported by Java.

Options Used in Both GOPTIONS and OPTIONS Statements

Several of the graphics options are also used as system options. These options include the following:

- these printer options:

BINDING	PAPERSIZE
COLLATE	PAPERSOURCE
DUPLEX	PAPERTYPE
PAPERDEST	

- the DEVICE option

With the exception of the DEVICE= option, the effect of setting these options depends on whether they are set as graphics options or system options. For the printer options, when they are set as graphics options in a GOPTIONS statement, they affect the SAS/GRAPH devices. When they are set as system options in an OPTIONS statement, they affect the Universal Printers and the Universal Printer shortcut devices. For information about the Universal Printers, Universal Printer shortcut devices, and the SAS/GRAPH devices, see "Using Graphics Devices" on page 89. Setting a printer graphics option does not affect the corresponding system option, and vice versa. For example, here is a statement that sets the graphics option PAPERSIZE= to A4.

```
goptions papersize=A4;
```

This statement does not affect the PAPERSIZE= system option. As a result, it sets the paper size to A4 for the SAS/GRAPH printer devices only. If you want to set the paper size for the Universal Printer devices or the Universal Printer shortcut devices, you must set the PAPERSIZE= system option instead of the graphics option as shown in the following example.

```
options papersize=A4;
```

For information about the GOPTIONS statement, see "GOPTIONS Statement" on page 430. For information about the OPTIONS statement, see *SAS DATA Step Statements: Reference*.

In contrast to the printer options listed previously, setting the DEVICE= option as a graphics option has the same effect as setting it as a system option. That is, setting DEVICE= as either a graphics option or a system option sets the SAS/GRAPH output device driver. As a result, the following two statements have the same effect:

```
options device=pdf;
goptions device=pdf;
```

Note: Only one of the two statements is needed to set the device. You can use either one.

Dictionary

ACCESSIBLE, NOACCESSIBLE Graphics Options

Specifies whether descriptive text and summary statistics representing your graphics output are written to the graph's HTML output file.

Used by: GOPTIONS statement

Default: NOACCESSIBLE

Restriction: Supported only by ACTIVEX when used with the ODS HTML output destination.

Syntax

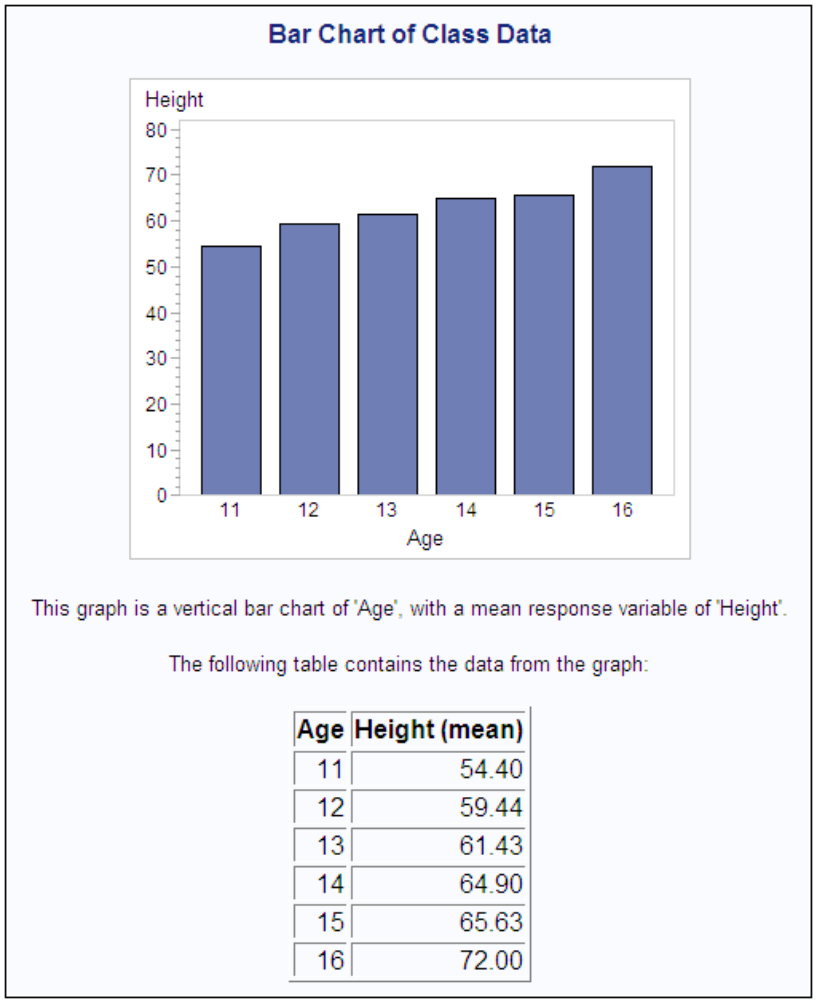
ACCESSIBLE | NOACCESSIBLE

Parameter Values

ACCESSIBLE

when used with an HTML ODS destination, generates descriptive text and data for your graphs to comply with section 508 of the Rehabilitation Acts and to meet usability requirements for disabled users. SAS/GRAPH writes accessibility information to the graph's output HTML file, and creates a left-justified footnote that provides a link to the information. The information and the link are not visible in the output HTML. However, both are detected by accessibility aids, such as screen readers. You can also access the information by pressing the tab key and enter. The information will be displayed once you press enter on the link in the footnote. The information will also be displayed if you move your mouse over the location of the left–justified footnote, and click the link when the mouse pointer shape changes.

Figure 26.1 ACCESSIBLE

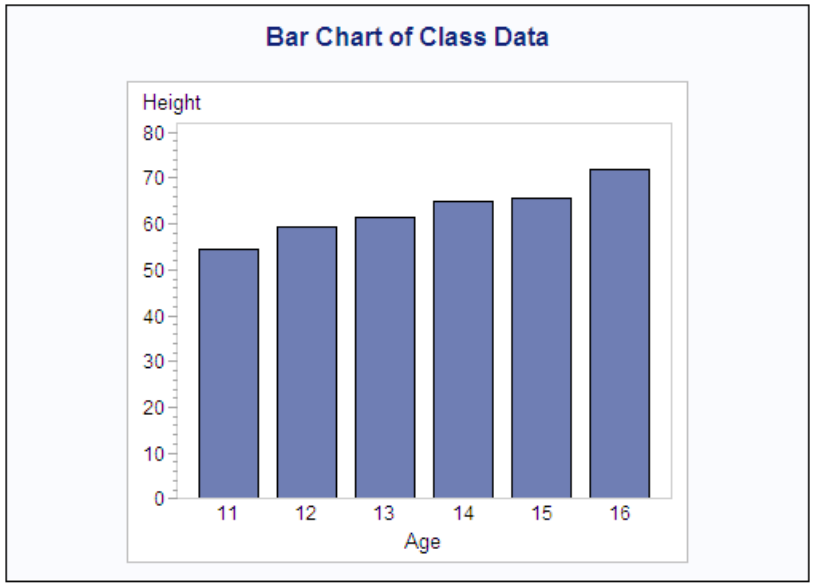


Age	Height (mean)
11	54.40
12	59.44
13	61.43
14	64.90
15	65.63
16	72.00

NOACCESSIBLE

toggles off the ACCESSIBLE option.

Figure 26.2 NOACCESSIBLE



ALTDESC, NOALTDESC Graphics Options

Specifies whether to write the DESCRIPTION= statement text to the ALT= text in an HTML file.

Used by: GOPTIONS statement

Default: ALTDESC

Restriction: Supported only when used with the HTML output destination and the DESCRIPTION= option.

Syntax

ALTDESC | NOALTDESC

Parameter Values

ALTDESC

when used with an HTML ODS destination, includes an HTML ALT tag in the HTML output that specifies alternate text for the graph. By default the entire output has an HTML ALT tag that specifies which procedure was used, and which variables were plotted. Or, if you have specified text using the DESCRIPTION= option, then that value is used for the HTML ALT tag rather than the default ALT tag (many users add a textual description of the graph using this technique, to help the vision-impaired, and to help meet 508-compliance).

If you prefer not to have an ALT tag for the entire graph, you can suppress it by specifying DESCRIPTION=" " (which might be more convenient on a graph by graph basis) or by using GOPTIONS NOALTDESC (which might be more

convenient for turning them off for all graphs, such as putting this in your AUTOEXEC.BAT).

NOALTDESC
toggles off the ALTDESC option.

ASPECT Device Parameter and Graphics Option

Sets the aspect ratio for graphics elements.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

ASPECT=*scaling-factor*

Parameter Values

scaling-factor
specifies a nonnegative integer or real number that determines the ratio of width to height for graphics elements. If you specify ASPECT=1, each graphics element has equal horizontal and vertical scaling factors; ASPECT=2 scales the graphics element twice as wide as its height; and so on. If ASPECT= is not specified or is set to 0 or null, SAS/GRAPH uses the aspect ratio of the hardware device.

Details

The aspect ratio affects many graphics characteristics, such as the shape of software characters and the roundness of pie charts. Some graphics drivers do not produce correct output if the aspect ratio is anything other than the default. When you use a device that uses local scaling (that is, the device itself can scale the output—for example, some plotters), use ASPECT= to tell SAS/GRAPH the scaling factor.

Note: You can get more reliable results if you use the default aspect ratio and use the HSIZE= and VSIZE= graphics options to set the dimensions.

AUTOCOPY, NOAUTOCOPY Device Parameters and Graphics Options

Specifies whether to generate hard copy automatically.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Defaults: GOPTIONS: NOAUTOCOPY
GDEVICE: AUTOCOPY=N

Restrictions: This option is device-dependent.
This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

AUTOCOPY | NOAUTOCOPY

Form 2: GDEVICE:

AUTOCOPY=Y | N

Parameter Values

AUTOCOPY
AUTOCOPY=Y
prints a copy of the graph automatically.

NOAUTOCOPY
AUTOCOPY=N
suppresses printing a copy of the graph. A blank `Autocopy` field in the Parameters window is the same as AUTOCOPY=N.

Details

AUTOCOPY is used only for older terminals with printers attached directly to the device.

AUTOFEED, NOAUTOFEED Device Parameters and Graphics Options

Specifies whether devices with continuous paper or automatic paper feed should roll or feed the paper automatically for the next graph.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Defaults: GOPTIONS: AUTOFEED (if a device is specified)
GDEVICE: AUTOFEED=Y

Restrictions: This option is device-dependent.
This option is not supported by Java or ActiveX.

See: PPDFILE on page 677

Syntax

Form 1: GOPTIONS:

AUTOFEED | NOAUTOFEED

Form 2: GDEVICE:

AUTOFEED=Y | N

Parameter Values

AUTOFEED
AUTOFEED=Y
causes the device to feed new paper automatically for the next graph. A blank `Autofeed` field in the Parameters window is the same as AUTOFEED=Y.

NOAUTOFEED
AUTOFEED=N
suppresses the automatic paper feed.

Details

For PostScript devices, if AUTOFEED is unaltered, the PostScript file is unchanged. If you specify NOAUTOFEED and do not select a PPD file with the PPDFILE option, a PostScript Level 1 MANUALFEED command is added to the driver output. If you specify NOAUTOFEED and select a PPD that contains a MANUALFEED option, the procedure code for that MANUALFEED option is sent. If there is no MANUALFEED option in the PPD, no MANUALFEED code is sent. See “PPDFILE Graphics Option” on page 677.

AUTOSIZE Graphics Option

Controls whether to change the size of the character cells in order to preserve the number of rows and columns specified in the device entry.

Used by: GOPTIONS statement

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

See: DEVOPTS on page 609

Syntax

AUTOSIZE=ON | OFF | DEFAULT

Parameter Values

ON
changes the cell size in order to preserve the number of rows and columns.

OFF
preserves the device's original cell size and temporarily changes the number of rows and columns.

DEFAULT
uses the default setting (ON or OFF) that is controlled by DEVOPTS bit 50. See "DEVOPTS Device Parameter" on page 609.

Details

AUTOSIZE is useful when you change the size of the graphics display area using one or more of the options PAPERSIZE, XPIXELS, YPIXELS, XMAX, or YMAX. It lets you control image text size without using PROC GDEVICE. Typically, AUTOSIZE is on for most image drivers and off for all other types of drivers.

Note: If you use HSIZE of VSIZE, the character cell size changes regardless of the AUTOSIZE setting.

BINDING Graphics Option

Specifies which edge of the document is the binding edge.

Used by: GOPTIONS statement, OPTIONS statement

Default: DEFAULTEDGE

Restrictions: PostScript and PCL printers only. PostScript printers require a PPD file.
Not supported by Java or ActiveX.

See: DUPLEX on page 617
PPDFILE on page 677
"Options Used in Both GOPTIONS and OPTIONS Statements" on page 582

Syntax

BINDING=DEFAULTEDGE | LONGEDGE | SHORTEDGE

Comparisons

BINDING controls how the page is flipped when DUPLEX is in effect. It does not change the orientation of the graph. DEFAULTEDGE refers to the hardware's factory-default setting. LONGEDGE and SHORTEDGE refer to the paper's long and short edges.

For PostScript printers, a PPD file must also be specified, using the PPDFILE= option. The PPD file contains the command that SAS/GRAPH needs to request the appropriate binding method on the printer being used. If a PPD file is not specified, the BINDING= option is ignored because SAS/GRAPH will lack the command needed to request the binding method.

BORDER, NOBORDER Graphics Options

Specifies whether to draw a border around the graphics output area.

Used by: GOPTIONS statement

Default: NOBORDER

Restriction: This option is ignored when the JAVAIMG, ACTIVEX, or ACTXIMG device is used.

Syntax

BORDER | NOBORDER

Details

The placement of the border on the display is defined by the HSIZE= and VSIZE= graphics options, if used. Otherwise, the placement is defined by the XMAX and YMAX device parameters.

CBACK Device Parameter and Graphics Option

Specifies the background color of the graphics output.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Gcolors window

Default: as specified in the Gcolors window

Syntax

CBACK=*background-color*

Parameter Values

background-color
specifies the background color using any SAS/GRAPH color name. See Chapter 23, “Using Colors in SAS/GRAPH Programs,” on page 363 for information about specifying colors.

Details

The CBACK= option is valid on all devices but can be ignored by some (for example, plotters). Specify the default in the Gcolors window of the device entry.

Note: This option overrides the Background and Foreground style attributes in the graph styles. For more information about graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: Procedures Guide*.

If you explicitly specify a background color with the CBACK= option, the background color that you select should contrast with the foreground colors.

If the IBACK= option is in effect, an image will appear in the background in place of the color specified with the CBACK= option.

CBY Graphics Option

Selects the color of the BY lines that appear in the graphics output.

Used by:	GOPTIONS statement
Defaults:	(1) CTEXT= graphics option, if used (2) first color in current color list
Restriction:	This option is not supported by Java or ActiveX.

Syntax

CBY=*By line-color*

Parameter Values

By line-color
specifies the By-line color using any SAS/GRAPH color name. See Chapter 23, “Using Colors in SAS/GRAPH Programs,” on page 363 for information about specifying colors.

Details

When you use a BY statement with a SAS/GRAPH procedure to process a data set in subgroups, each graph produced by that procedure is headed by a BY line that displays the BY variables and their values that define the current subgroup.

CELL, NOCELL Device Parameters and Graphics Options

Controls whether to use cell alignment.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

CELL | NOCELL

Form 2: GDEVICE:

CELL=Y | N

Parameter Values

CELL
CELL=Y
causes the device to use cell alignment. In that case, SAS/GRAPH attempts to place hardware (or simulated hardware) characters inside character cells. This restriction on the location of characters means that in some cases the SAS/GRAPH procedure can generate axes that do not occupy the entire procedure output area or might be unable to create the requested graph. A blank `Cell` field in the Parameters window is the same as CELL=Y.

NOCELL
CELL=N
suppresses cell alignment, causing the procedure to use the entire procedure output area and place axis and tick mark labels without regard to cell alignment.

Details

Specify N in the device entry or use NOCELL in a GOPTIONS statement if you want to preview a graph on a cell-aligned display but intend to produce the final graph on a device that is not cell-aligned, such as a pen plotter.

CHARACTERS, NOCHARACTERS Device Parameters and Graphics Options

Specifies whether the device-resident font is used when no font or FONT=NONE is specified in a SAS statement.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Defaults: GOPTIONS: CHARACTERS
GDEVICE: CHARACTERS=Y

Restriction: This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

CHARACTERS | NOCHARACTERS

Form 2: GDEVICE:

CHARACTERS=Y | N

Parameter Values

CHARACTERS
CHARACTERS=Y
causes SAS/GRAPH to use the device-resident font when you do not specify a font in a SAS program. A blank `Characters` field in the Parameters window is the same as CHARACTERS=Y.

NOCHARACTERS
CHARACTERS=N
causes SAS/GRAPH to draw the characters using the SIMULATE font and suppresses the use of *all* device—resident fonts, regardless of values that you specify in other SAS statements.

Details

The device—resident font is not used if you changed the HPOS= and VPOS= graphics options from the default, or if you used the HEIGHT= option in a SAS statement *and* the device does not have scalable characters.

CHARREC Device Parameter

Specifies a device-resident font by associating a CHARTYPE number with a device-resident font. Also defines a default size to use with that font.

Used by: GDEVICE procedure

Default: device-dependent

Syntax

```
CHARREC=(
    type, rows, cols, 'font', 'Y' | 'N'
      <type, rows, cols, 'font', 'Y' | 'N'>
      <…>
)
```

Parameter Values

Each list contains values that correspond to the fields in the Chartype window. *Charrec-list* in this form:

type
specifies the CHARTYPE number, which can be an integer in the range 0–9999. See "CHARTYPE Device Parameter and Graphics Option" on page 595 for more information.

rows
specifies the number of rows of text in the font that will fit on the display. See "ROWS Device Parameter" on page 694 for more information.

cols
specifies the number of columns of text in the font that will fit on the display. See "COLS Device Parameter" on page 600 for more information.

'font'
specifies a character string enclosed in quotation marks that contains the name of the corresponding device-resident font. See "FONT NAME Device Parameter" on page 625 for more information.

'Y' | 'N'
specifies whether the font is scalable.

'Y'
specifies a scalable font.

A scalable font can be displayed at any size. See "SCALABLE Device Parameter" on page 694 for more information.

'N'
specifies a nonscalable font.

A nonscalable font can be displayed only at a fixed size. See "SCALABLE Device Parameter" on page 694 for more information.

Example These values assign the device's Helvetica font to be the first device-resident font in the CHARTYPE window of the driver entry:

```
charrec=(1, 100, 75, 'helvetica', 'y')
```

CHARTYPE Device Parameter and Graphics Option

Selects the number of the default hardware character set.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

CHARTYPE=*hardware-font-chartype*

Parameter Values

hardware-font-chartype
specifies the hardware font character type as an integer in the range 0–999. *Hardware-font-chartype* refers to the actual number for the device-resident font that you want to use as listed in the Chartype window of the device entry for the selected device driver. By default, CHARTYPE is 0, which is the default device-resident font for the device.

CIRCLEARC, NOCIRCLEARC Device Parameters and Graphics Options

Specifies whether SAS/GRAPH should use the device's hardware circle-drawing capability, if available.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

CIRCLEARC | NOCIRCLEARC

Form 2: GDEVICE:

CIRCLEARC=Y | N

Parameter Values

CIRCLEARC
CIRCLEARC=Y
causes SAS/GRAPH to use the built-in hardware circle- and arc-drawing capability of the device. A blank `Circlearc` field in the Parameters window is the same as CIRCLEARC=Y.

Tip Hardware drawing is faster, but not all devices have the capability. SAS/GRAPH does not try to use the capability if your device does not support it.

NOCIRCLEARC
CIRCLEARC=N
causes SAS/GRAPH to use software move and draw commands to draw circles and arcs.

CMAP Device Parameter

Specifies a color map for the device.

Used by: GDEVICE procedure, GDEVICE Colormap window

Syntax

CMAP=('*from-color*:*to-color*' <,'*from-color-n*:*to-color-n*'> <…>)

Parameter Values

'from-color:to-color'
specifies a color map entry.

from-color
specifies the name that you want to assign to the color designated by the *color* value.

In the Colormap window, enter this value in the `From` field.

to-color
specifies any SAS/GRAPH color name up to eight characters long.

In the Colormap window, enter this value in the `To` field. See Chapter 23, “Using Colors in SAS/GRAPH Programs,” on page 363 for information about specifying colors.

Details

Once you have defined the color mapping, you use the new color name in any color option. For example, if your device entry maps the color name DAFFODIL to the SAS color value PAOY, you can specify the following:

```
pattern1 color=daffodil;
```

and the driver will map this to the color value PAOY.

COLLATE, NOCOLLATE Graphics Options

Specifies whether to collate the output, if collation is supported by the device.

Used by: GOPTIONS statement, OPTIONS statement

Default: NOCOLLATE

Restrictions: This option is hardware-dependent.
PostScript printers require a PPD file.
This option is not supported by Java or ActiveX.

See: GPROLOG on page 634
PPDFILE on page 677
“Options Used in Both GOPTIONS and OPTIONS Statements” on page 582

Syntax

COLLATE | NOCOLLATE

Comparisons

A limited number of printers can *collate* output, which means to separate each copy of printed output when you print multiple copies of output.

For PostScript printers, if a device's PPD file has Collate defined as “True”, the COLLATE option is supported.

For PCL printers that support collation, use the GPROLOG= option to specify a Printer Job Language (PJL) command to enable the collation. For information about the appropriate PJL command, consult the Printer Commands section of your printer's user manual.

COLORS Device Parameter and Graphics Option

Specifies the foreground colors used to produce your graphics output if you do not specify colors explicitly in program statements.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Gcolors window

Default: device-dependent

Example: "Example 3: Rotating Plot Symbols through the Color List" on page 535

Syntax

Form 1: GOPTIONS:

COLORS=<(*colors-list* | NONE)>

Form 2: GDEVICE:

COLORS=(<*colors-list*>)

Parameter Values

colors-list

specifies one or more SAS color names. If you specify more than one color, separate each name with a blank. See Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363 for information about specifying colors and using a color list.

To change some of the colors in the color list and retain others, you can use a null value for colors that you do not want to change. For example, to change COLORS=(RED GREEN BLUE) to COLORS=(WHITE GREEN BROWN), you can specify COLORS=(WHITE,BROWN).

NONE

tells SAS/GRAPH to use only the colors that you explicitly specify in program statements and to ignore the device's default color list.

Note: If you specify COLORS=(NONE) in your GOPTIONS statement or COLORS=() in your GDEVICE procedure statement, and you omit a color specification for a graphics element such as patterns, SAS/GRAPH selects at random one of the colors already specified in your program.

Details

The order of the colors in the list is important when you use default colors. For example, the colors used for titles, axes, and surfaces in the G3D procedure are assigned by default according to their position in the color list.

Note: Colors can be assigned to graph elements in different orders by different devices such as Java and ActiveX.

If you omit or reset COLORS=, SAS/GRAPH uses the default color list for the current device. To explicitly reset the color list to the device default, specify either

```
goptions colors=;
goptions colors=();
```

If you use default patterns with a color list specified by COLORS= option, the patterns rotate through every color in the list. If the color list contains only one color (for example, COLORS=(BLUE)), the solid pattern is skipped and the patterns rotate through only the appropriate default hatch patterns for the graph.

Note: By default, if black is the first color in a device's color list, default pattern rotation skips black as a pattern color, but uses black as the area-outline color. Thus, the outline color is never the same as an area's fill color. Using COLORS= to change the color list changes this default pattern behavior. When COLORS= is used, all colors in the specified color list are used in color rotation, and the outline color is the first color in the specified color list. Thus, the outline color will match any area using the first color as its fill.

See "PATTERN Statement" on page 455 for more information about pattern rotation.

TIP You can use the COLOR= option on the global PATTERN statement, the CTEXT= option, or the CTITLE= option in the GOPTIONS statement to control the color.

COLORTBL Device Parameter

An eight-character field in the Gcolors window that is not currently implemented. SAS/GRAPH ignores any value entered into this field.

COLORTYPE Device Parameter

Specifies the color space used by the user-written part of the Metagraphics device driver.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Default: NAME

Syntax

COLORTYPE=NAME | RGB | RGBA | HLS | GRAY | CMY | CMYK | HSV | HSB

Parameter Values

See Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363 for a description of these color types.

NAME
specifies SAS predefined color names.

RGB
specifies red-green-blue (RGB) colors.

RGBA
specifies RGB colors with an alpha channel to control color opacity.

HLS
specifies hue-lightness-saturation(HLS) colors.

GRAY
specifies gray-scale level colors.

CMY
specifies cyan-magenta-yellow colors.

CMYK
specifies cyan-magenta-yellow-black colors..

HSV | HSB
specifies hue-saturation-value (HSV) or hue-saturation-brightness (HSB) colors.

Details

Use the COLORTYPE device parameter also to specify the color-naming scheme that is used for devices that support more than one color-naming scheme.

For information about Metagraphics drivers, contact Technical Support.

COLS Device Parameter

Sets the number of columns that the device-resident font uses.

Used by: GDEVICE Chartype window, GDEVICE procedure, CHARREC= option

Default: 0

See: CHARREC on page 594

Details

For information about the syntax, see “CHARREC Device Parameter” on page 594.

If you are using a device driver from SASHELP.DEVICES, this parameter is already set for device-resident fonts that have been defined for your installation. If you are adding to or modifying the device-resident fonts available for a particular device driver, specify a positive value for the COLS device parameter. If COLS is greater than 0, it overrides the values of the LCOLS and PCOLS device parameters. For scalable fonts, you can specify 1 for COLS, and the actual number of columns will be computed based on the current text width.

CPATTERN Graphics Option

Selects the default color for PATTERN definitions when a color has not been specified.

Used by: GOPTIONS statement

Default: first color in current color list

Restriction: This option is not supported by Java or ActiveX.

Syntax

CPATTERN=*pattern-color*

Parameter Values

pattern-color
specifies the pattern color using any SAS/GRAPH color name. See Chapter 23, “Using Colors in SAS/GRAPH Programs,” on page 363 for information about specifying colors.

Details

CPATTERN= is overridden by any color specification in a PATTERN statement. For details about how CPATTERN= affects the PATTERN statement, see “The Effect of the CPATTERN= Graphics Option” on page 468.

If you specify CPATTERN=, the solid pattern is skipped and the patterns rotate through only the appropriate default hatch patterns for the graph. See “PATTERN Statement” on page 455 for more information about pattern rotation.

CSYMBOL Graphics Option

Specifies the default color for SYMBOL definitions when a color has not been specified.

Used by: GOPTIONS statement

Default: first color in current color list

Restriction: This option is not supported by Java or ActiveX.

Syntax

CSYMBOL=*symbol-color*

Parameter Values

symbol-color
specifies the symbol color using any SAS/GRAPH color name. See Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363 for information about specifying colors.

Details

CSYMBOL= is overridden by any color specification in a SYMBOL statement. See "SYMBOL Statement" on page 470.

CTEXT Graphics Option

Selects the default color for all text and the border.

Used by:	GOPTIONS statement
Defaults:	Java and ActiveX devices: black all other devices: the first color in current color list
Restriction:	This option is partially supported by Java.
See:	CTITLE on page 603

Syntax

CTEXT=*text-color*

Parameter Values

text-color
specifies the text color using any SAS/GRAPH color name. See Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363 for information about specifying colors.

Details

The CTITLE= graphics option overrides CTEXT= for all titles, notes, and footnotes, as well as the border. Any other color specifications for text in SAS statements also override the value of the CTEXT= graphics option.

Note: When you use ODS to send graphics to an HTML destination, and titles and footnotes are rendered as part of the HTML body file instead of the graphic image,

you must specify the ODS USEGOPT statement for this option to work. See "Using Graphics Options with ODS (USEGOPT)" on page 116 for more information.

CTITLE Graphics Option

Selects the default color for all titles, footnotes, and notes, and the border.

Used by: GOPTIONS statement

Defaults: (1) color specified by CTEXT=, if used

(2) based on device:

Java and ActiveX:	black
all other devices:	the first color in current color list

See: CTEXT on page 602

Syntax

CTITLE=*title-color*

Parameter Values

title-color

specifies the title color using any SAS/GRAPH color name. See Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363 for information about specifying colors.

Details

Any color specification in a TITLE, FOOTNOTE, or NOTE statement overrides the value of the CTITLE= graphics option for the text. The border, however, still uses the color specified in the CTITLE= graphics option.

Note: When you use ODS to send graphics to an HTML destination, and titles and footnotes are rendered as part of the HTML body file instead of the graphic image, you must specify the ODS USEGOPT statement for this option to work. See "Using Graphics Options with ODS (USEGOPT)" on page 116 for more information.

DASH, NODASH Device Parameters and Graphics Options

Specifies whether to use the device's hardware dashed-line capability, if available.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

See: DASHLINE on page 604

Syntax

Form 1: GOPTIONS:

DASH | NODASH

Form 2: GDEVICE:

DASH=Y | N

Parameter Values

DASH
DASH=Y
causes SAS/GRAPH to use the built-in hardware dashed-line drawing capability of the device when generating graphics output. A blank `Dash` field in the Parameters window is the same as DASH=Y.

Tip Hardware drawing is faster, but not all devices have the capability. SAS/GRAPH does not try to use the capability if your device does not support it.

NODASH
DASH=N
causes SAS/GRAPH to draw the dashed lines.

DASHLINE Device Parameter

Specifies which dashed lines should be generated by hardware means if possible.

Used by: GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

See: DASH on page 604

Syntax

DASHLINE='*dashed-line-hex-string*'X

Parameter Values

dashed-line-hex-string

specifies a 16-character hexadecimal string where each bit in the string corresponds to a line type. All 64 bits must be specified. Setting a bit to 1 selects the correspond line type. The bits are numbered 1–64, from left to right, where bit 1 is the most-significant bit in the hexadecimal string. Figure 25.52 on page 503 lists the line type that correspond to each bit. To use line type 1, turn on bit 1; to use line type 2, turn on bit 2; and so on. For example, in the following option the first byte is '1000'; only bit 1 is on and only line type 1 is selected:

```
dashline='8000000000000000'x
```

To turn on both bits 1 and 2, specify '1100' as the first byte as shown in the following:

```
dashline='c000000000000000'x
```

Bit 1 should always be on because it corresponds to a solid line.

Details

If the DASH device parameter is N in the device entry or if NODASH is used in a GOPTIONS statement, SAS/GRAPH ignores the hexadecimal string in the DASHLINE device parameter.

DASHSCALE Graphics Option

Scales the lengths of the dashes in a dashed line.

Used by:	GOPTIONS statement
Default:	DASHSCALE=1
Restriction:	This option is not supported by Java or ActiveX.

Syntax

DASHSCALE=*scaling-factor*

Parameter Values

scaling-factor
can be any number greater than 0. For example, GOPTIONS DASHSCALE=.5 reduces any existing dash length by one-half.

Details

Only dashes or spaces with lengths greater than one pixel are scaled. Dots are not scaled because their length is effectively zero. DASHSCALE= always uses system line styles instead of the device's dashed line capabilities.

DELAY Graphics Option

Controls the amount of time between graphs in the animation sequence.

Used by:	GOPTIONS statement
Default:	0
Restrictions:	This option applies to the GIF and SVGANIM device drivers only. This option is not supported by all browsers.
See:	Chapter 16, “Generating Animations,” on page 227 “Example: Creating an SVG Animation” on page 245

Syntax

DELAY=*delay-time*

Parameter Values

delay-time
specifies the length of time between graphs in units of 0.01 seconds. For example, to specify a delay of .03 seconds, specify DELAY=3.

Details

SAS/GRAPH puts the DELAY= value into the image file. Based on this value, the browser determines how to display the series of graphs.

DESCRIPTION Device Parameter

Provides a description of the device entry.

Used by: GDEVICE procedure, GDEVICE Detail window

Alias: DES

Default: none

Syntax

DESCRIPTION='*text-string*'

Parameter Values

text-string
is a string up to 256 characters long. This is a comment field and does not affect the graphics output.

DEVADDR Graphics Option

Specifies the location of the device to which the output of device drivers is sent.

Used by: GOPTIONS statement

Default: host-dependent

Restriction: IBM mainframe systems only

Syntax

DEVADDR=*device-address*

Parameter Values

device-address
specifies the address of the device to which the device driver output is to be sent.

DEVICE Graphics Option

Specifies the device driver to which SAS/GRAPH sends the procedure output. The device driver controls the format of graphics output.

Used by: GOPTIONS statement, OPTIONS statement

Alias: DEV

Default: device-dependent

Note: When you are running SAS without a terminal, you must either specify the DEVICE= option in the GOPTIONS statement, or specify the DEVICE= or PRINTER= option in the ODS destination statement. A terminal is not associated with your SAS session if you are running in batch mode on z/OS, if you are running SAS on UNIX without an X Server, or if you have specified the NOTERMINAL system option.

See: "Options Used in Both GOPTIONS and OPTIONS Statements" on page 582

Syntax

DEVICE=*device-entry*

Parameter Values

device-entry
specifies the name of a device entry that is stored in a device catalog.

Details

A device driver can direct graphics output to a hardware device, such as a terminal or a printer, or can create an external file in another graphics file format, such as TIF, GIF, or PostScript. Some device drivers also generate both graphics files and HTML files that can be viewed with a web browser.

Usually, a device driver is assigned by default. If a default driver is not assigned or if you specify RESET=ALL in a GOPTIONS statement, and you do not specify a device driver, SAS/GRAPH prompts you to enter a driver name when you execute a procedure that produces graphics output. If you are producing a graph to the screen and the Graph window is active, SAS/GRAPH selects the display driver for you automatically.

For a description of device drivers and for more information about selecting a device entry and changing device parameters, see "Using Graphics Devices" on page 89.

For information about using device drivers to display and print graphics output, see Chapter 11, "SAS/GRAPH Output," on page 119.

For information about using device drivers to export graphics output to external files, see "Specifying the Graphics Output File Type for Your Graph" on page 124. For

information about using device drivers to create output for the web, see “Developing Web Presentations with the PNG, SVG, and GIF Devices” on page 179and “Developing Web Presentations with the JAVAIMG and ACTXIMG Devices” on page 180.

DEVMAP Device Parameter and Graphics Option

Specifies the device map to be used when device-resident fonts are used.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

DEVMAP=*device-map-name* | NONE

Parameter Values

device-map-name
is a string up to eight characters long that is the name of the device map entry.

NONE
specifies that you do not want to use a device map. This can cause text to be displayed incorrectly or not at all.

Details

Device maps usually are used only when national characters appear in the text and you want them to display properly.

DEVOPTS Device Parameter

Specifies the hardware capabilities of the device.

Used by: GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Syntax

DEVOPTS='*hardware-capabilities-hex-string*'X

Parameter Values

hardware-capabilities-hex-string

specifies a 16-character hexadecimal string where each bit in the string corresponds to a device capability. All 64 bits must be specified. Setting a bit to 1 specifies the corresponding capability. The bits are numbered 0–63, from left to right, where bit 0 is the most-significant bit in the hexadecimal string. Setting a bit to 1 turns the bit on, and setting a bit to 0 turns it off. The following table lists the capability that corresponds to each bit.

Table 26.1 *Device Capabilities Represented in the DEVOPTS String*

Bit On	Capability
0	hardware circle generation
1	hardware pie fill supported
2	scalable hardware characters
3	device is a CRT-type (See TYPE device parameter)
4	translate table needed for non-ASCII hosts
5	hardware polygon fill available
6	hardware characters cell-aligned
7	user-definable colors supported
8	hardware polygons with multiple boundaries supported
9	not used
10	not used
11	adjustable hardware line width
12	double-byte font (non-US) supported
13	hardware repaint supported
14	hardware characters supported
15	no hard limit on x coordinate
16	no hard limit on y coordinate
17	not used
18	ability to justify proportional text

Bit On	Capability
19	driver can produce dependent catalog entries
20	device cannot draw in default background color
21	flush device buffer when filled
22	colors defined using HLS
23	colors defined using RGB
24	not used
25	polyline supported
26	polymarker supported
27	graphics clipping supported
28	not used
29	linkable device driver
30	select CHARTYPE by name in CHARREC entries
31	device-dependent pattern support
32	treat SCALABLE=Y CHARREC as metric
33	size CHARTYPE as HW from CHARREC entries
34	device supports rotated arcs
35	device supports target fonts
36	device supports drawing images
37	device supports multiple color maps
38	image rotation direction
39	device requires sublib for image rotation
40	device is a 24 bit truecolor machine
41	device supports setting font attributes
42	use scan line font rendering

Bit On	Capability
43	device can scale images
44	text clipping supported
45	non-interactive color device
46	driver does prolog processing
47	driver does epilog processing
48	driver output uses only a file
49	driver output requires a directory or PDS
50	autosize text to fit rows and columns
51	default binding is SHORTEDGE
52	driver supports duplex printing
53	device does right edge binding
54	ActiveX device
55	Java device
56	device uses a universal printer driver

Details

Each capability in the table corresponds to a bit in the value of the DEVOPTS device parameter. For example, if your device can generate hardware pie fills, the second bit in the first byte of the DEVOPTS string should be turned on if you want the driver to use that capability. If your device is capable of generating only hardware circles and pie fills, specify a value of 'C000000000000000'X as your DEVOPTS value (the first byte is '1100' so that the first 2 bits of the first byte are set to 1). Many of the hardware capabilities specified in the DEVOPTS string are overridden by graphics options or other device parameters.

CAUTION
Do not modify the DEVOPTS device parameter unless you are building a Metagraphics driver. If you want to prevent a driver supplied by SAS from using certain hardware capabilities, change the specific device parameter or use the corresponding graphics option.

If the DEVOPTS string indicates that a capability is available, the driver uses it unless it is explicitly disabled by another device parameter or graphics option. If the DEVOPTS string indicates that the capability is not available, it is not used by the driver, even if the corresponding device parameter or graphics option indicates that it should be used. For example, if the DEVOPTS value indicates that the device can do a hardware pie fill, the driver uses the hardware pie fill capability unless the PIEFILL device parameter is set to N or NOPIEFILL has been specified in a GOPTIONS statement. However, if the DEVOPTS device parameter indicates that the device cannot do a hardware pie fill, the driver does not attempt to use one, even if the PIEFILL device parameter is set to Y or PIEFILL is used in a GOPTIONS statement.

DEVTYPE Device Parameter

Specifies the information required by SAS/GRAPH routines to determine the nature of the output device.

Used by: GDEVICE procedure, GDEVICE Host File Options window

Default: device-dependent

Syntax

DEVTYPE=*device-type*

Parameter Values

device-type

specifies an eight-character string containing either blanks or some token name that is interpreted by the host. A token name can be one of the following:

GTERM

indicates that the output device is a graphics device that will be receiving graphics data. Most device drivers use this value.

G3270

indicates that the output device is an IBM 3270 graphics data stream.

If your device is an IBM 3270 type of device, DEVTYPE= must be G3270.

Note: GTERM and G3270 are SAS/GRAPH device types. Other valid values depend on your operating environment. DEVTYPE supports any of the device-type values supported in the FILENAME statement. Refer to the SAS Help facility for the device types the FILENAME statement supports in your operating environment. In most cases, this field should not be changed.

DISPLAY, NODISPLAY Graphics Options

Specifies whether output is displayed on the graphics device but does not affect whether a graph is placed in a catalog.

Used by: GOPTIONS statement

Default: DISPLAY

Restriction: This option is not supported by Java or ActiveX.

Syntax

DISPLAY | NODISPLAY

Comparisons

In most cases, NODISPLAY suppresses *all* output except the catalog entry written to the catalog selected in the GOUT= option. Therefore, you usually specify NODISPLAY when you want to generate a graph in a catalog but do not want to display the graph on your monitor or terminal while the catalog entry is being produced.

DISPOSAL Graphics Option

Specifies what happens to the graphic after it is displayed.

Used by: GOPTIONS statement

Default: NONE

Restriction: GIF driver only

Syntax

DISPOSAL=NONE | BACKGROUND | PREVIOUS | UNSPECIFIED

Parameter Values

NONE
causes the graphic to be left in place after displaying.

BACKGROUND
causes the background color to be returned and the graph erased after displaying.

PREVIOUS
causes the graphic area to be restored with what was displayed in the area previously.

UNSPECIFIED
indicates that no action is necessary.

DRVINIT1, DRVINIT2 Device Parameters and Graphics Options

Specifies host commands to be executed before driver initialization.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host Commands window

Restriction: This option is not supported by Java or ActiveX.

Syntax

DRVINIT1='*system-command(s)*'
DRVINIT2='*system-command(s)*'

Parameter Values

system-command(s)
specifies a character string of up to 72 characters that contains one or more valid system commands and can be in upper- or lowercase letters. You can include more than one command in the string if you separate the commands with a command delimiter, which is host-specific. For example, some operating environments use a semicolon as a command delimiter.

Restriction The length of the entire string cannot exceed 72 characters.

Details

The DRVINIT command is executed before the driver is initialized. DRVINIT is typically used with FILECLOSE=DRIVERTERM to allocate a host file needed by the device driver.

DRVQRY, NODRVQRY Device Parameters

Specifies whether the device can be queried for information about the current device configuration.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

Syntax

DRVQRY | NODRVQRY

Comparisons

Generally, this setting is device-dependent and you should not change it.

DRVTERM1, DRVTERM2 Device Parameters and Graphics Options

Specifies host commands to be executed after the driver terminates.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host Commands window

Restriction: This option is not supported by Java or ActiveX.

Syntax

DRVTERM1='*system-command(s)*'
DRVTERM2='*system-command(s)*'

Parameter Values

system-command(s)
specifies a character string of up to 72 characters that contains one or more valid system commands and can be in upper- or lowercase letters. specifies a character string that is a valid system command and can be in upper- or lowercase letters. You can include more than one command in the string if you separate the commands with a command delimiter, which is host-specific. For example, some operating environments use a semicolon as a command delimiter.

Restriction The length of the entire string cannot exceed 72 characters.

Details

The DRVTERM command is executed after the driver terminates. DRVTERM is typically used with FILECLOSE=DRIVERTERM to de-allocate a host file and execute utility programs that send the data to the graphics device. For example, DRVTERM might specify commands to send the file to a host print queue.

DUPLEX, NODUPLEX Graphics Options

Specifies whether to use duplex printing if available on the device.

Used by: GOPTIONS statement, OPTIONS statement

Default: NODUPLEX

Restriction: duplex printers only

See: BINDING on page 589
GSFMODE on page 638
PPDFILE on page 677
“Options Used in Both GOPTIONS and OPTIONS Statements” on page 582

Syntax

DUPLEX | NODUPLEX

Comparisons

When DUPLEX is on, the driver sets up the printer for duplex operation. Before producing the first graph, set GSFMODE=REPLACE in the GOPTIONS statement, and DUPLEX on an OPTIONS or GOPTIONS statement. You can also use the BINDING= option in conjunction with DUPLEX. Before producing the second graph, set GSFMODE=APPEND in the GOPTIONS statement so that the driver knows to place succeeding graphs on the next available side of paper.

If DUPLEX is in effect, the page's inside (binding) margin is set equal to the current HORIGIN setting, and the outside margin is set equal to XMAX – HSIZE – HORIGIN.

In terms of even- and odd-numbered pages, this means the following:

odd-numbered pages
: HORIGIN determines the left margin, and XMAX – HSIZE – HORIGIN determines the right margin

even-numbered pages
: XMAX – HSIZE – HORIGIN determines the left margin, and HORIGIN determines the right margin

For PostScript printers, if you do not use the PPDFILE= option to specify a PPD (PostScript Printer Description) file, a generic PostScript Level 1 duplex command is added to the driver output. If PPDFILE= is used, the duplex command is obtained from the PPD file.

ERASE, NOERASE Device Parameters and Graphics Options

Specifies whether to erase graph after display.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Defaults: GOPTIONS: NOERASE
GDEVICE: ERASE=N

Restriction: This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

ERASE | NOERASE

Form 2: GDEVICE:

ERASE=Y | N

Parameter Values

ERASE
ERASE=Y
: causes the graph to be erased when you press Enter after the graph has been displayed.

NOERASE
ERASE=N
: causes the graph to remain on the display when you press Enter after the graph has been displayed. A blank `Erase` field in the Parameters window is the same as ERASE=N.

Details

ERASE is useful for those devices that overlay the graphics area and the message area – that is, those devices that have separate dialog box and graphics areas. On other devices, the graph is erased.

EXTENSION Graphics Option

Specifies the file extension for an external graphics file.

Used by: GOPTIONS statement

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

See: GACCESS on page 629
GSFNAME on page 639

Syntax

EXTENSION='*file-type*'

Parameter Values

file-type
a string up to eight characters long that is a file extension, such as GIF or CGM, that you want to append to an external file.

Details

The extension specified on EXTENSION= is used when the output destination is a storage location. The extension is ignored when the output destination is a file. To specify the output destination, you can use a FILENAME statement, or the graphics options GACCESS= or GSFNAME=.

Assuming that the output destination is a storage location,

- if EXTENSION='.', no extension is added to the file name
- if EXTENSION=' ' or EXTENSION= is not used, the driver's default extension is added to the file name
- if the driver has no default extension, SAS/GRAPH uses the default extension .GSF.

FASTTEXT, NOFASTTEXT Graphics Options

Specifies whether to use integer-based font processing for faster font rendering.

Used by: GOPTIONS statement

Default: NOFASTTEXT

Restriction: This option is not supported by Java or ActiveX.

Note: When the FASTTEXT option is specified and you use the Annotate facility, some round symbols in the Marker font, such as W, might appear noncircular, especially if you use the PNG device. In that case, use the NOFASTTEXT option instead.

Syntax

FASTTEXT | NOFASTTEXT

FBY Graphics Option

Selects the font for BY lines.

Used by: GOPTIONS statement

Defaults: (1) font specified by FTEXT=, if used
(2) deviceresident font
(3) simulate font

Restriction: This option is not supported by Java or ActiveX.

See: "BY Statement" on page 424

Syntax

FBY=*By line-font*

Parameter Values

By line-font
specifies the font for all BY lines on the graphics output. See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349 for information about specifying fonts.

Details

When you use a BY statement with a SAS/GRAPH procedure to process a data set in subgroups, each graph produced by that procedure is headed by a BY line that displays the BY variables and their values that define the current subgroup.

FCACHE Graphics Option

Specifies the number of system fonts to keep open at one time.

Used by: GOPTIONS statement

Default: FCACHE=3

Restriction: This option is not supported by Java or ActiveX.

Syntax

FCACHE=*number-fonts-open*

Parameter Values

number-fonts-open
specifies the number of system fonts to keep open. *Number-fonts-open* must be greater than or equal to zero.

Details

Each font requires from 4K to 10K memory. Graphs that use many fonts can run faster if you set the value of *number-fonts-open* to a higher number. However, graphs that use multiple fonts might require too much memory on some computer systems if all the fonts are kept open. In such cases, set the value of *number-fonts-open* to a lower number to conserve memory.

FILECLOSE Device Parameter and Graphics Option

Controls when the graphics stream file (GSF) is closed when you are using the device driver to send graphics output to a hard copy device.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

Default: DRIVERTERM (if a device is specified)

Restriction: This option is not supported by Java or ActiveX.

See: “Specifying the Graphics Output File Type for Your Graph” on page 124

Syntax

FILECLOSE=DRIVERTERM | GRAPHEND

Parameter Values

DRIVERTERM
closes the GSF and makes it available to the device after all graphs have been produced and the procedure or driver terminates. A host command might be needed to actually send the GSF to the device. Host commands can be specified with the DRVINIT or DRVTERM parameters or entered in the Host File Options window of the device entry. If multiple graphs are produced by a procedure, this specification creates one large file. Specifying DRIVERTERM is appropriate for batch processing because it is slightly more efficient to allocate the file only once.

Alias DRIVER

GRAPHEND
closes the GSF after each separate graph is produced and releases it to the device before sending another. This method creates smaller files if multiple graphs are produced by a procedure. You can specify a command that sends the graph to the device with the POSTGRAPH parameter or use the Host File Options window. Specifying GRAPHEND is appropriate for drivers that are used interactively, or for devices that require only one graph per physical file.

Alias GRAPH

FILEONLY, NOFILEONLY Graphics Options

Specifies whether a file location or a storage location is the default destination for graphics output.

Used by: GOPTIONS statement

Default: device-dependent

Restrictions: This option is ignored if the device requires the output destination to be a storage location.
This option is not supported by Java or ActiveX.

See: DEVOPTS on page 609
GSFNAME on page 639

Syntax

FILEONLY | NOFILEONLY

Parameter Values

FILEONLY
specifies that a file rather than a storage location is the default destination for graphics output.

NOFILEONLY
specifies that a storage location is the default destination for graphics output, unless a file of the same name exists.

Details

Most devices use FILEONLY as the default. However, devices that require the output destination to be a storage location use NOFILEONLY as the default. For example, the HTML device requires a storage location because it produces two types of output (HTML files and GIF image files) that cannot be written to the same file.

To determine what the default is for a particular device, look at the settings for DEVOPTS bits 48 and 49.

For more information, see "Specifying the Graphics Output File Type for Your Graph" on page 124.

FILL, NOFILL Device Parameters and Graphics Options

Specifies whether to use the device's hardware rectangle-fill capability.

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
Default:	device-dependent
Restriction:	This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

FILL | NOFILL

Form 2: GDEVICE:

FILL=Y | N

Parameter Values

FILL
FILL=Y
causes SAS/GRAPH to use the built-in hardware rectangle-filling capability of the device. A blank `Fill` field in the Parameters window is the same as FILL=Y.

Tip Hardware drawing is faster, but not all devices have the capability. SAS/GRAPH does not try to use the capability if your device does not support it.

NOFILL
FILL=N
causes SAS/GRAPH to use software fills to fill rectangles.

FILLINC Device Parameter and Graphics Option

Specifies the number of pixels to move before drawing the next line in a software fill of a solid area.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

See: FILL on page 623
PIEFILL on page 673
POLYGONFILL on page 675

Syntax

FILLINC=0–9999

Comparisons

In order for FILLINC to have any effect, a software fill must be used. To force a software fill, use the options NOFILL, NOPIEFILL, and NOPOLYGONFILL in a GOPTIONS statement.

If FILLINC is set to 0 or 1, adjacent lines are used (solid fill with no gaps). If FILLINC is set to 2, a pixel-width line is skipped before drawing the next line of a fill.

This option can be useful for keeping plotters from over saturating a solid area and for speeding the plotting. Some inks spread on paper. The type of paper used can also affect ink spread.

FONT NAME Device Parameter

Specifies the device-resident font associated with CHARTYPE.

Used by: GDEVICE Chartype window, GDEVICE procedure, CHARREC= option

Requirement: if adding or modifying a CHARREC

See: CHARREC on page 594

Details

For information about the syntax, see “CHARREC Device Parameter” on page 594.

Use FONT NAME if you are adding to or modifying the device-resident fonts available for a particular device driver. The fonts that you specify must be valid for the output device. If you are using a device entry supplied by SAS, this parameter already is set for most available device-resident fonts.

FONTRES Graphics Option

Controls the resolution of Bitstream fonts.

Used by: GOPTIONS statement

Default: NORMAL

Restriction: Not supported by Java or ActiveX

See: FASTTEXT on page 619
FCACHE on page 621
RENDER on page 688
RENDERLIB on page 689
SWFONTRENDER on page 697

Syntax

FONTRES=NORMAL | PRESENTATION

Parameter Values

NORMAL
renders fonts in memory using integer rendering routines, which improves character drawing speed for most host systems. NORMAL has the same effect as specifying the default values for these graphics options.

```
render=memory
renderlib=work
fasttext
fcache=0
```

PRESENTATION
disables the storage or use of rendered versions of Bitstream fonts, but produces the fonts at their highest resolution. FONTRES=PRESENTATION has the same effect as specifying these graphics options:

```
render=none
renderlib=work
nofasttext
fcache=3
```

FORMAT Device Parameter

Sets the file format of the metacode file produced by the part of the Metagraphics device driver that is supplied by SAS.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Default: CHARACTER

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

FORMAT=CHARACTER | BINARY

Comparisons

A blank field defaults to CHARACTER. For information about Metagraphics drivers, contact Technical Support.

FTEXT Graphics Option

Sets the default font for all text.

Used by: GOPTIONS statement

Default: Default deviceresident font (except the first title)

Restriction: This option is partially supported by Java and ActiveX.

See: FTITLE on page 627

Syntax

FTEXT=*text-font*

Parameter Values

text-font
specifies the font for all text on the graphics output. See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349 for information about specifying fonts.

Details

The FTITLE= graphics option overrides FTEXT= for the *first* title. Not all fonts are supported by the ACTIVEX and ACTXIMG devices.

Note: When you use ODS to send graphics to an HTML destination, and titles and footnotes are rendered as part of the HTML body file instead of the graphic image, you must specify the ODS USEGOPT statement for this option to work. See "Using Graphics Options with ODS (USEGOPT)" on page 116 for more information.

FTITLE Graphics Option

Selects the default font for the first TITLE line.

Used by:	GOPTIONS statement
Defaults:	(1) font specified by FTEXT=, if used (2) value of the style variable (3) device-resident font (4) simulate font
See:	FTEXT on page 626

Syntax

FTITLE=*title-font*

Parameter Values

title-font
specifies the font for the TITLE1 statement. See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349 for information about specifying fonts.

Details

Note: When you use ODS to send graphics to an HTML destination, and titles and footnotes are rendered as part of the HTML body file instead of the graphic image, you must specify the ODS USEGOPT statement for this option to work. See "Using Graphics Options with ODS (USEGOPT)" on page 116 for more information.

FTRACK Graphics Option

Controls the amount of space between letters in the Bitstream fonts supplied by SAS (Brush, Century, Swiss, and Zapf).

Used by: GOPTIONS statement

Default: TIGHT

Restriction: This option is not supported by Java or ActiveX.

Syntax

FTRACK=LOOSE | NONE | NORMAL | TIGHT | TOUCH | V5

Parameter Values

LOOSE
leaves the most visible space between characters and produces a longer string.

NONE
spacing depends on the size of the font. NONE might produce a shorter or longer string than LOOSE for the same font at different point sizes, because some sizes add space between the characters while others remove it.

NORMAL
is the recommended setting.

TIGHT
reduces the space between characters.

TOUCH
leaves the least visible space between characters.

V5
places a fixed amount of space between the characters and does not adjust for the shape of the character. That is, it does not support kerning. This spacing is compatible with Version 5 Bitstream fonts.

Details

The spacing that you specify with FTRACK= affects all Bitstream text in a graph. For example, you cannot produce TIGHT Century type and LOOSE Zapf type simultaneously. This option has no effect on other font types.

Because the value of FTRACK= is stored with the graph, the spacing that you specify when the graph is created is always used when the graph is replayed.

GACCESS Device Parameter and Graphics Option

Specifies the format or the destination or both of graphics data written to a traditional SAS/GRAPH device driver or to a graphics stream file (GSF).

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

Default: device-dependent

Restriction: This option is not supported by Java, ActiveX, or shortcut devices.

See: “Using Graphics Devices” on page 89 for more information about devices.

Syntax

GACCESS=*output-format* | '*output-formatdestination*'

Parameter Values

output-format

specifies the format or the destination (the SAS log or a fileref) of the graphics data. *Output-format* varies according to the operating environment. These values can be specified in all operating environments:

SASGASTD

specifies that a continuous stream of data is written.

SASGASTD is the default for most devices and is typically appropriate when the output file will be sent directly to a device. If you specify GACCESS=SASGASTD, use the GSFNAME= and GSFMODE= graphics options or device parameters to direct your graphics output to a GSF.

SASGAEDT

specifies that the file be host-specific edit format.

Some hosts allow editing by inserting characters at the end of each record. SASGAEDT is typically used when the output file is to be edited later. If you specify GACCESS=SASGAEDT, use the GSFNAME= and GSFMODE= graphics options or device parameters to direct your graphics output to a GSF.

SASGAFIX

specifies that fixed-length records be written.

The record length is controlled by the value of the GSFLEN= graphics option or device parameter, or the sixth byte of the PROMPTCHARS value. The records are padded with blanks where necessary. SASGAFIX is typically used when the output file will be transferred to a computer that requires fixed-length records. If you specify GACCESS=SASGAFIX, use the GSFNAME= and GSFMODE= graphics options or device parameters to direct your graphics output to a GSF.

Note: The value of the GPROTOCOL= graphics option or device parameter can greatly affect the length of the records; for example, if GPROTOCOL=SASGPLCL, the length of the records is doubled.

SASGALOG
specifies that records are to be written to the SAS log.

GSASFILE
specifies that the records are to be written to the destination whose fileref is GSASFILE.

The fileref can point to a specific external file or to an aggregate file location. See "FILENAME Statement" on page 52 for more information about specifying a fileref.

'*output-format destination*'
specifies the destination in addition to one of these output format values: SASGASTD, SASGAEDT, or SASGAFIX. *Destination* is the physical name of an external file or aggregate file location, or of a device. For details about specifying the physical name of a destination, see the SAS documentation for your operating environment.

This form is not available in all operating environments. See "Specifying the Graphics Output File Type for Your Graph" on page 124 for more information about creating graphics stream files.

Note: In the `Gaccess` field of the Host File Options window, you can specify a destination without an output format. In that case, the format defaults to SASGASTD. When you specify a value in the `Gaccess` field, you do not need to quote it.

Operating Environment Information: Depending on your operating environment, you might be able to specify other values for GACCESS=. See the SAS companion for your operating environment for additional values.

GCOPIES Device Parameter and Graphics Option

Sets the current and maximum number of copies to be printed.

Used by:	GOPTIONS statements, GDEVICE Parameters window, GDEVICE procedure, OPTIONS statement
Defaults:	GOPTIONS: GCOPIES=(0,20) GDEVICE: GCOPIES=0
Restriction:	This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

GCOPIES=(<*current-copies*> <,*max-copies*>)

Form 2: GDEVICE:

GCOPIES=*current-copies*

Parameter Values

current-copies
specifies the current number of copies as an integer in the range 0–*max-copies*. A value of 0 or 1 produces a single copy.

Restriction *Current-copies* cannot exceed *max-copies*.

max-copies
specifies the maximum copies as an integer in the range 1–255.

Details

If you do not specify GCOPIES, a default number of copies is searched for in this order:

1. the number of copies specified on an OPTIONS COPIES setting
2. 0 current copies, and 20 maximum copies.

Not all devices have the capability to print multiple copies. See the `Gcopies` field in the Parameters window for your device to determine its capabilities.

GEND Device Parameter and Graphics Option

Appends an ASCII string to every graphics data record that is sent to a device or file.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Gend window

Restriction: This option is not supported by Java or ActiveX.

See: GSTART on page 641

Syntax

GEND='*string*' <'*string-n*'>

Parameter Values

'*string*'

specifies the ASCII string to append to every graphics data record that is sent to a device or file. *String* be either of the following:

- '*hex-string*'X
- '*character-string*'

In a GOPTIONS statement or in the GDEVICE procedure ADD or MODIFY statement, you can specify multiple strings with the GEND= option. In this case, you can mix the formats, specifying some as ASCII hexadecimal strings and some as character strings. Multiple strings are concatenated automatically.

In the GEND window, enter the hexadecimal string without either quotation marks or a trailing x. Note, however, that the string must be entered as a hexadecimal string.

PROC GOPTIONS always reports the value as a hexadecimal string.

Details

GEND is useful if you are creating a file and want to insert a carriage return at the end of every record. You can also use GEND in conjunction with the GSTART= graphics option or device parameter.

If you must specify the long and complicated initialization strings required by some devices (for example, PostScript printers), it is easier to use the GOPTIONS GEND= option rather than the GDEVICE Gend window because it is easier to code the string as text with GEND= than it is to convert the string to its ASCII representation, which is required to enter the string in the GDEVICE Gend window.

Note: On non-ASCII hosts, only ASCII hexadecimal strings produce consistent results in all instances because of how the character strings are translated. In addition, the only way to specify a value for GEND that can be used by all hosts is to use an ASCII hexadecimal string. Therefore, using an ASCII hexadecimal string to specify a value for GEND is the recommended method.

GEPILOG Device Parameter and Graphics Option

Sends a string to a device or file after all graphics commands are sent.

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Gepilog window
Restriction:	This option is not supported by Java or ActiveX.
See:	PREGEPILOG on page 678 POSTGEPILOG on page 675

Syntax

GEPILOG='*string*' <'*string-n*'>

Parameter Values

'*string*'
specifies the string to a device or file after all graphics commands are sent. *String* can be either of the following:

- '*hex-string*'X
- '*character-string*'

In a GOPTIONS statement or in the GDEVICE procedure ADD or MODIFY statement, you can specify multiple strings with the GEPILOG= option. In this case, you can mix the formats, specifying some as ASCII hexadecimal strings and some as character strings. Multiple strings are concatenated automatically.

In the Gepilog window, enter the hexadecimal string without either quotation marks or a trailing x. Note, however, that the string must be entered as a hexadecimal string.

PROC GOPTIONS always reports the value as a hexadecimal string.

Details

GEPILOG can be used in conjunction with the GPROLOG= graphics option or device parameter.

If you must specify the long and complicated initialization strings required by some devices (for example, PostScript printers), it is easier to use the GOPTIONS GEPILOG= option rather than the Gepilog window because it is easier to code the string as text with GEPILOG= than it is to convert the string to its ASCII representation, which is required to enter the string in the Gepilog window.

Note: On non-ASCII hosts, only ASCII hexadecimal strings produce consistent results in all instances because of how the character strings are translated. In addition, the only way to specify a value for GEPILOG that can be used by all hosts is to use an ASCII hexadecimal string. Therefore, using an ASCII hexadecimal string to specify a value for GEPILOG is the recommended method.

GOUTMODE Graphics Option

Appends to or replaces the graphics output catalog.

Used by:	GOPTIONS statement
Default:	APPEND
Restriction:	This option is not supported by Java or ActiveX.

Syntax

GOUTMODE=APPEND | REPLACE

Parameter Values

APPEND
adds each new graph to the end of the current catalog.

REPLACE
replaces the contents of the catalog with the graph or graphs produced by a single procedure.

CAUTION
If you specify REPLACE, the entire contents of the catalog are replaced, not just graphs of the same name. Graphs are added to the catalog for the duration of the procedure, but when the procedure ends and a new procedure begins, the contents of the catalog are deleted and the new graph or graphs are added

GPROLOG Device Parameter and Graphics Option

Sends a string to device or file before graphics commands are sent.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Gprolog window

Restriction: This option is not supported by Java or ActiveX.

See: PREGPROLOG on page 679
POSTGPROLOG on page 676

Syntax

GPROLOG='*string*' <'*string-n*'>

Parameter Values

'*string*'
specifies the string to a device or file before all graphics commands are sent. *String* can be either of the following:

- '*hex-string*'X
- '*character-string*'

In a GOPTIONS statement or in the GDEVICE procedure ADD or MODIFY statement, you can specify multiple strings with the GPROLOG= option. In this case, you can mix the formats, specifying some as ASCII hexadecimal strings and some as character strings. Multiple strings are concatenated automatically.

In the GPROLOG window, enter the hexadecimal string without either quotation marks or a trailing x. Note, however, that the string must be entered as a hexadecimal string.

PROC GOPTIONS always reports the value as a hexadecimal string.

Details

GPROLOG can be used in conjunction with the GEPILOG= graphics option or device parameter.

If you must specify the long and complicated initialization strings required by some devices (for example, PostScript printers), it is easier to use the GOPTIONS GPROLOG= option rather than the GDEVICE Gprolog window because it is easier to code the string as text with GPROLOG= than it is to convert the string to its ASCII representation, which is required to enter the string in the GDEVICE Gprolog window.

Note: On non-ASCII hosts, only ASCII hexadecimal strings produce consistent results in all instances because of how the character strings are translated. In addition, the only way to specify a value for GEND that can be used by all hosts is to use an ASCII hexadecimal string. Therefore, using an ASCII hexadecimal string to specify a value for GEND is the recommended method.

GPROTOCOL Device Parameter and Graphics Option

Specifies the protocol module to use when routing output directly to a printer or creating a graphics stream file (GSF) to send to a device attached to your host by a protocol converter.

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window
Default:	host-dependent
Restriction:	This option is not supported by Java or ActiveX.

Syntax

GPROTOCOL=*module-name*

Parameter Values

module-name
specifies the protocol module name. *Module-name* can be one of the following:

SASGPADE* SASGPHYD* SASGPMIC*

SASGPAGL*	SASGPIDA*	SASGPRTM*
SASGPASC	SASGPIDX*	SASGPSCS*
SASGPAXI*	SASGPIMP*	SASGPSTD
SASGPCAB*	SASGPIOC*	SASGPSTE*
SASGPCHK*	SASGPISI*	SASGPTCX*
SASGPDAT*	SASGPI24*	SASGPVAT*
SASGPDCA*	SASGPLCL*	SASGP497*
SASGPHEX	SASGPNET*	SASGP71

*Valid only for IBM mainframe systems.

Details

GPROTOCOL= specifies whether the graphics data generated by the SAS/GRAPH device driver should be altered and how the data should be altered. Unless you are using a protocol converter on an IBM mainframe, most devices do not require that the data be altered, and ordinarily, you do not have to change the default of GPROTOCOL.

On IBM hosts, the protocol module converts the graphics output to a format that can be processed by protocol converters. On other hosts, it can be used to produce a file in ASCII hexadecimal format.

Refer to the SAS Help facility for descriptions of these protocol modules.

Operating Environment Information: GPROTOCOL is valid only in certain operating environments.

GRAPHRC, NOGRAPHRC Graphics Options

Specifies whether to return a step code at graphics procedure termination.

Used by: GOPTIONS statement

Default: GRAPHRC

Restriction: This option is not supported by Java or ActiveX.

Syntax

GRAPHRC | NOGRAPHRC

Parameter Values

GRAPHRC
allows a return code at procedure termination. If the return code is not 0, the entire job might terminate.

NOGRAPHRC
always returns a step code of 0, even if the SAS/GRAPH program produced errors. As a result, the entire job's return code is unaffected by errors in any graphics procedure. NOGRAPHRC also overrides the ERRABEND system option.

Details

You typically use this option when you are running multiple jobs in a batch environment. It is useful primarily in an z/OS batch environment.

GSFLEN Device Parameter and Graphics Option

Controls the length of records written to the graphics stream file (GSF).

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window
Default:	device-dependent
Restriction:	This option is not supported by Java or ActiveX.
See:	PROMPTCHARS on page 683

Syntax

GSFLEN=*record-length*

Parameter Values

record-length
specifies the length of the records written by the driver to a GSF or to the device as an integer in the range 0–99999. If GSFLEN is 0, SAS/GRAPH uses the sixth byte of the PROMPTCHARS string to determine the length of the records. If the sixth byte of the PROMPTCHARS string is 00, the device driver sets the record length.

If you specify GACCESS=SASGAFIX and omit GSFLEN=, SAS/GRAPH uses the default length for the device.

Some values of the GPROTOCOL device parameter cause each byte in the data stream to be expanded to two bytes. This expansion is done after the length of the record is set by GSFLEN. If you are specifying a value for GPROTOCOL that does this (for example, SASGPHEX, SASGPLCL, or SASGPAGL), specify a value for GSFLEN that is half of the actual record length desired. For example, a value of 64 produces a 128-byte record after expansion by the GPROTOCOL module.

GSFMODE Device Parameter and Graphics Option

Specifies the disposition of records written to a graphics stream file (GSF) or to a device or communications port by the device driver.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

Default: REPLACE

Restriction: This option is not supported by Java or ActiveX.

See: GACCESS on page 629
GSFNAME on page 639

Syntax

GSFMODE=APPEND | PORT | REPLACE

Parameter Values

APPEND
adds the records to the end of a GSF designated by the GACCESS= or GSFNAME= graphics option or device parameter.

Note: The APPEND option is useful only with the GIF graphics output device, and the PCL and PostScript Universal Printer devices. When used with other devices, such as PNG, SVG, or PDF, only the first graph is displayed in the viewer.

If the file does not already exist, it is created. The destination can be either a specific file or an aggregate file storage location. If the destination of the GSF is a specific file and you specify APPEND, SAS/GRAPH will add the new records to an existing GSF of the same name. If the destination of the GSF is a file location and not a specific file, SAS/GRAPH will add the records to an external file whose name matches the name of the newly created catalog entry. For more information about how SAS/GRAPH names catalog entries, see “Specifying the Graphics Output File Type for Your Graph” on page 124.

Note: Some viewers of bitmapped output can view only one graph, even though multiple graphs are stored in the file. Therefore, it might appear that a file contains only one graph when in fact, it contains multiple graphs.

PORT
sends the records to a device or communications port. The GACCESS= graphics option or device parameter should point to the desired port or device.

REPLACE
replaces the existing contents of a GSF designated by the GACCESS= or GSFNAME= graphics option or device parameter. If the file does not exist, it is

created. REPLACE is always the default, regardless of the destination of the GSF. If the destination of the GSF is a specific file and you specify REPLACE, SAS/GRAPH will replace an existing GSF with the contents of a newly created GSF of the same name. If the destination of the GSF is a file location and not a specific file, SAS/GRAPH will replace an external file whose name matches the name of the newly created catalog entry. For more information about how SAS/GRAPH names catalog entries, see "Specifying the Graphics Output File Type for Your Graph" on page 124.

Details

When you create a GSF, the GSFNAME= or GACCESS= graphics option or device parameter controls where the output goes, and GSFMODE= controls how the driver writes graphics output records. If the output is to go to a file, specify APPEND or REPLACE. If the output is to go directly to a device or to a communications port, specify PORT. See "Specifying the Graphics Output File Type for Your Graph" on page 124 for more information about creating a graphics stream file.

GSFNAME Device Parameter and Graphics Option

Specifies the fileref of the file or aggregate file location to which graphics stream file records are written.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

Restrictions: Not valid for IBM32*xx*, linkable, Metagraphics, Java, or ActiveX drivers.

In SAS 9.4M6 and earlier releases, only the PNG and JPEG graphics devices support file references to the SAS Viya Files service (FILESRVC access method). Starting with SAS Viya 3.5 and SAS Studio 5.2, the GIF and SVG graphics devices also support FILESRVC file references.

See: GACCESS on page 629
GSFMODE on page 638

Syntax

GSFNAME=*fileref*

Parameter Values

fileref

specifies a fileref that points to the destination for the graphics stream file (GSF) output. *Fileref* must be a valid SAS fileref up to eight characters long and must be assigned with a FILENAME statement before running a SAS/GRAPH procedure that uses that fileref. The destination specified by the FILENAME statement can be either a specific file or an aggregate file location. See "FILENAME Statement" on page 52 for additional information about the FILENAME statement.

Details

Whether the resulting graphs are stored as one file or many files depends on both the type of destination and the setting of the GSFMODE= option.

If you specify a fileref with GSFNAME= and forget the FILENAME statement that defines the fileref, and if a destination is specified by the GACCESS= graphics option or device parameter, SAS/GRAPH assigns that destination to the fileref and sends the graphics output there. See also "GACCESS Device Parameter and Graphics Option" on page 629.

See "Specifying the Graphics Output File Type for Your Graph" on page 124 for more information about creating graphics stream files.

GSFPROMPT, NOGSFPROMPT Graphics Options

Specifies whether to write prompt messages to the graphics stream file (GSF).

Used by: GOPTIONS statement

Default: NOGSFPROMPT

Restriction: This option is not supported by Java or ActiveX.

Syntax

GSFPROMPT | NOGSFPROMPT

Comparisons

When the GSF is processed by another program, that program can display the prompt messages. The default, NOGSFPROMPT, is compatible with Release 6.06.

Although the prompt messages appear if the graphics device is in eavesdrop mode, they do not wait for user response. If GSFPROMPT is on, the prompt messages are sent with the GSF to the device, regardless of the status of the graphics options PROMPT, GACCESS=, GSFMODE=, or GSFNAME=.

GSIZE Device Parameter and Graphics Option

Sets the number of lines of display used for graphics for devices whose displays can be divided into graphics and text areas.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

GSIZE=*lines*

Parameter Values

lines
specifies the number of lines to be used for graphics. *Lines* is a nonnegative integer up to three digits long (0–999), and can be larger or smaller than the total number of lines that can be displayed at one time. If the number is larger, scroll the graph to see it all. If GSIZE is 0, all lines are used for text.

GSTART Device Parameter and Graphics Option

Prefixes every record of graphics data sent to a device or file with a string of characters.

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Gstart window
Default:	none
Restriction:	This option is not supported by Java or ActiveX.
See:	GEND on page 631

Syntax

GSTART='*string*' <'*string-n*'>

Parameter Values

'*string*'
specifies the string that prefixes every record of graphics data sent to a device or file. *String* can be either of the following:

- '*hex-string*'X
- '*character-string*'

In a GOPTIONS statement or in the GDEVICE procedure ADD or MODIFY statement, you can specify multiple strings with the GSTART= option. In this case, you can mix the formats, specifying some as ASCII hexadecimal strings and some as character strings. Multiple strings are concatenated automatically.

In the GSTART window, enter the hexadecimal string without either quotation marks or a trailing x. Note, however, that the string must be entered as a hexadecimal string.

PROC GOPTIONS always reports the value as a hexadecimal string.

Details

GSTART is useful when sending a file to a device that requires each record be prefixed with some character. You can use GSTART= in conjunction with the GEND= graphics option or device parameter.

If you must specify the long and complicated initialization strings required by some devices (for example, PostScript printers), it is easier to use the GOPTIONS GSTART= option rather than the GDEVICE Gstart window because it is easier to code the string as text with GSTART= than it is to convert the string to its ASCII representation, which is required to enter the string in the GDEVICE Gstart window.

Note: On non-ASCII hosts, only ASCII hexadecimal strings produce consistent results in all instances because of how the character strings are translated. In addition, the only way to specify a value for GEND that can be used by all hosts is to use an ASCII hexadecimal string. Therefore, using an ASCII hexadecimal string to specify a value for GEND is the recommended method.

GUNIT Graphics Option

Specifies the default unit of measure to use with height specifications.

Used by:	GOPTIONS statement
Default:	CELLS
Restrictions:	This option is partially supported by Java and ActiveX. This option is not honored by the ODS styles.

Syntax

GUNIT=CELLS | CM | IN | PCT | PT

Parameter Values

CELLS
specifies character cells as the units.

CM
specifies centimeters as the units.

IN
specifies inches as the units.

PCT
specifies percentage of the graphics output area as the units.

PT
specifies points as the units (there are approximately 72 points in an inch).

Details

Used with options in the AXIS, FOOTNOTE, LEGEND, NOTE, SYMBOL, and TITLE statements and in some graphics options. If you specify a value but do not specify an explicit unit, the value of the GUNIT= graphics option is used. If the HSIZE= and VSIZE= options are specified, GUNIT is ignored and inches will be used.

GWAIT Graphics Option

Specifies the time between each graph displayed in a series.

Used by: GOPTIONS statement

Default: GWAIT=0

Restriction: This option is not supported by Java or ActiveX.

Syntax

GWAIT=*seconds*

Parameter Values

seconds
specifies the number of seconds between graphs. *Seconds* can be any reasonable positive integer. By default, GWAIT=0, which means that you must press the Enter key between each display in a series of graphs.

Details

GWAIT= enables you to view a series of graphs without having to press the Enter key between each display. For example, if you specify GWAIT=5, five seconds elapse between the display of each graph in a series. If you use the NOPROMPT graphics option, the GWAIT= graphics option is disabled.

HANDSHAKE Device Parameter and Graphics Option

Specifies the type of flow control used to regulate the flow of data to a hard copy device.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: host-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

HANDSHAKE=HARDWARE | NONE | SOFTWARE | XONXOFF

Parameter Values

HARDWARE
specifies that SAS/GRAPH instruct the device to use the hardware CTS and RTS signals for flow control.

Alias HARD

Note Hardware flow control is not appropriate for some devices.

NONE
specifies that SAS/GRAPH send data without providing flow control. Specify NONE only if the hardware or interface program that you are using provides its own flow control.

SOFTWARE
specifies that SAS/GRAPH use programmed flow control with plotters in eavesdrop mode.

Alias SOFT

XONXOFF
specifies that SAS/GRAPH instruct the device to use ASCII characters DC1 and DC3 for flow control.

Alias X

Note XON/XOFF flow control is not appropriate for some devices.

Details

HANDSHAKE regulates flow of control by specifying how and if a device can signal to the host to temporarily halt transmission and then resume it. Flow control is important because it is possible to send commands to a hard copy device faster than they can be executed.

HANDSHAKE can be used when you are using a protocol converter, interface program, or host computer that can perform XONXOFF or hardware handshaking. You can also use this option if you are routing output through flow-control programs of your own, as in a multiple-machine personal computer environment where the graphics plotter is a shared resource. SAS/GRAPH software sends output to a server (the file transfer does not require flow control). The server queues incoming graphs and sends them to the plotter. The server, rather than SAS/GRAPH software, is responsible for handling flow control. An interface program is usually invoked by the line printer daemon and provides formatting or control signals for a system destination. The interface program typically includes port configuration options, such as baud, parity, and special character processing requirements (raw or cooked mode) for that destination.

If you do not use HANDSHAKE, the value in the driver entry is used.

If you use HANDSHAKE=XONXOFF or HANDSHAKE=HARDWARE, SAS/GRAPH does not actually do the handshaking. It tells the device which type of handshake is being used. The protocol converter, interface program, or host computer actually does the handshake.

Note: If you are creating a graphics stream file using a driver for a plotter and you specify HANDSHAKE=SOFTWARE, the software that you use to send the file to the plotter must be able to perform a software handshake. You will probably want to specify one of the alternative values if you route output to a file.

HBY Graphics Option

Specifies the height of BY lines generated when you use BY-group processing.

Used by:	GOPTIONS statement
Default:	One cell unless the HTEXT= option is used
Restriction:	This option is not supported by Java or ActiveX.
See:	"BY Statement" on page 424

Syntax

HBY=*By line-height* <*units*>

Parameter Values

By line-height <units>

specifies the height of BY line text; by default *By line-height* is 1. If you specify HBY=0, the BY headings are suppressed. For a description of *units*, see "Specifying Units of Measurement" on page 581.

Note: If a value for *units* is not specified, the current units associated with the GUNIT graphics option are used.

Details

When you use a BY statement with a SAS/GRAPH procedure to process a data set in subgroups, each graph produced by that procedure is headed by a BY line that displays the BY variables and their values that define the current subgroup.

HEADER Device Parameter

Specifies the command that executes a user-supplied program to create HEADER records for the driver.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

See: HEADERFILE on page 646

Syntax

HEADER='*command*'

Parameter Values

command
specifies a command that runs a user-written program that creates the file of HEADER records. *Command* is a string up to 40 characters long.

Details

For information about Metagraphics drivers, contact Technical Support.

HEADERFILE Device Parameter

Specifies the fileref for the file from which the Metagraphics driver reads HEADER records.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

See: HEADER on page 646

Syntax

HEADERFILE=*fileref*

Parameter Values

fileref
specifies a valid SAS fileref up to eight characters long. *Fileref* must have been previously assigned with a FILENAME statement or a host command before running the Metagraphics driver. See "FILENAME Statement" on page 52 for details.

Details

For information about Metagraphics drivers, contact Technical Support.

HORIGIN Device Parameter and Graphics Option

Sets the horizontal offset from the lower left corner of the display area to the lower left corner of the graph.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Restriction: This option is not supported by Java or ActiveX.

See: VORIGIN on page 706

Syntax

HORIGIN=*horizontal-offset*<IN | CM | PT>

Parameter Values

***horizontal-offset*<IN | CM | PT>**
specifies the horizontal offset as a nonnegative integer optionally followed by a units specification of IN for inches (default), CM for centimeters, or PT for points. If you do not specify HORIGIN, a default offset is searched for in this order:

1. the left margin specification on an OPTIONS LEFTMARGIN setting
2. HORIGIN setting in the device catalog.

Details

The display area is defined by the XMAX and YMAX device parameters. By default, the origin of the graphics output area is the lower left corner of the display area; the graphics output is offset from the lower left corner of the display area by the values of HORIGIN and VORIGIN. HORIGIN + HSIZE cannot exceed XMAX.

Note: When sending output to the PRINTER destination (ODS PRINTER), if you specify the VSIZE= option without specifying the HSIZE= option, the default origin of

the graphics output area changes. The default placement of the graph changes from the lower left corner of the display area to the top-center of the graphics output area. Likewise, if you specify the HSIZE= option without specifying the VSIZE= option, the graph is positioned at the top-center of the graphics output area by default.

See "The Graphics Output and Device Display Areas" on page 80 for details.

HOSTSPEC Device Parameter

Stores FILENAME statement options in the device entry.

Used by: GDEVICE procedure, GDEVICE Host File Options window

Syntax

HOSTSPEC='*text-string*'

Parameter Values

text-string
specifies FILENAME statement options that are valid for the operating environment. *Text-string* accepts characters in upper or lower case. See the SAS documentation for your operating environment for details.

Details

HOSTSPEC can be used when the driver dynamically allocates a graphics stream file or spool file. It can specify the attributes of the file, such as record format or record length. It cannot be used with Metagraphics drivers.

HPOS Graphics Option

Specifies the number of columns in the graphics output area.

Used by: GOPTIONS statement

Default: device-dependent: the value of the LCOLS or PCOLS device parameter

Restriction: This option is not supported by Java or ActiveX.

See: LCOLS on page 657
PCOLS on page 670
VPOS on page 707

Syntax

HPOS=*columns*

Parameter Values

columns
specifies the number of columns in the graphics output area, which is equivalent to the number of hardware characters that can be displayed horizontally. Specifying HPOS=0 causes the device driver to use the default hardware character cell width for the device.

Details

The HPOS= graphics option overrides the values of the LCOLS or PCOLS device parameters and temporarily sets the number of columns in the graphics output area. HPOS= does not affect the width of the graphics output area but merely divides it into columns. Therefore, you can use HPOS= to control cell width.

The values specified in the HPOS= and VPOS= graphics options determine the size of a character cell for the graphics output area and consequently the size of many graphics elements, such as device–resident text. The larger the size of the HPOS= and VPOS= values, the smaller the size of each character cell.

See "Overview" on page 79 for more information.

HSIZE Device Parameter and Graphics Option

Sets the horizontal size of the graphics output area.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Restriction: This option is partially supported by Java and ActiveX.

See: VSIZE on page 708
XMAX on page 709

Syntax

HSIZE=*horizontal-size*<IN | CM | PT>

Parameter Values

***horizontal-size*<IN | CM | PT>**
specifies the width of the graphics output area as a positive number optionally followed by a units specification of IN for inches (default), CM for centimeters, or

PT for points. If you do not specify HSIZE=, a default size is searched for in this order:

1. the horizontal size is calculated as XMAX − LEFTMARGIN − RIGHTMARGIN. Note that LEFTMARGIN and RIGHTMARGIN are used in the OPTIONS statement.
2. HSIZE setting in the device catalog.

HTEXT Graphics Option

Specifies the default height of the text in the graphics output.

Used by: GOPTIONS statement

Default: One cell

Restriction: This option is partially supported by Java.

Syntax

HTEXT=*text-height* <*units*>

Parameter Values

***text-height* <*units*>**
specifies the height of the text. For a description of *units*, see "Specifying Units of Measurement" on page 581.

Default 1

Note If a value for *units* is not specified, the current units associated with the GUNIT graphics option are used.

Details

HTEXT= is overridden by the HTITLE= graphics option for the *first* TITLE line.

Note: When you use ODS to send graphics to an HTML destination, and titles and footnotes are rendered as part of the HTML body file instead of the graphic image, you must specify the ODS USEGOPT statement for this option to work. See "Using Graphics Options with ODS (USEGOPT)" on page 116 for more information.

HTITLE Graphics Option

Selects the default height used for the first TITLE line.

Used by: GOPTIONS statement

Default: Two cells unless HTEXT= is used

Syntax

HTITLE=*title-height* <*units*>

Parameter Values

title-height <units>
specifies the height of the text in the TITLE1 statement. For a description of *units*, see "Specifying Units of Measurement" on page 581.

Default 2

Note If a value for *units* is not specified, the current units associated with the GUNIT graphics option are used.

Details

If you omit the HTITLE= option, TITLE1 uses the height specified by the HTEXT= graphics option, if used.

Note: When you use ODS to send graphics to an HTML destination, and titles and footnotes are rendered as part of the HTML body file instead of the graphic image, you must specify the ODS USEGOPT statement for this option to work. See "Using Graphics Options with ODS (USEGOPT)" on page 116 for more information.

IBACK Graphics Option

Specifies an image file to display in a graph's background area.

Used by: GOPTIONS statement

Restriction: This option is partially supported by Java.

See: CBACK on page 590
IMAGESTYLE on page 653

Syntax

IBACK=*fileref* | "*external-file*" | "*URL*" | " "

Parameter Values

fileref
specifies a fileref that points to the image file that you want to use. *Fileref* must be a valid SAS fileref up to eight characters long and must have been previously assigned with a FILENAME statement.

external-file
specifies the complete file name of the image file that you want to use. The format of external-file varies across operating environments.

URL
specifies the URL of the image file that you want to use.

" "
suppresses a background image.

Details

The image can be used with any procedures that produce a picture or support the CBACK= option.

This option overrides the BackGroundImage and Image styles attribute in the graph styles. To suppress a background image that is defined in a style or to reset the value of the IBACK= option, specify a blank space:

IBACK=" "

For more information about graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: Procedures Guide*.

For a list of the file types that you use, see "Image File Types Supported by SAS/GRAPH" on page 384.

ID Device Parameter

Specifies the description string used by the Metagraphics driver.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

ID='*description*'

Parameter Values

description
is a character string up to 70 characters long. If this field is blank, the name and description of the graph as specified in the PROC GREPLAY window of the GREPLAY procedure are used.

Details

For information about Metagraphics drivers, contact Technical Support.

IMAGEPRINT, NOIMAGEPRINT Graphics Options

Enables or disables image output

Used by: GOPTIONS statement

Default: IMAGEPRINT

Restriction: This option is not supported by Java or ActiveX.

Syntax

IMAGEPRINT | NOIMAGEPRINT

Parameter Values

IMAGEPRINT
default value specifies that any images are to be included in graphics output.

NOIMAGEPRINT
specifies that images are to be withheld from graphics output.

IMAGESTYLE Graphics Option

Specifies how to display the image file that is specified on the IBACK= option.

Used by: GOPTIONS statement

Default: TILE

Restriction: This option is not supported by Java.

Syntax

IMAGESTYLE=TILE | FIT

Parameter Values

TILE
tile the image within the specified area. This copies the images as many times as needed to fit the area.

FIT
fit the image within the background area. This stretches the image, if necessary.

Details

Note: This option overrides the BackGroundImage and Image styles attribute in the graph styles. For more information about graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: Procedures Guide*.

INTERACTIVE Device Parameter

Sets level of interactivity for Metagraphics driver.

Used by:	GDEVICE procedure, GDEVICE Metagraphics window
Default:	USER
Restriction:	Used only with user-supplied Metagraphics drivers.

Syntax

INTERACTIVE=USER | GRAPH | PROC

Parameter Values

USER
specifies that the user-written part of the driver be executed outside of SAS/GRAPH.

GRAPH
specifies that the user-written part be invoked for each graph.

PROC
specifies that the user-written part of the Metagraphics driver be invoked after the procedure is complete.

Details

For information about Metagraphics drivers, contact Technical Support.

INTERLACED, NONINTERLACED Graphics Options

Specifies whether images are to be displayed as they are received in the browser.

Used by: GOPTIONS statement

Default: NONINTERLACED

Restriction: driver-dependent, GIF series of drivers only

Syntax

INTERLACED | NONINTERLACED

Comparisons

With interlacing it is possible to get a rough picture of what a large image will look like before it is completely drawn in your browser. Your browser might enable you to set an option that will determine how images are displayed.

INTERPOL Graphics Option

Sets the default interpolation value for the SYMBOL statement.

Used by: GOPTIONS statement

Restriction: This option is not supported by Java or ActiveX.

Syntax

INTERPOL=*interpolation-method*

Parameter Values

interpolation-method
specifies the default interpolation to be used when the INTERPOL= option is not specified in the SYMBOL statement. See "SYMBOL Statement" on page 470 for the complete syntax of all interpolation methods.

ITERATION Graphics Option

Specifies the number of times to repeat the GIF or SVG animation loop.

Used by: GOPTIONS statement

Default: 0

Restriction: GIF and SVGANIM drivers only

See: DELAY on page 606
Chapter 16, "Generating Animations," on page 227
"Example: Creating an SVG Animation" on page 245

Syntax

Form 1: For GIF animations:

ITERATION=0 | *iteration-count*

Form 2: For SVG animations:

ITERATION=0 | 1

Parameter Values

0 | *iteration-count*
specifies the number of times that your complete GIF animation loop is repeated. It is assumed that the animation is always played once; this option specifies how many times the animation is repeated. A value of 0 causes the animation to loop continuously. The value of *iteration-count* can be up to 65535.

0 | 1
specifies whether your SVG animation loop is repeated continuously (0) or is played only once (1). When ITERATION=1, your users can use the SVG Viewer reset and play buttons to replay the animation as desired.

Note: If a value other than 0 or 1 is specified, a value of 1 is assumed when the SVG animation is replayed.

When an SVG animation plays, each page is held in view for a period of time. After that time, the page display fades out as the next page fades in. You can use the DELAY= graphics option to change the length of time that each page is held in view.

KEYMAP Graphics Option

Selects the key map to use.

Used by: GOPTIONS statement

Default: installation-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

KEYMAP=*key-map-name* | NONE

Parameter Values

key-map-name
specifies the name of a key map.

NONE
suppresses the key map assigned by default to a non-U.S. keyboard. If you specify KEYMAP=NONE, text might be displayed incorrectly or not at all.

Details

Non-default key maps usually are used only with non-U.S. Keyboards.

LCOLS Device Parameter

Sets the number of columns in the graphics output area for landscape orientation.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

See: HPOS on page 648
LROWS on page 659
PCOLS on page 670

Syntax

LCOLS=*landscape-columns*

Parameter Values

landscape-columns
must be a nonnegative integer in the range 0–999.

Details

Either the LROWS and LCOLS pair of device parameters or the PROWS and PCOLS pair of device parameters are required and must be nonzero.

The HPOS= graphics option overrides the value of LCOLS.

See "Overview" on page 79 for more information.

LFACTOR Device Parameter and Graphics Option

Selects the default hardware line thickness.

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
Default:	device-dependent
Restrictions:	Used only with devices that can draw hardware lines of varying thicknesses. Not supported by Java or ActiveX.

Syntax

LFACTOR=*line-thickness-factor*

Parameter Values

line-thickness-factor
specifies the default hardware line thickness as an integer in the range 0–9999. A value of 0 for LFACTOR is the same as a factor of 1. Lines are drawn *line-thickness-factor* times as thick as normal.

Details

LFACTOR is useful when you are printing graphics output on a plotter. Depending on the orientation and type of device, some plotters might require LFACTOR=10 to get the same thickness of lines as on the display of some devices.

LROWS Device Parameter

Sets the number of rows in the graphics output area for landscape orientation.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

See: LCOLS on page 657
PROWS on page 686
VPOS on page 707

Syntax

LROWS=*landscape-rows*

Parameter Values

landscape-rows
is a nonnegative integer in the range 0–999.

Details

Either the LROWS and LCOLS pair of device parameters or the PROWS and PCOLS pair of device parameters are required and must be nonzero.

The VPOS= graphics option overrides the value of LROWS.

See “Overview” on page 79 for more information.

MAXCOLORS Device Parameter

Sets the total number of colors that can be displayed at once.

Used by: GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

See: PENMOUNTS on page 671

Syntax

MAXCOLORS=*number-of-colors*

Parameter Values

number-of-colors
specifies the number of colors that can be displayed at once as an integer in the range 2–256. The total number of colors includes the foreground colors plus the background color.

Details

The PENMOUNTS= graphics option overrides the value of MAXCOLORS.

MAXPOLY Device Parameter

Sets the maximum number of vertices for hardware-drawn polygons.

Used by: GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Syntax

MAXPOLY=*number-of-vertices*

Parameter Values

number-of-vertices
specifies the maximum number of vertices for hardware-drawn polygons as an integer in the range 0–9999. A value of 0 means that there is no limit to the number of vertices that can be specified in the hardware's polygon-drawing command. The maximum value of MAXPOLY depends on the number of vertices your device can process.

MODEL Device Parameter

Specifies the model number of the output device.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

Syntax

MODEL=*model-number*

Parameter Values

model-number
specifies the model number designated by SAS for the corresponding device as an integer in the range 0–99999. It is not the same as a manufacturer's model number.

Details

Do not change this field in drivers supplied by SAS or in drivers that you copy from drivers supplied by SAS .

MODULE Device Parameter

Specifies the name of the corresponding executable driver module for the device.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

Syntax

MODULE=*driver-module*

Parameter Values

driver-module
specifies the executable driver module name as a literal string up to eight characters long. All standard driver modules begin with the characters SASGD.

Details

Do not change this field in drivers supplied by SAS or in drivers that you copy from drivers supplied by SAS.

NAK Device Parameter

Specifies the negative response for software handshaking for Metagraphics drivers.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

NAK='*negative-handshake-response*'X

Parameter Values

negative-handshake-response
specifies the negative response for software handshaking for Metagraphics drivers as a hexadecimal string up to 16 characters long.

Details

For information about Metagraphics drivers, contact Technical Support.

OFFSHADOW Graphics Option

Controls the width and depth of the drop shadow in legend frames.

Used by: GOPTIONS statement

Default: (0.0625, 0.0625) IN

Restriction: This option is not supported by Java or ActiveX.

Syntax

OFFSHADOW=(*x*<*units*> , *y*<*units*>) | (*x*,*y*<*units*>)

Parameter Values

x*,*y
specify the width (*x*) and depth (*y*) of the drop shadow generated by the LEGEND statement.

Details

If a value for *units* is not specified, the current units associated with the GUNIT graphics option are used. For a description of *units*, see "Specifying Units of Measurement" on page 581.

The values specified by OFFSHADOW= are used with the CSHADOW= and CBLOCK= options in a LEGEND statement. For details, see "LEGEND Statement" on page 432.

PAPERDEST Graphics Option

Specifies which output bin the printer should use if multiple bins are available on the device.

Used by: GOPTIONS statement, OPTIONS statement

Default: 1 (the upper output bin)

Restrictions: This option is hardware-dependent.
PostScript printers require a PPD file.
This option is not supported by Java or ActiveX.

See: PAPERSOURCE on page 666
PPDFILE on page 677
"Options Used in Both GOPTIONS and OPTIONS Statements" on page 582

Syntax

PAPERDEST=*bin*

Parameter Values

***bin* | '*long bin name*'**
specifies the name or number of the output bin. Values for *bin* depend on the type of printer and can be one of the following:

bin
specifies the name or number of the output bin.

Examples PAPERDEST=4

PAPERDEST=BIN2

PAPERDEST=SIDE

'***long bin name***'
specifies the name of the output bin as a character string.

Requirement Names with blanks or special characters must be quoted.

Example PAPERDEST='Top Output Bin'

For PostScript printers, the value for *bin* must correspond to an OutputBin value in the PPD file.

For PCL printers, consult the printer's documentation for valid bin values. If a numeric value exceeds the maximum bin value allowed for the printer, a warning message is issued . For string values, the string is checked against a list of strings that are valid for the driver (for example, 'UPPER', 'LOWER', or 'OPTIONALOUTBIN*n*', where *n* is the bin number). If the string is not valid for the driver, a warning message is issued.

PAPERFEED Device Parameter and Graphics Option

Specifies the increment of paper that is ejected when a graph is completed.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Default: PAPERFEED=0.0 IN

Restrictions: This option is device-dependent.
This option is not supported by Java or ActiveX.

Syntax

PAPERFEED=*feed-increment*<IN | CM>

Parameter Values

***feed-increment*<IN | CM>**
specifies the feed increment as a nonnegative number optionally followed by a units specification of IN for inches (default) or CM for centimeters.

Details

PAPERFEED does not control the total length of the ejection. If you specify PAPERFEED=1, the driver ejects paper in 1-inch increments until the total amount of paper ejected is at least half an inch greater than the size of the graph last printed. If you specify PAPERFEED=8.5 IN, the paper is ejected in increments of 8.5 inches, measuring from the origin of the first graph.

PAPERFEED is provided mainly for plotters that use fanfold or roll paper. If you are using fanfold paper, specify a value for PAPERFEED that is equal to the distance between the perforations.

PAPERLIMIT Graphics Option

Sets the width of the paper used with plotters.

Used by: GOPTIONS statement

Default: maximum dimensions specified in the device driver

Restriction: ZETA plotters and KMW rasterizers

Syntax

PAPERLIMIT=*width*<IN | CM>

Parameter Values

***width*<IN | CM>**
specifies the paper width as a numeric value optionally followed by a units specification of IN for inches (default) or CM for centimeters. If PAPERLIMIT= is not specified, the maximum dimensions of the graph are restricted by the hardware limits of the graphics device.

Details

If you want to use a driver with a device that has a larger plotting area than the device for which the driver is intended (for example, using the ZETA887 driver with a ZETA 836 plotter), the PAPERLIMIT= graphics option can be used to override the size limit of the driver.

PAPERSIZE Graphics Option

Specifies the name of a paper size.

Used by: GOPTIONS statement, OPTIONS statement

Default: device-dependent

Restrictions: This option is hardware-dependent.
PostScript printers require a PPD file.
This option is not supported by Java or ActiveX.

See: PAPERSOURCE on page 666
PPDFILE on page 677
“Options Used in Both GOPTIONS and OPTIONS Statements” on page 582

Syntax

PAPERSIZE='*size-name*'

Parameter Values

size-name
specifies the name of a paper size, such as LETTER, LEGAL, or A4. If you do not specify the PAPERSIZE= option, the PAPERSIZE= option setting in an OPTIONS statement is used. If no OPTIONS statement sets a paper size, the value for paper size is device-dependent:

- The universal printing devices use the size specified in the Page Setup dialog box.
- All other printer devices use the LETTER paper size.

Details

Typically, you might use the PAPERSIZE= option with the Output Delivery System (ODS). For some printers, the PAPERSIZE= option overrides the PAPERSOURCE= option selection.

For PostScript devices, the name must match the name of a paper size in the PPD file. Refer to the PPD file for a list of valid names. *Size-name* is case-insensitive and can contain a subset of the full name. For example, if the name in the PPD file is *PageSize A4/A4, you can specify PAPERSIZE='A4'. If a PPD file is not specified, the PAPERSIZE= option is ignored.

For PCL devices, the device driver searches the SAS Registry for supported paper size values. To see the supported list of sizes, submit the following statements:

```
proc registry listhelp
     startat='options\papersize';
run;
```

For more information about the SAS Registry, refer to the SAS Help facility.

PAPERSOURCE Graphics Option

Specifies which paper tray the printer should use if multiple trays are available on the device.

Used by:	GOPTIONS statement, OPTIONS statement
Default:	device-dependent
Restrictions:	This option is hardware-dependent. PostScript printers require a PPD file. This option is not supported by Java or ActiveX.
See:	PAPERDEST on page 663 PAPERSIZE on page 665 PPDFILE on page 677

"Options Used in Both GOPTIONS and OPTIONS Statements" on page 582

Syntax

PAPERSOURCE=*tray*

Parameter Values

tray
specifies the name or number of the paper tray. Values for *tray* depend on the type of printer and can be one of the following:

tray
specifies the name or number of the paper tray.

Examples PAPERSOURCE=3

PAPERSOURCE=TRAY3

PAPERSOURCE=Upper

'***long tray name***'
specifies the name of the paper tray as a character string.

Requirement Names with blanks or special characters must be quoted.

Example PAPERSOURCE='Optional Output Tray'

Details

On some printers, if the PAPERSIZE= option is also specified, it overrides the setting on the PAPERSOURCE= option.

For PostScript printers, a tray number, such as PAPERSOURCE='tray3', must correspond to an InputSlot value in the PPD file.

For PCL printers, consult the printer's documentation for valid tray values. If a numeric value exceeds the maximum tray value allowed for the printer, a warning message is issued . For string values, the string is checked against a list of strings that are valid for the driver:

- 'AUTO'
- 'HCI' or 'HCI*n*', where *n* is a number from 2 to 21
- 'MANUAL'
- 'MANUAL_ENVELOPE'
- 'TRAY*n*', where *n* is 1, 2, or 3.

If the string is not valid for the driver, a warning message is issued.

PAPERTYPE Graphics Option

Specifies the name of a paper type.

Used by: GOPTIONS statement, OPTIONS statement

Default: PLAIN

Restrictions: This option is hardware-dependent.
PostScript printers require a PPD file.
This option is not supported by Java or ActiveX.

See: PPDFILE on page 677
"Options Used in Both GOPTIONS and OPTIONS Statements" on page 582

Syntax

PAPERTYPE='*type-name*'

Parameter Values

type-name
specifies the name of a paper type. Valid values depend on the type of printer.

For PostScript devices, *type-name* must match the name of a paper type in the PPD file, such as TRANSPARENCY or PLAIN. Refer to the PPD file for a list of valid names. *Type-name* is case-insensitive and can contain a subset of the full name. For example, if the name in the PPD file is *MediaType Plain/Paper, you can specify PAPERTYPE='PLAIN/PAPER'.

For PCL devices, *type-name* specifies the name of a paper type that is available on the current printer, such as GLOSSY, PLAIN, SPECIAL, or TRANSPARENCY. Consult your printer's user manual for the complete list of available paper types on your printer.

Details

For PostScript devices, if a PPD file is not specified, the PAPERTYPE= option is ignored.

PATH Device Parameter

Sets the increment of the angle for device-resident text rotation.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Default: PATH=0

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

PATH=*angle-increment*

Parameter Values

angle-increment
specifies the angle at which to rotate the text baseline as an integer in the range 0–360. A value of 0 means that the device uses its default orientation.

Note Specify 0 if your device does not perform string angling in hardware.

Details

For information about Metagraphics drivers, contact Technical Support.

PCLIP, NOPCLIP Graphics Options

Specifies whether a clipped polygon is stored in its clipped or unclipped form.

Used by: GOPTIONS statement

Default: NOPCLIP

Restriction: This option is not supported by Java or ActiveX.

See: POLYGONCLIP on page 674

Syntax

PCLIP | NOPCLIP

Parameter Values

PCLIP
stores clipped polygons with the graph in the default catalog WORK.GSEG, or in the catalog that you specify.

NOPCLIP
stores the unclipped form of the polygon and causes the polygon to be clipped when replayed.

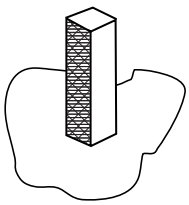
Details

The effects of this option are seen only when you use the graphics editor to edit a graph.

When a procedure produces a graph with intersecting polygons or blanking areas, it clips portions of the polygons to prevent the ones behind from showing through. When the graph is created and stored in a catalog, if PCLIP is in effect, the clipped form of the polygon is stored with it. If NOPCLIP is specified, the complete polygon is stored in the catalog and the graph is clipped each time it is replayed.

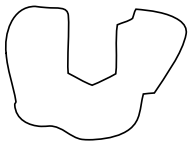
For example, suppose you create a block map like the one in Figure 26.63 on page 670.

Figure 26.3 *Intersecting Polygons*



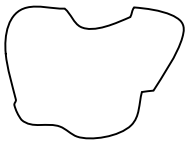
The block clips the boundary of the map area polygon. If you specify PCLIP, the map area polygon is stored in its clipped form, as shown in Figure 26.64 on page 670.

Figure 26.4 *Clipped Polygon with PCLIP Option*



NOPCLIP stores the map area in its unclipped form, as shown in Figure 26.65 on page 670.

Figure 26.5 *Clipped Polygon with NOPCLIP Option*



In this case, when the graph is recalled from the catalog, the map area polygon must be clipped before it is displayed with the block. If you plan to edit the graph with the graphics editor, specify NOPCLIP so that polygons retain their original form.

PCOLS Device Parameter

Sets the number of columns in the graphics output area for portrait orientation.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

See: HPOS on page 648
LCOLS on page 657
PROWS on page 686

Syntax

PCOLS=*portrait-columns*

Parameter Values

portrait-columns
specifies the number of columns in the graphics output area for portrait orientation as an integer in the range 0–999.

Details

Either the LROWS and LCOLS pair of device parameters or the PROWS and PCOLS pair of device parameters are required and must be nonzero.

The HPOS= graphics option overrides the value of PCOLS.

See "Overview" on page 79 for more information.

PENMOUNTS Graphics Option

Specifies the number of active pens or colors.

Used by: GOPTIONS statement

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

See: MAXCOLORS on page 659

Syntax

PENMOUNTS=*active-pen-mounts*

Parameter Values

active-pen-mounts
specifies the number of pens for a plotter with multiple pens. After the specified number of pens have been used, you are prompted to change the pens.

Details

For devices that are not pen plotters, PENMOUNTS= can be used to indicate the number of colors that can be displayed at one time. In this case, PENMOUNTS= performs the same function as the MAXCOLORS device parameter except that the value specified for MAXCOLORS includes the background color and PENMOUNTS refers to foreground colors only. Thus, PENMOUNTS=4 implies MAXCOLORS=5.

PENMOUNTS= overrides the value of the MAXCOLORS device parameter. You can specify MAXCOLORS= in a GOPTIONS statement as a synonym for PENMOUNTS=.

PENSORT, NOPENSORT Device Parameters and Graphics Options

Specifies whether plotters draw graphics elements in order of color.

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
Default:	device-dependent
Restriction:	This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

PENSORT | NOPENSORT

Form 2: GDEVICE:

PENSORT=Y | N

Parameter Values

PENSORT
PENSORT=Y
causes the plotter to draw all graphics elements of one color at one time. For example, it draws all the red elements in the output, then all the blue elements, and so on. This specification is compatible with previous releases. Use it for plotters with real pens.

NOPENSORT
PENSORT=N
causes the plotter to draw each element as it is encountered, regardless of its color. For example, the plotter might draw a red circle, then a blue line, and then a red line, and so on. This method is best for electrostatic printers implemented with Metagraphics drivers of TYPE=PLOTTER. In addition, NOPENSORT enables you to specify nonstandard color names.

PIEFILL, NOPIEFILL Device Parameters and Graphics Options

Specifies whether to use the device's hardware pie-fill capability.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

PIEFILL | NOPIEFILL

Form 2: GDEVICE:

PIEFILL=Y | N

Parameter Values

PIEFILL
PIEFILL=Y
causes SAS/GRAPH to use the built-in hardware capability of the device, if available, to fill pies and pie sections. A blank `Piefill` field in the Parameters window is the same as PIEFILL=Y.

Tip Hardware drawing is faster, but not all devices have the capability. SAS/GRAPH does not try to use the capability if your device does not support it.

NOPIEFILL
PIEFILL=N
causes SAS/GRAPH to fill pies and pie sections using software pie fills.

POLYGONCLIP, NOPOLYGONCLIP Graphics Options

Specifies the type of clipping used when two polygons overlap.

Used by: GOPTIONS statement

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

See: PCLIP on page 669

Syntax

POLYGONCLIP | NOPOLYGONCLIP

Parameter Values

POLYGONCLIP
specifies polygon clipping, which enables a clipped polygon to be filled with a hardware pattern. POLYGONCLIP affects only graphs that have blanking areas or intersecting polygons.

NOPOLYGONCLIP
specifies line clipping; a polygon that has been line-clipped cannot use a hardware pattern.

Details

Clipping is the process of removing part of one polygon when two polygons intersect. For example, in a block map, a block might overlap the boundary of its map area. In this case, the polygon that makes up the map area is clipped so that you do not see the boundary line behind the block. (See Figure 26.63 on page 670 for an illustration of a clipped polygon.) The type of clipping used by a graph affects whether a clipped area can use hardware patterns.

POLYGONCLIP is affected by the PCLIP graphics option:

POLYGONCLIP with PCLIP or NOPCLIP
all areas can use hardware patterns

NOPOLYGONCLIP with NOPCLIP
all areas use only software patterns

NOPOLYGONCLIP with PCLIP
areas can use either hardware or software patterns depending on the nature of the clipped polygons.

Under some conditions the polygons might not be clipped correctly. Specifying both POLYGONCLIP and NOPCLIP will produce the correct graph.

POLYGONFILL, NOPOLYGONFILL Device Parameters and Graphics Options

Specifies whether to use the hardware polygon-fill capability.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

POLYGONFILL | NOPOLYGONFILL

Form 2: GDEVICE:

POLYFILL=Y | N

Parameter Values

POLYGONFILL
POLYFILL=Y
causes SAS/GRAPH to use the built-in hardware capability of the device to fill polygons. A blank `Polyfill` field in the Parameters window is the same as POLYGONFILL.

Tip Hardware drawing is faster, but not all devices have the capability. SAS/GRAPH does not try to use the capability if your device does not support it.

NOPOLYGONFILL
POLYFILL=N
causes SAS/GRAPH to use software fills to fill polygons.

POSTGEPILOG Graphics Option

Specifies data to send immediately after the data that is stored in the Gepilog field of the device entry is sent.

Used by: GOPTIONS statement

Default: Null string

Restriction: This option is not supported by Java or ActiveX.

See: GEPILOG on page 632
PREGEPILOG on page 678

Syntax

POSTGEPILOG='*string*'

Parameter Values

'*string*'
specifies the data to send immediately after the data that is stored in the Gepilog field of the device entry is sent. *String* can be either of the following:

- '*hex-string*'X
- '*character-string*'

PROC GOPTIONS always reports the value as a hexadecimal string.

POSTGPROLOG Graphics Option

Specifies the data to send immediately after the data that is stored in the Gprolog field of the device entry is sent.

Used by: GOPTIONS statement

Default: Null string

Restriction: This option is not supported by Java or ActiveX.

See: GPROLOG on page 634
PREGPROLOG on page 679

Syntax

POSTGPROLOG='*string*'

Parameter Values

'*string*'
specifies the data to send immediately after the data that is stored in the Gprolog field of the device entry is sent. *String* can be either of the following:

- '*hex-string*'X
- '*character-string*'

PROC GOPTIONS always reports the value as a hexadecimal string.

POSTGRAPH1, POSTGRAPH2 Device Parameters and Graphics Options

Specifies host commands to be executed after the graph is produced.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host Commands window

Restriction: This option is not supported by Java or ActiveX.

See: FILECLOSE on page 621

Syntax

POSTGRAPH1='*system-command(s)*'
POSTGRAPH2='*system-command(s)*'

Parameter Values

system-command(s)
specifies one or more valid system commands. The string can contain upper- or lowercase characters. Separate multiple commands with a command delimiter, which is host-specific; for example, some operating environments use a semicolon. The total length of the string cannot exceed 72 characters. The commands are executed right after the graph is produced.

Details

If you want to use a host command to send output to the device after each graph executes, use the POSTGRAPH parameter with FILECLOSE=GRAPHEND.

PPDFILE Graphics Option

Specifies the location of an external file containing PostScript Printer Description (PPD) information.

Used by: GOPTIONS statement

Restriction: PostScript printers only

See: BINDING on page 589
COLLATE on page 597
DUPLEX on page 617
PAPERDEST on page 663
PAPERSIZE on page 665

PAPERSOURCE on page 666
PAPERTYPE on page 668
REVERSE on page 691

Syntax

PPDFILE=*fileref* | '*external-file*'

Parameter Values

fileref
specifies a fileref that points to the PPD file that you want to use. *Fileref* must be a valid SAS fileref up to eight characters long and must have been previously assigned with a FILENAME statement.

external-file
specifies the complete file name of the PPD file that you want to use. The format of *external-file* varies across operating environments. For details, see the SAS documentation for your operating environment.

Details

A PostScript Printer Description (PPD) file is a text file that contains commands required to access features of the device. These files are available from Adobe. Also, many printer manufacturers provide the appropriate PPD file for their PostScript printers.

PREGEPILOG Graphics Option

Specifies data to send immediately before the data that is stored in the Gepilog field of the device entry is sent.

Used by:	GOPTIONS statement
Default:	Null string
Restriction:	This option is not supported by Java or ActiveX.
See:	GEPILOG on page 632 POSTGEPILOG on page 675

Syntax

PREGEPILOG='*string*'

Parameter Values

'*string*'
specifies data to send immediately before the data that is stored in the Gepilog field of the device entry is sent. *String* can be either of the following:

- '*hex-string*'X
- '*character-string*'

PROC GOPTIONS always reports the value as a hexadecimal string.

PREGPROLOG Graphics Option

Specifies the data to send immediately before the data that is stored in the Gprolog field of the device entry is sent.

Used by: GOPTIONS statement

Default: Null string

Restriction: This option is not supported by Java or ActiveX.

See: GPROLOG on page 634
POSTGPROLOG on page 676

Syntax

PREGPROLOG='*string*'

Parameter Values

'*string*'
specifies the data to send immediately before the data that is stored in the Gprolog field of the device entry is sent. *String* can be either of the following:

- '*hex-string*'X
- '*character-string*'

PROC GOPTIONS always reports the value as a hexadecimal string.

PREGRAPH1, PREGRAPH2 Device Parameters and Graphics Options

Specifies host commands to be executed before the graph is produced.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host Commands window

Restriction: This option is not supported by Java or ActiveX.

See: FILECLOSE on page 621

Syntax

PREGRAPH1='*system-command(s)*'
PREGRAPH2='*system-command(s)*'

Parameter Values

system-command(s)
specifies one or more valid system commands. The string can contain upper- or lowercase characters. Separate multiple commands with a command delimiter, which is host-specific; for example, some operating environments use a semicolon. The total length of the string cannot exceed 72 characters. The commands are executed immediately before the graph is produced.

Details

The PREGRAPH parameter should be used with FILECLOSE=GRAPHEND.

PROCESS Device Parameter

Specifies the command that translates the metafile into commands for the device.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

See: INTERACTIVE on page 654

Syntax

PROCESS='*command*'

Parameter Values

command
specifies a character string of up to 40 characters that contains the command that translates the metafile produced by the Metagraphics driver into commands for the device. The command runs your program to produce the output.

Restriction *Command* cannot exceed 40 characters.

Details

PROCESS is required if the value of the INTERACTIVE device parameter is PROC or GRAPH.

For information about Metagraphics drivers, contact Technical Support.

PROCESSINPUT Device Parameter

Specifies the fileref for the file that contains input for the user-written part of the Metagraphics driver.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

PROCESSINPUT=*fileref*

Parameter Values

fileref
specifies a valid SAS fileref up to eight characters long. *Fileref* must be assigned with a FILENAME statement or a host command before running the Metagraphics driver. See "FILENAME Statement" on page 52 *SAS/GRAPH: Reference* for additional information.

Details

For information about Metagraphics drivers, contact Technical Support.

PROCESSOUTPUT Device Parameter

Specifies the fileref for the file that receives output from the user-written part of the Metagraphics driver.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

PROCESSOUTPUT=*fileref*

Parameter Values

fileref
specifies a valid SAS fileref up to eight characters long. *Fileref* must be assigned with a FILENAME statement or a host command before running the Metagraphics driver. See "FILENAME Statement" on page 52 for additional information.

Details

For information about Metagraphics drivers, contact Technical Support.

PROMPT, NOPROMPT Device Parameters and Graphics Options

Specifies whether prompts are issued.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

PROMPT | NOPROMPT

Form 2: GDEVICE:

PROMPT=0–7

Parameter Values

PROMPT
causes all prompts to be displayed.

NOPROMPT
suppresses all prompts. NOPROMPT overrides the GWAIT= graphics option.

PROMPT=0–7
specifies the level of prompting. The levels are as follows:

0
provides no prompting.

1
issues start-up messages only. Start-up messages are messages such as PLEASE PRESS RETURN TO CONTINUE.

2
signals end of graph if device is a video display or sends message to change paper if device is a plotter.

3
combines the effects of 1 and 2.

4
sends a message to mount pens if the device is a plotter.

5
combines the effects of 4 and 1.

6
combines the effects of 4 and 2.

7
sends all prompts.

Valid in GDEVICE procedure statement

Note If you specify either 0 for the PROMPT device parameter or NOPROMPT in a GOPTIONS statement for a display device, the display clears immediately after the graph is drawn.

Details

In the GDEVICE Parameters window, the PROMPT parameter consists of four fields that describe the type of prompt:

start up
issues a message to turn the device on (if the device is a hard-copy device) or the message PLEASE PRESS RETURN AFTER EACH BELL TO CONTINUE.

end of graph
signals, usually by a bell, when the graph is complete (valid for video displays only).

mount pens
issues a message to mount pens in a certain order and (for certain devices only) to ask for pen priming strokes for plotters.

change paper
prompts the user to change the paper (valid for plotters only).

Enter an X for each prompt that you want to be given. If no Xs appear in these fields, no prompt messages are issued, and the device does not wait for you to respond between graphs.

PROMPTCHARS Device Parameter and Graphics Option

Selects the prompt characters to be used by SAS/GRAPH device drivers.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: host-dependent

Restriction: This option is not supported by Java or ActiveX.

See: GSFLEN on page 637
HANDSHAKE on page 643

Syntax

PROMPTCHARS='*prompt-chars-hex-string*'X

Parameter Values

prompt-chars-hex-string

specifies the prompt characters to be used by SAS/GRAPH device drivers as an 8-byte hexadecimal string that is specified as 16 hexadecimal characters. In GDEVICE procedure statements, enclose the string in single quotation marks, followed by an X. In the Parameters window, enter the hexadecimal string without either quotation marks or a trailing X.

Note: Bytes 1, 4, and 5 are the safest for you to change because you are most likely to know the correct value for them. Check with Technical Support before changing any of the other bytes.

The following table lists and describes each byte in the string.

Byte	Description
byte 1	is the ASCII code of the system prompt character (for software handshaking). The system prompt character is the last character that the host sends before waiting for a response from the plotter. For example, 11 means the host sends an XON or DC1 character as a prompt. If the host does not send a special character for a prompt, set this byte to 00.
byte 2	is the ASCII code of the echo-terminator character (for software handshaking). This character is sent at the beginning of each record.
byte 3	prevents splitting commands across records if the value is 01. If you are creating a graphics stream file to send to a device at a later time, and there is the possibility that extra characters will be added between records during transmission, setting the third byte to 01 reduces the likelihood that the extra characters are interpreted as graphics commands and cause stray lines or other device characters. If the third byte is set to 00, the driver makes the records as long as possible and splits device commands across records if necessary. Setting the third byte to 00 is more efficient but is more likely to

Byte	Description
	result in device errors if output is written to a file and later transmitted to the device.
byte 4	is the line-end character (for software handshaking). It indicates that more data can be sent. This character is almost always a carriage-return character, 0D.
byte 5	specifies turnaround delay in tenths of a second (for software handshaking). The turnaround delay is the amount of time the device waits after receiving the prompt character before sending the line-end character. For example, a value of 05 represents a half-second delay.
byte 6	sets default record length using a hexadecimal value 00–FF. This byte sets the length of the records sent to the device or to a file. If this byte is set to 00 (the default), SAS/GRAPH uses the longest record length possible for the device. To specify an alternate length, set the sixth byte to the hexadecimal value for the desired length. For example, to generate records of length 80, specify 50 for the sixth byte. If the GSFLEN device parameter or graphics option is specified, its value overrides the value of the sixth prompt character. Some values of the GPROTOCOL device parameter cause each byte in the data stream to be expanded to two bytes. This expansion is done after the length of the record is set by PROMPTCHARS. If you are specifying a value for GPROTOCOL that does this (for example, SASGPHEX, SASGPLCL, or SASGPAGL), specify a value for the sixth byte of PROMPTCHARS that is half of the actual record length desired. For example, a hexadecimal value of 40 (64 decimal) produces a 128-byte record after expansion by the GPROTOCOL module.
bytes 7 and 8	are unused and should be set to 0000.

Details

PROMPTCHARS is most commonly used to specify parameters used in software handshaking (see “HANDSHAKE Device Parameter and Graphics Option” on page 643), but it can also be used to control the length of records written by most drivers. You can also use the GSFLEN= graphics option for this purpose. PROMPTCHARS can be used to control the length of records written by most drivers. You can also use the GSFLEN= graphics option for this purpose.

PROWS Device Parameter

Sets the number of rows in the graphics output area for portrait orientation.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

See: LROWS on page 659
PCOLS on page 670
VPOS on page 707

Syntax

PROWS=*portrait-rows*

Parameter Values

portrait-rows
specifies the number of rows in the graphics output area for portrait orientation as an integer in the range 0–999.

Details

Either the LROWS and LCOLS pair of device parameters or the PROWS and PCOLS pair of device parameters are required and must be nonzero.

The VPOS= graphics option overrides the value of PROWS.

See “Overview” on page 79 for more information.

QMSG, NOQMSG Device Parameters

Specifies whether log messages are held until after the graphics output is displayed.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

Syntax

Form 1: GOPTIONS:

QMSG | NOQMSG

Form 2: GDEVICE:

QMSG=Y | N

Parameter Values

QMSG
QMSG=Y
queues driver messages while the device is in graphics mode (default for video devices).

NOQMSG
QMSG=N
prevents the queuing of messages (default for plotters, cameras, and printers).

Details

Message queuing is desirable on display devices that do not have a separate dialog box and graphics area. If messages are not queued, they are written to the log as the graphics output is being generated. This behavior can cause problems on some devices.

A blank `Queued messages` field in the Parameters window can mean either Y or N, depending on the device.

RECTFILL Device Parameter

Specifies which rectangle fills should be performed by hardware.

Used by: GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

See: FILL on page 623

Syntax

RECTFILL='*rectangle-fill-hex-string*'X

Parameter Values

rectangle-fill-hex-string
specifies which rectangle fills should be performed by hardware as a hexadecimal string that is 16 characters long. In GDEVICE procedure statements, enclose the string in single quotation marks, followed by an X. In the Parameters window, enter the hexadecimal string without either quotation marks or a trailing X.

The following table shows which bit position (left-to-right) within the hexadecimal string controls each fill pattern.

Table 26.2 Bit Positions That Control Fill Patterns

Bit	Fill pattern	Bit	Fill pattern
1	R1	9	L4
2	R2	10	L5
3	R3	11	X1
4	R4	12	X2
5	R5	13	X3
6	L1	14	X4
7	L2	15	X5
8	L3	16	S

For example, if you want the driver to use only the L1 and R1 fills in hardware, the first and sixth bits of the first byte of the hexadecimal string should be turned on, which corresponds to a value of '8400000000000000'X ('84'X is equivalent to '1 0 0 0 0 1 0 0' in binary). If a particular hardware rectangle fill is not available or not to be used (as indicated by the value of RECTFILL), the fill is generated by the software.

See "PATTERN Statement" on page 455 for an illustration of the fill patterns.

Details

Note: Not all devices support this capability. If FILL=N is specified or the NOFILL option is used in a GOPTIONS statement, RECTFILL is ignored.

RENDER Graphics Option

Controls the creation and disposition of rendered Bitstream fonts.

Used by: GOPTIONS statement

Default: MEMORY

Restriction: This option is not supported by Java or ActiveX.

See: RENDERLIB on page 689

Syntax

RENDER=APPEND | DISK | MEMORY | NONE | READ

Parameter Values

APPEND
creates files to store rendered versions of Bitstream fonts if the files do not already exist, reads previously rendered characters from the font files, and appends rendered versions of new characters to the font files when the SAS/GRAPH procedure terminates.

DISK
creates files to store rendered versions of Bitstream fonts if the files do not already exist, reads previously rendered characters from the font files, and appends rendered versions of new characters to the font files as they are encountered. This method is slower on some hosts, but it can work in memory-constrained conditions where the other rendering methods fail.

MEMORY
renders all fonts in memory without creating any font files on disk. Font files are not used even if they already exist. New characters are not written to existing font files when SAS/GRAPH procedures terminate.

This is the default and should be the fastest method on hosts that support virtual memory.

NONE
disables the font rendering features.

READ
reads existing rendered font files but does not create new font files or write new characters to existing font files. This is useful only when font files already exist in the rendered font library.

Details

The memory capacity and input/output characteristics of your host system determine which value for the RENDER= option provides the best performance.

RENDERLIB Graphics Option

Specifies the SAS library in which rendered font files are stored.

Used by:	GOPTIONS statement
Default:	WORK
Restriction:	This option is not supported by Java or ActiveX.
See:	RENDER on page 688

Syntax

RENDERLIB=*libref*

Parameter Values

libref
specifies a previously defined libref that identifies the SAS library. The default library is WORK. See "LIBNAME Statement" on page 52 for more information about assigning a libref.

REPAINT Device Parameter and Graphics Option

Specifies how many times to redraw the graph.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

REPAINT=*redraw-factor*

Parameter Values

redraw-factor
specifies how many times to redraw the graph as a integer in the range 0–999.

Details

Use this option with printers that produce light images after only one pass. This option also is useful for producing transparencies; multiple passes make the colors more solid or more intense.

Not all devices have this capability.

RESET Graphics Option

Resets graphics options to their default values, cancels global statements, or both.

Used by: GOPTIONS statement

Example: "Example 9: Creating a Web Page with Drill-Down Functionality Using the ODS HTML Statement" on page 568

Syntax

RESET=ALL | GLOBAL | *statement-name* | (*statement-name(s)*)

Parameter Values

ALL
sets all graphics options to defaults and cancels all global statements.

GLOBAL
cancels all global statements (AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, and TITLE). Options in the GOPTIONS statement are unaffected.

statement-name
resets or cancels only the specified global statements. For example, RESET=PATTERN cancels all PATTERN statements only. To cancel several statements at one time, enclose the statement names in parentheses. For example, RESET=(TITLE FOOTNOTE AXIS) cancels all TITLE, FOOTNOTE, and AXIS statements.

Note RESET=GOPTIONS sets all graphics options to defaults but does not cancel any global statements.

Details

RESET=ALL or RESET=GOPTIONS must be the first option specified in the GOPTIONS statement. Otherwise, the graphics options that precede the RESET= option in the GOPTIONS statement are reset. Other options can follow the RESET= graphics option in the statement.

REVERSE, NOREVERSE Graphics Options

Specifies whether to print the output in reverse order, if reverse printing is supported by the device.

Used by: GOPTIONS statement

Default: NOREVERSE

Restrictions: This option is hardware-dependent.
PostScript printers require a PPD file.
This option is not supported by Java or ActiveX.

See: PPDFILE on page 677

Syntax

REVERSE | NOREVERSE

Comparisons

The purpose of REVERSE is to control the stacking order of printer output, depending on how the printer stacks paper. On some printers, reverse implies using the alternate output bin (back of the printer).

For PCL devices, REVERSE sends output to the LOWER out bin, which is the face-up output bin.

For PostScript devices, if the PPD file has an “OutputOrder” entry and one of its entries is “Reverse,” the device supports reverse order printing and the appropriate PostScript code to activate reverse will be used. If the PPD file does not have an “OutputOrder” entry but does have a “PageStackOrder” entry and corresponding OutputBin value, then reverse order printing is supported indirectly, using the PPD file's PageStackOrder or OutputBin entries.

Note: Some PostScript devices implement Reverse as the default output mode for one of the output bins. In this case, selecting either the “reverse” output bin or specifying REVERSE mode produces identical results.

ROTATE, NOROTATE Device Parameters and Graphics Options

Specifies whether and how to rotate the graph.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Restrictions: This option is supported by printer-type Universal Printer devices only, which include PDF, POSTSCRIPT, and PCL.

This option is not supported by the SVG, SVGZ, SVGVIEW, SVGT, and SVGANIM devices.

This option is partially supported by the display devices.

Syntax

Form 1: GOPTIONS:

ROTATE | NOROTATE

ROTATE=LANDSCAPE | PORTRAIT

Form 2: GDEVICE:

ROTATE=LANDSCAPE | PORTRAIT

Parameter Values

ROTATE | NOROTATE
specifies whether to rotate the graph 90 degrees from its default orientation.

ROTATE=LANDSCAPE
specifies landscape orientation (the graph is wider than it is high).

ROTATE=PORTRAIT
specifies portrait orientation (the graph is higher than it is wide).

Details

If you do not specify a rotation, a default orientation is searched for in this order:

1. the ORIENTATION setting in an OPTIONS statement
2. the device-dependent default

By default, many of the graphics output devices such as PNG and GIF use the landscape orientation while the printer devices use the portrait orientation. When you direct the output of a landscape device to printer device, the graph is scaled to fit the portrait dimensions, which might not be desirable. You can use the ROTATE or ROTATE=LANDSCAPE graphics option to rotate the graph 90 degrees on the portrait-oriented printer easel in order to maintain the original dimensions of the graph. For example:

```
goptions reset=all rotate=landscape;
```

However, because the graph is being rendered by devices with different orientations in that case, unexpected results might occur. When directing landscape output to a printer device, the best approach is to use the ORIENTATION=LANDSCAPE system option rather than the ROTATE or ROTATE=LANDSCAPE graphics option. For example:

```
options orientation=landscape;
```

The ORIENTATION=LANDSCAPE system option sets the orientation of all of the output devices to landscape, which eliminates any problems associated with different device orientations.

See "ORIENTATION= System Option" in *SAS System Options: Reference* for information about the ORIENTATION= system option.

ROTATION Device Parameter

Sets the increment of the angle by which the device can rotate any given letter in a string of text in a Metagraphics driver.

Used by:	GDEVICE procedure, GDEVICE Metagraphics window
Default:	ROTATION=0
Restriction:	Used only with user-supplied Metagraphics drivers.

Syntax

ROTATION=*angle-increment*

Parameter Values

angle-increment
specifies the increment of the angle at which to rotate individual characters (for example, every 5 degrees, every 45 degrees, and so on). *Angle-increment* is an integer in the range 0–360. A value of 0 means that the device uses its default character rotation. Specify 0 if your device does not perform hardware character rotation.

Details

For information about Metagraphics drivers, contact Technical Support.

ROWS Device Parameter

Specifies the number of rows the device-resident font uses in graphics output.

Used by: GDEVICE Chartype window, GDEVICE procedure, CHARREC= option

Default: 0

See: CHARREC on page 594

Details

For information about the syntax, see “CHARREC Device Parameter” on page 594.

If you are using a device driver from SASHELP.DEVICES, this parameter already is set for device–resident fonts that have been defined for your installation. For scalable fonts, you can specify 1 for ROWS, and the actual number of rows are computed based on the current text width. If you are adding to or modifying device-resident fonts available for a particular device driver, specify a positive value for the ROWS device parameter. If ROWS is greater than 0, it overrides the values of the LROWS and PROWS device parameters.

SCALABLE Device Parameter

Specifies whether a font is scalable.

Used by: GDEVICE Chartype window, GDEVICE procedure, CHARREC= option

Default: device-dependent

See: CHARTYPE on page 595

Details

The SCALABLE option is set in the CHARREC= option in the GDEVICE procedure statement. See "CHARREC Device Parameter" on page 594 for information about the CHARREC= option syntax.

A device-resident font is scalable if it can be used with any combination of rows and columns. Use the SCALABLE device parameter if you are adding to or modifying the fonts available for a particular device driver. If you are using a device driver from SASHELP.DEVICES, this parameter already is set for device-resident fonts that have been defined for your installation.

SIMFONT Graphics Option

Specifies a SAS/GRAPH font to use if the default device-resident font cannot be used.

Used by: GOPTIONS statement

Default: SIMULATE

Restriction: This option is not supported by Java or ActiveX.

Syntax

SIMFONT=*SAS/GRAPH-font*

Parameter Values

SAS/GRAPH-font
specifies a SAS/GRAPH font to use instead of the default device-resident font.

Default SIMULATE font, which is stored in the SASHELP.FONTS catalog

Details

SAS/GRAPH substitutes the SAS/GRAPH font specified by the SIMFONT= option for the default device-resident font in these cases:

- when you use the NOCHARACTERS option in a GOPTIONS statement
- when you specify a non-default value for the HPOS= or VPOS= graphics option and your device does not have scalable hardware characters
- when you replay a graph using a device driver other than the one used to create the graph

- when you specify an angle or rotation for your hardware text that the device is not capable of producing
- when you specify a device-resident font that is not supported by your device.

See Chapter 22, “Specifying Fonts in SAS/GRAPH Programs,” on page 349 for details.

SPEED Device Parameter and Graphics Option

Selects pen speed for plotters with variable speed selection.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

SPEED=*pen-speed*

Parameter Values

pen-speed
specifies a percentage (1 through 100) of the maximum pen speed for the device. For example, SPEED=50 slows the drawing speed by half. In general, slowing the drawing speed produces better results.

By default, the value of SPEED is the normal speed for the device.

SWAP, NOSWAP Device Parameters and Graphics Options

Specifies whether to reverse BLACK and WHITE in the graphics output.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Defaults: GOPTIONS: NOSWAP
GDEVICE: SWAP=N

Restriction: This option is not supported by Java or ActiveX.

Syntax

Form 1: GOPTIONS:

SWAP | NOSWAP

Form 2: GDEVICE:

SWAP=Y | N

Parameter Values

SWAP
SWAP=Y
swaps BLACK for WHITE and vice versa.

NOSWAP
SWAP=N
does not swap the colors. A blank `Swap` field in the Parameters window is the same as SWAP=N.

Details

SWAP does not affect the background color and affects only BLACK and WHITE foreground colors specified as predefined SAS color names. SWAP ignores BLACK and WHITE specified in HLS, RGB, or gray-scale format. This option is useful when you want to preview a graph on a video device and send the final copy to a printer that uses a white background.

```
goptions reset=all cback=blue ctitle=black swap;
title1 h=8 'swap test';
title2 h=8 'another title';
proc gslide border;
run;
```

SWFONTRENDER Graphics Option

Specifies the method used to render system fonts.

Used by: GOPTIONS statement

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

SWFONTRENDER=POLYGON | SCANLINE

Parameter Values

POLYGON
uses polygon rendering

SCANLINE
uses scanline rendering

Details

SWFONTRENDER determines the method used to render system text to a vector graphics file. In some graphics formats, SCANLINE rendering can produce better quality output might be distorted if the output is replayed on a device with a different resolution than the original device. If the system text is rendered as a POLYGON, resizing the graph will not distort the text.

SYMBOL, NOSYMBOL Device Parameters and Graphics Options

Specifies whether to use the device's symbol-drawing capability.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

See: SYMBOLS on page 699

Syntax

Form 1: GOPTIONS:

SYMBOL | NOSYMBOL

Form 2: GDEVICE:

SYMBOL=Y | N

Parameter Values

SYMBOL
SYMBOL=Y
causes SAS/GRAPH to use the built-in symbol-drawing capability of the device, if available. A blank `Symbol` field in the Parameters window is the same as SYMBOL=Y.

Tip Hardware drawing is faster, but not all devices have the capability. SAS/GRAPH does not try to use the capability if your device does not support it.

NOSYMBOL
SYMBOL=N
causes SAS/GRAPH to draw the symbols using SAS/GRAPH fonts.

SYMBOLS Device Parameter

Specifies which symbols can be generated by hardware.

Used by: GDEVICE procedure, GDEVICE Parameters window

Default: device-dependent

See: "SYMBOL Statement" on page 470

Syntax

SYMBOLS='*hardware-symbols-hex-string*'X

Parameter Values

hardware-symbols-hex-string

is a hexadecimal string that is 16 characters long and must be completely filled. This table shows which bit position (left-to-right) within the hexadecimal string controls each hardware symbol.

Table 26.3 *Bit Positions That Control Hardware Symbols*

Bit to Turn On	Symbol Name	Symbol
1	PLUS	+
2	X	×
3	STAR	✳
4	SQUARE	□
5	DIAMOND	◇
6	TRIANGLE	△
7	HASH	#
8	Y	Y
9	Z	Z
10	PAW	⁚

Bit to Turn On	Symbol Name	Symbol
11	POINT	.
12	DOT	●
13	CIRCLE	○

For example, if you want the driver to do only the PLUS and X symbols in hardware, the first and second bits of the first byte of the hexadecimal string should be turned on, which would correspond to a value of 'C000000000000000'X ('C0'X is equivalent to '1 1 0 0 0 0 0 0' in binary).

Details

These are not the only symbols that can be generated for graphics output but are the symbols that can be drawn by the hardware. SAS/GRAPH can draw other symbols.

Note: Not all devices are capable of drawing every symbol. If a particular hardware symbol is not available or not to be used (as indicated by the value of SYMBOLS), the symbol is generated by the software. If the value of the SYMBOL device parameter in the device entry is N or the NOSYMBOL graphics option is used, the value of SYMBOLS is ignored.

TARGETDEVICE Graphics Option

Displays the output as it would appear on a different device. Also, specifies the device driver for the PRINT command.

Used by: GOPTIONS statement

Alias: TARGET

Restriction: This option is not supported by Java or ActiveX.

Syntax

TARGETDEVICE=*target-device-entry*

Parameter Values

target-device-entry
specifies the name of a device entry in a catalog.

Details

Use TARGETDEVICE= to specify a device driver when you want to do the following:

- preview graphics output on your monitor as it would appear on a different output device. For details, see "Previewing Output" on page 150.
- print output from the Graph window or the Graphics Editor window with the PRINT command. For details, see "Printing Your Graph" on page 151.
- specify a device driver for graphics output created by the ODS HTML statement.

TRAILER Device Parameter

Specifies the command that creates TRAILER records for the Metagraphics driver.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers

See: TRAILERFILE on page 702

Syntax

TRAILER='*command*'

Parameter Values

command
specifies a command that runs a user-written program that creates the TRAILER file.

Restriction *Command* cannot exceed 40 characters.

Details

For information about Metagraphics drivers, contact Technical Support.

TRAILERFILE Device Parameter

Specifies the fileref of the file from which the Metagraphics driver reads TRAILER records.

Used by: GDEVICE procedure, GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers

See: TRAILER on page 701

Syntax

TRAILERFILE=*fileref*

Parameter Values

fileref

specifies a valid SAS fileref up to eight characters long. *Fileref* must have been previously assigned with a FILENAME statement or a host command before running the Metagraphics driver. See "FILENAME Statement" on page 52 for additional information about the FILENAME statement.

Details

For information about Metagraphics drivers, contact Technical Support.

TRANSPARENCY, NOTRANSPARENCY Graphics Options

Specifies whether the background of the image should appear to be transparent when the image is displayed in the browser.

Used by: GOPTIONS statement

Default: NOTRANSPARENCY

Restriction: This option is supported by the following device drivers only:

- the PNG device driver
- the SVG device drivers
- the GIF device drivers
- the ACTIVEX and ACTXIMG device drivers only when the output is used in a Microsoft PowerPoint presentation

Syntax

TRANSPARENCY | NOTRANSPARENCY

Comparisons

When the image is displayed and TRANSPARENCY is in effect, the browser's background color replaces the driver's background color, causing the image to appear transparent.

Note: It is recommended that you set the background color of your GIF output to match the background color of the presentation in which you want to use the GIF image.

TRANTAB Device Parameter and Graphics Option

Selects a translate table for your system that performs ASCII-to-EBCDIC translation.

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window
Default:	host-dependent
Restriction:	This option is not supported by Java or ActiveX.

Syntax

TRANTAB=*table* | *user-defined-table*

Parameter Values

table
specifies a translate table stored as a SAS/GRAPH catalog entry. *Table* can be one of the following:

- SASGTAB0 (default translate table for your operating environment)
- GTABVTAM
- GTABTCAM

user-defined-table
specifies the name of a user-created translate table.

Details

TRANTAB is set by the SAS Installation Representative and is needed when an EBCDIC host sends data to an ASCII graphics device. See the SAS/GRAPH installation instructions for details. You can also create your own translate table using the TRANTAB procedure. For a description of the TRANTAB Procedure, see *Base SAS Procedures Guide*.

TYPE Device Parameter

Specifies the type of output device to which graphics commands are sent.

Used by: GDEVICE procedure, GDEVICE Detail window

Default: device-dependent

Syntax

TYPE=CAMERA | CRT | EXPORT | PLOTTER | PRINTER

Parameter Values

CAMERA
specifies a film-recording device.

CRT
specifies a monitor or terminal.

EXPORT
identifies the list in which the device appears under SAS/ASSIST software. This is used for drivers that produce output to be exported to other software applications, such as CGM or HPGL.

PLOTTER
specifies a pen plotter.

PRINTER
specifies a printer

Details

You should not modify this value for device drivers supplied by SAS.

UCC Device Parameter and Graphics Option

Sets the user-defined control characters for the device.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Restrictions: This option is device-dependent.
This option is not supported by Java or ActiveX.

Syntax

UCC='*control-characters-hex-string*'X

Parameter Values

control-characters-hex-string
specifies the user-defined control characters for the device as a hexadecimal string that can be up 32 bytes (64 characters) long. You need to specify only up to the last nonzero byte; the remaining bytes will be set to zero.

Details

Not all devices support this feature, and the meaning of each byte of the string varies from device to device.

Typically, the UCC byte position is indicated by a bracketed value. For example, UCC[2] refers to the second byte of the string. For assistance with determining UCC values for your specific device, please contact SAS Technical Support.

USERINPUT, NOUSERINPUT Graphics Options

Determines whether user input is enabled for the device.

Used by: GOPTIONS statement

Default: NOUSERINPUT

Restrictions: This option is supported by the GIF driver only.
This option is not supported by all browsers.

Syntax

USERINPUT | NOUSERINPUT

Parameter Values

USERINPUT
enables user input

NOUSERINPUT
disables user input

Details

When user input is enabled, processing of the animation is suspended until a carriage return, mouse click, or some other application-dependent event occurs. The user input feature works with the delay time setting so that processing continues when user input occurs or the delay time has elapsed, whichever comes first.

VORIGIN Device Parameter and Graphics Option

Sets the vertical offset from the lower left corner of the display area to the lower left corner of the graph.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Restriction: This option is not supported by Java or ActiveX.

See: HORIGIN on page 647

Syntax

VORIGIN=*vertical-offset*<IN | CM | PT>

Parameter Values

***vertical-offset*<IN | CM | PT>**
specifies the vertical offset as a nonnegative number optionally followed by a units specification of IN for inches (default), CM for centimeters, or PT for points. If you do not specify VORIGIN, a default offset is searched for in this order:

1. the bottom margin specification on an OPTIONS BOTTOMMARGIN setting
2. VORIGIN setting in the device catalog.

Details

The display area is defined by the XMAX and YMAX device parameters. By default, the origin of the graphics output area is the lower left corner of the display area; the graphics output is offset from the lower left corner of the display area by the values of HORIGIN and VORIGIN. VORIGIN + VSIZE cannot exceed YMAX.

Note: When sending output to the PRINTER destination (ODS PRINTER), if you specify the VSIZE= option without specifying the HSIZE= option, the default origin of the graphics output area changes. The default placement of the graph changes from

the lower left corner of the display area to the top-center of the graphics output area. Likewise, if you specify the HSIZE= option without specifying the VSIZE= option, the graph is positioned at the top-center of the graphics output area by default.

See "The Graphics Output and Device Display Areas" on page 80 for details.

VPOS Graphics Option

Sets the number of rows in the graphics output area.

Used by: GOPTIONS statement

Default: device-dependent: the value of the LROWS or PROWS device parameter

Restriction: This option is not supported by Java or ActiveX.

See: HPOS on page 648
LROWS on page 659
PROWS on page 686

Syntax

VPOS=*rows*

Parameter Values

rows
specifies the number of rows in the graphics output area, which is equivalent to the number of hardware characters that can be displayed vertically. Specifying VPOS=0 causes the device driver to use the default hardware character cell height for the device.

Details

The VPOS= graphics option overrides the values of the LROWS or PROWS device parameters and temporarily sets the number of columns in the graphics output area. VPOS= does not affect the height of the graphics output area but merely divides it into rows. Therefore, you can use VPOS= to control cell height.

The values specified in the HPOS= and VPOS= graphics options determine the size of a character cell for the graphics output area and consequently the size of many graphics elements, such as hardware text. The larger the size of the HPOS= and VPOS= values, the smaller the size of each character cell.

See "Overview" on page 79 for more information.

VSIZE Device Parameter and Graphics Option

Sets the vertical size of the graphics output area.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Restriction: This option is partially supported by Java and ActiveX.

See: HSIZE on page 649
YMAX on page 712

Syntax

VSIZE=*vertical-size*<IN | CM | PT>

Parameter Values

***vertical-size*<IN | CM | PT>**
specifies the height of the graphics output area as a positive number optionally followed by a units specification of IN for inches (default), CM for centimeters, or PT for points. If you do not specify the VSIZE= option, a default size is searched for in this order:

1. the vertical size is calculated as YMAX − BOTTOMMARGIN − TOPMARGIN. Note that BOTTOMMARGIN and TOPMARGIN are used in the OPTIONS statement.
2. VSIZE setting in the device catalog.

V6COMP, NOV6COMP Graphics Options

Allows programs that are run in the current version of SAS to run with selected Version 6 defaults.

Used by: GOPTIONS statement

Default: NOV6COMP

Restrictions: This option is partially supported by Java and ActiveX.
This option is ignored unless OPTIONS NOGSTYLE is also specified.

Syntax

V6COMP | NOV6COMP

Parameter Values

V6COMP
causes SAS/GRAPH programs to use certain Version 6 behaviors for compatibility. The Version 6 behaviors are as follows:

- By default, patterns are hatched patterns, not solid, and the default outline color matches the pattern color.
- By default, the GCHART and GPLOT procedures do not draw a frame around the axis area.

NOV6COMP
causes SAS/GRAPH programs to use all the features of the current SAS version.

Details

V6COMP performs the necessary conversions so that, for selected defaults, you get the same results in the current SAS version that you did in Version 6.

Note: V6COMP does not convert Version 6 catalogs to catalogs with the current SAS catalog format.

XMAX Device Parameter and Graphics Option

Specifies the width of the addressable graphics display area; affects the horizontal resolution of the device and the horizontal dimension of the graphics output area.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Restriction: Ignored by default display drivers, universal printing drivers, Java, and ActiveX

See: HSIZE on page 649
PAPERSIZE on page 665
XPIXELS on page 711

Syntax

XMAX=*width*<IN | CM | PT>

Parameter Values

width
specifies the width of the addressable graphics display area as a positive number optionally followed by a units specification of IN for inches (default), CM

for centimeters, or PT for points. If you do not specify XMAX, a default width is searched for in this order:

1. the *width* specification on an OPTIONS PAPERSIZE setting
2. XMAX in the device entry catalog.

If XMAX=0, default behavior is used. If both XMAX and PAPERSIZE have been specified on GOPTIONS, the last request is used.

Details

Like the XPIXELS device parameter, XMAX controls the width of the display area, but the width is in inches, centimeters, or points rather than pixels. Typically, you might use XMAX to change the width of the display area for a hard-copy device.

SAS/GRAPH uses the value of XMAX in calculating the horizontal resolution of the device: *x-resolution* = XPIXELS / XMAX.

However, changing XMAX does not necessarily change the resolution:

- If you use the GOPTIONS statement to change only the value of XMAX= and do not change XPIXELS=, SAS/GRAPH retains the default resolution of the device and recalculates XPIXELS, temporarily changing the width.
- If you specify values for both XMAX= and XPIXELS=, SAS/GRAPH recalculates the resolution of the device using both of the specified values. The new resolution might be different. For example, both of these pairs of values produce the same resolution, 300dpi:

  ```
  XPIXELS=1500 and XMAX=5
  ```

  ```
  XPIXELS=1800 and XMAX=6
  ```

XMAX also affects the value of HSIZE, which controls the horizontal dimension of the graphics output area.

- If you change the value of XMAX and do not change HSIZE=, SAS/GRAPH calculates a new value for HSIZE=, using this formula:

  ```
  HSIZE = XMAX - margins
  ```

  Note: The *margins* quantity, here, is not a device parameter. It represents the value of the left margin plus the right margin. The left margin is the value of HORIGIN. The right margin is whatever is left over when you subtract HSIZE and HORIGIN from XMAX. The value of *margins* is always based on the original XMAX and HSIZE values that are stored in the device entry.

- If you specify values for both XMAX= and HSIZE=, SAS/GRAPH uses the specified values plus the value of device parameter HORIGIN. Anything left over is added to the right margin. For example, if XMAX=6IN and HSIZE=4IN and HORIGIN=.5IN, the right margin will be 1.5in. If HSIZE= is larger than XMAX=, HSIZE= is ignored.

To permanently change the value of the XMAX device parameter in the device entry, use the GDEVICE procedure. This can change the resolution.

To temporarily change the size of the display and the resolution of the device for the current graph or for the duration of your SAS session, use XMAX= and XPIXELS= in the GOPTIONS statement.

To reset the value of XMAX to the default, specify XMAX=0. To return to the default resolution for the device, specify both XMAX=0 and XPIXELS=0.

See "Overview" on page 79 for more information.

XPIXELS Device Parameter and Graphics Option

Specifies the width of the addressable display area in pixels and in conjunction with XMAX determines the horizontal resolution for the device.

Used by: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Default: device-dependent

Restriction: This option is partially supported by Java and ActiveX.

See: XMAX on page 709

Syntax

XPIXELS=*width-in-pixels*

Parameter Values

width-in-pixels
specifies the width of the addressable display area in pixels as an integer in the range 0–99999999.

Details

Like the XMAX device parameter, XPIXELS controls the width of the display area, but the width is in pixels rather than inches, centimeters, or points. Typically, you might use XPIXELS to change the width of the display area for an image format device.

Note: This option overrides the OutputWidth style attribute in the graph styles. For more information about graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: Procedures Guide*.

The value of XPIXELS is used in calculating the resolution of the device: *x-resolution* = XPIXELS / XMAX. However, changing XPIXELS does not necessarily change the device resolution:

- If you use the GOPTIONS statement to change only the value of XPIXELS= and do not change XMAX=, SAS/GRAPH retains the default resolution of the device

and recalculates XMAX, temporarily changing the width of the display. If HSIZE= is also not specified, SAS/GRAPH uses the new XMAX value to calculate a new HSIZE value as follows: HSIZE = XMAX − *margins*.

Note: *margins* are not device parameters, but represent the value of HORIGIN (the left margin) plus the right margin. The right margin is whatever is left over when you subtract HSIZE and HORIGIN from XMAX. The values of *margins* are always based on the original XMAX and HSIZE values that are stored in the device entry.

If HSIZE= is specified and its value is larger than XMAX, HSIZE= is ignored.

- If you use the GDEVICE procedure to permanently change the value of the XPIXELS device parameter in the device entry, SAS/GRAPH automatically recalculates the resolution of the device is using the value of XMAX device parameter.
- If you change the values of both XMAX= and XPIXELS=, SAS/GRAPH recalculates the resolution of the device using both of the specified values.

Note: When SAS/GRAPH recalculates the resolution, the resolution does not necessarily change. For example, both of these pairs of values produce the same resolution, 300dpi:

```
XPIXELS=1500 and XMAX=5
XPIXELS=1800 and XMAX=6
```

To reset the value of XPIXELS to the default, specify XPIXELS=0. To return to the default resolution for the device, specify both XPIXELS=0 and XMAX=0.

YMAX Device Parameter and Graphics Option

Specifies the height of the addressable graphics display area; affects the vertical resolution of the device and the vertical dimension of the graphics output area.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Restrictions: This option is ignored by the default display drivers and the universal printing drivers.
This option is not supported by Java or ActiveX.

See: PAPERSIZE on page 665
VSIZE on page 708
YPIXELS on page 713

Syntax

YMAX=*height*<IN | CM | PT>

Parameter Values

height

specifies the height of the addressable graphics display area as a positive number optionally followed by a units specification of IN for inches (default), CM for centimeters, or PT for points. If you do not specify YMAX, a default height is searched for in this order:

1. the *height* specification on an OPTIONS PAPERSIZE setting
2. YMAX in the device entry catalog.

If YMAX=0, default behavior is used. If both YMAX and PAPERSIZE have been specified on GOPTIONS, the last request is used.

See Also

"XMAX Device Parameter and Graphics Option" on page 709

YPIXELS Device Parameter and Graphics Option

Specifies the height of the addressable display area in pixels and in conjunction with YMAX determines the horizontal resolution for the device.

Used by:	GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window
Default:	device-dependent
Restriction:	This option is partially supported by Java and ActiveX.
See:	YMAX on page 712

Syntax

YPIXELS=*height-in-pixels*

Parameter Values

height-in-pixels

specifies the height of the addressable display area in pixels as an integer in the range 0–99999999.

Details

Note: This option overrides the OutputHeight style attribute in the graph styles. For more information about graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: Procedures Guide*.

See Also

"XPIXELS Device Parameter and Graphics Option" on page 711

27

SAS System Options Used by SAS/GRAPH

Introduction to System Options

This chapter provides a detailed description of the system options used with SAS/GRAPH software. The descriptions provide the syntax, defaults, and related options for each option.

The system options are listed alphabetically.

System options are instructions that affect the processing of an entire SAS program or interactive SAS session from the time the option is specified until it is changed. Examples of items that are controlled by SAS system options include the appearance of SAS output, the handling of some files that are used by SAS, the use of system variables, and the processing of observations in SAS data sets.

For detailed information about using SAS System Options, see *SAS System Options: Reference*

Dictionary

DEVICE= System Option

Specifies the device driver to which SAS/GRAPH sends procedure output.

Valid in:	SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Graphics: Driver Settings
PROC OPTIONS GROUP=	GRAPHICS
Alias:	DEV=
Requirement:	This option is required for the LISTING destination when you run SAS in a non-interactive mode.
Note:	This option can be restricted by a site administrator. For more information, see "Restricted Options" in *SAS System Options: Reference*.
See:	"DEVICE System Option: UNIX" in *SAS Companion for UNIX Environments*, "DEVICE System Option: Windows" in *SAS Companion for Windows* "DEVICE= System Option: z/OS" in *SAS Companion for z/OS*

Syntax

DEVICE=*device-driver-specification*

Syntax Description

device-driver-specification
specifies the name of a device driver.

Details

If you omit the device-driver name, the Output Delivery System selects a device driver for the opened destinations. If you specify a device that is not compatible with the opened destinations, the Output Delivery System selects a device that is valid. If you enter an invalid device driver, SAS prompts you to enter a device driver when you execute a procedure that produces graphics.

A best practice is to let the Output Delivery System select the device driver, except when you use the LISTING destination. When the LISTING destination is open, the default device is the Graph window.

See Also

"Using Graphics Devices" on page 89

GSTYLE System Option

Specifies whether ODS styles can be used to generate graphs that are stored as GRSEG catalog entries.

Valid in:	SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Categories:	Graphics: Driver Settings Log and Procedure Output Control: ODS Printing
PROC OPTIONS GROUP=	GRAPHICS ODSPRINT
Note:	This option can be restricted by a site administrator. For more information, see "Restricted Options" in *SAS System Options: Reference*.

Syntax

GSTYLE | **NOGSTYLE**

Syntax Description

GSTYLE
specifies that ODS styles can be used in the generation of graphs that are stored as GRSEG catalog entries. If no style is specified, the default style for the given output destination is used. This is the default.

NOGSTYLE
specifies to not use ODS styles in the generation of graphs that are stored as GRSEG catalog entries.

Tip Use NOGSTYLE for compatibility of graphs generated prior to SAS 9.2.

Details

The GSTYLE system option affects only graphic output that is generated using GRSEGs. The GSTYLE option does not affect the use of ODS styles in graphs that are generated by the following means:

- ActiveX device driver
- SAS/GRAPH statistical graphic procedures
- SAS/GRAPH template language
- ODS GRAPHICS ON statement

GWINDOW System Option

Specifies whether SAS displays SAS/GRAPH output in the GRAPH window.

Valid in:	SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Graphics: Driver Settings
PROC OPTIONS GROUP=	GRAPHICS
Note:	This option can be restricted by a site administrator. For more information, see "Restricted Options" in *SAS System Options: Reference*.

Syntax

GWINDOW | **NOGWINDOW**

Syntax Description

GWINDOW
displays SAS/GRAPH software output in the GRAPH window, if your site licenses SAS/GRAPH software and if your personal computer has graphics capability.

NOGWINDOW
displays graphics outside of the windowing environment.

PART 6

The Annotate Facility

28

Using Annotate Data Sets

Overview: The Annotate Facility

Overview of the Annotate Facility

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 generate custom graphics by using an Annotate data set with GANNO (Chapter 33, “GANNO Procedure,” on page 887), or GSLIDE (Chapter 45, “GSLIDE Procedure,” on page 1475). You can enhance your graphics output by applying an Annotate data set to graphics that you generated using procedures such as GCHART (Chapter 36, “GCHART Procedure,” on page 983), GPLOT (Chapter 42, “GPLOT Procedure,” on page 1257), and GMAP (“GMAP Procedure” in *SAS/GRAPH and Base SAS: Mapping Reference*), among others.

In addition, SAS/GRAPH supports using annotation in the following procedures with the JAVAIMG, ACTIVEX, or ACTXIMG device: GCHART, GCONTOUR, GMAP, GPLOT, GRADAR, 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 do the following:

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

This figure shows the GCHART procedure using the VBAR statement to create annotated vertical bars. The units ordered, the city, and type text are all annotated.

See "Example 1: Labeling Subgroups in a Vertical Bar Chart" on page 751 for the program that generated this bar chart.

Creating Custom Graphs

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

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

Figure 28.1 on page 723 is an example of a custom graph that uses Annotate commands to draw the graphics elements.

Figure 28.1 *Custom Graphics Using Only Annotate Commands*

The program that creates this output is in "Example 2: Drawing a Circle of Stars" on page 754.

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 725. Once you understand how 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 732.)
2. Build an Annotate data set of graphics commands using the Annotate variables and functions. (See "Creating an Annotate Data Set" on page 739.)
3. Submit a SAS/GRAPH procedure to produce the graphics output. (See "Producing Graphics Output from Annotate Data Sets" on page 741.)

The Annotate Function, Variable, and Macro Dictionaries

You can find detailed information about how to use the Annotate facility functions, variables, and macros in the Annotate dictionaries. The following table lists the dictionaries that are available for the Annotate facility:

Functions, Variables, or Macros	Dictionary
Functions	"Annotate Function Dictionary" (p. 759)
Variables	"Annotate Variable Dictionary" (p. 803)
Macros	"Annotate Macro Dictionary" (p. 853)

You can find information about error messages that might appear when you invoke an Annotate function or macro in "Annotate Error Messages" on page 746.

About the Annotate Data Set

Overview of Annotate Data Sets

In an Annotate data set, each observation represents a command to draw a graphics element or perform an action. The observations use a set of predefined annotate variables on page 804. Annotate functions on page 759 determine what is to be done with each observation. Annotate macros on page 854 simplify the process of drawing a graphics element. Annotate error messages on page 746 are sent to the SAS log. For usage information and example programs , refer to "Using Annotate Macros" on page 854 and Chapter 28, "Using Annotate Data Sets," on page 721.

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 739.

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 28.1 on page 725 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 739.)

Output 28.1 *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 value 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 that it performs:

1. Create a label. This instruction draws a green 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. They also specify 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.

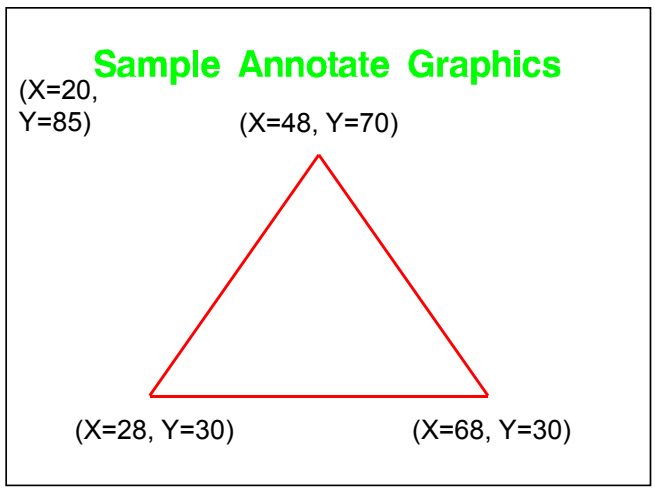
3. Draw the first line of the triangle. The value of the FUNCTION variable (DRAW) tells the program to draw a line. The line is drawn from the current point (the one specified by MOVE in the second observation) to the new point specified by X and Y. The value of the COLOR variable changes to red.

4. Draw the second line of the triangle.

5. Draw the third line of the triangle.

Figure 28.2 on page 726 shows the green title and the red triangle produced by the TRIANGLE data set and displayed with the GANNO on page 887 procedure. Notes on the figure in black contain the X and Y coordinates of the graphics elements.

Figure 28.2 *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 variable that is used to generate a drill-down graph that is viewed in a web browser.

When used with text labels and SVG devices, the HTML variable enables you to animate the text. For more information, see "LABEL Function" on page 774.

Table 28.1 on page 727 lists all Annotate variables, grouped by task, and briefly describes each one. See "About the Annotate Variables" on page 804 for a complete description of each variable.

Table 28.1 *Summary of Annotate Variables*

Task Group	Variable	Description
Variable that defines an action	FUNCTION	specifies a drawing or programming action; Table 28.2 on page 731 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
	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

Task Group	Variable	Description
	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
	IMGPATH	path to an image file to be displayed.
	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

Task Group	Variable	Description
	WHEN	whether a graphics element is drawn before or after procedure graphics output
web variable	HTML	specifies link information for a drill-down graph for text labels, can be used to animate the text when using SVG devices

See Figure 28.3 on page 730 for a table that shows you which Annotate functions are used with which Annotate variables.

Figure 28.3 Annotate Variables Used with Annotate Functions

Variables \ Functions	ARROW	BAR	CNTL2TXT	COMMENT	DEBUG	DRAW	DRAW2TXT	FRAME	IMAGE	LABEL	MOVE	PIE	PIECNTR	PIEXY	POINT	POLY	POLYGONT	POP	PUSH	SWAP	SYMBOL	TXT2CNTL
ANGLE	x									x		x		x								
CBORDER										x											x	
CBOX										x											x	
COLOR	x	x				x	x	x		x		x			x	x	x				x	
FUNCTION	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
GROUP	x	x				x				x	x	x	x		x	x	x				x	
HSYS	x					x	x	x		x		x	x								x	
HTML	x	x						x	x	x		x				x					x	
IMGPATH									x													
LINE	x	x				x	x	x				x				x						
MIDPOINT	x	x				x				x	x	x	x		x	x	x				x	
POSITION										x												
ROTATE										x		x										
SIZE	x	x				x	x	x		x		x	x	x		x					x	
STYLE (fonts)										x											x	
STYLE (images)									x													
STYLE (patterns)	x	x						x				x				x						
SUBGROUP	x	x				x				x	x	x	x		x	x	x				x	
TEXT				x						x											x	
WHEN	x	x				x	x	x		x	x	x	x	x	x	x	x				x	
X	x	x				x			x	x	x	x	x		x	x	x				x	
XC	x	x				x				x	x	x	x		x	x	x				x	
XSYS	x	x				x		x		x	x	x	x		x	x	x				x	
Y	x	x				x			x	x	x	x	x		x	x	x				x	
YC	x	x				x				x	x	x	x		x	x	x				x	
YSYS	x	x				x		x		x	x	x	x		x	x	x				x	
Z	x	x				x			x	x	x	x	x		x	x	x				x	
ZSYS	x	x				x			x	x	x	x	x		x	x	x				x	
XLAST YLAST	x	x	x			x			x		x	x	x	x	x			x	x	x		x
XLSTT YLSTT			x							x								x	x	x	x	x

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 732.

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 737, "Using the LIFO Stack" on page 744, and "Debugging" on page 745.

Table 28.2 on page 731 summarizes the tasks that are performed by the Annotate functions. See "About the Annotate Functions" on page 759 for a complete description of the FUNCTION variable and its values.

Table 28.2 *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 specify a fill color and pattern	POLY
	continue drawing a polygon (additional vertex) and specify an outline color of the polygon	POLYCONT
	draw an arrow from the current (X,Y) position (see MOVE and TXT2CNTL)	ARROW
	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) and 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; animate the text	LABEL
	move to the specified point (X,Y)	MOVE
	put a frame around the area defined by XSYS and YSYS and 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

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

See Figure 28.3 on page 730 for a table that shows you which Annotate functions work with which Annotate variables.

About Annotate Graphics

Overview of 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).

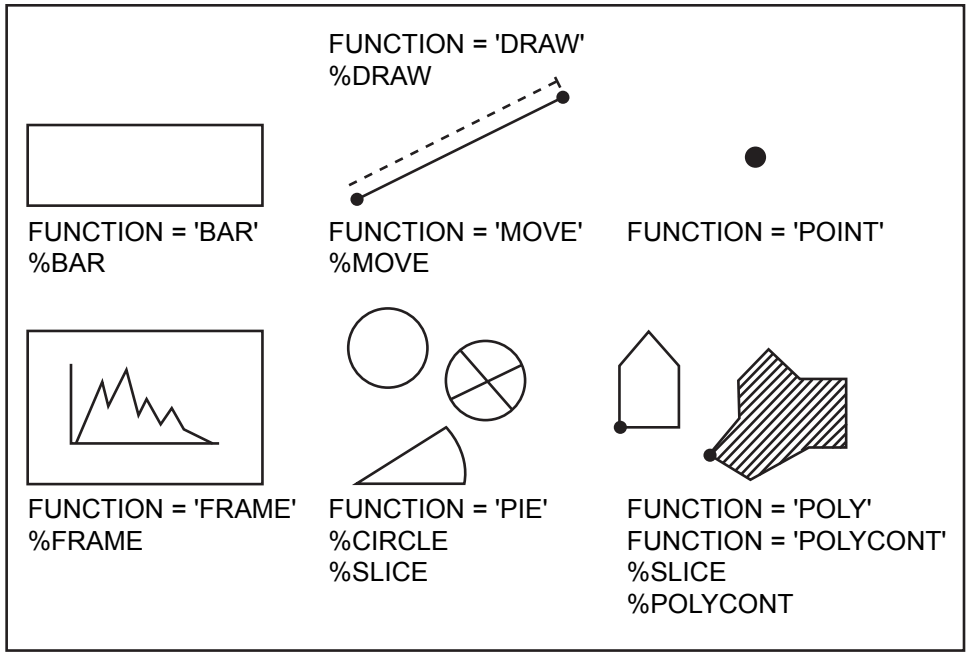
The following sections describe the components of the graphics output that are produced by an Annotate data set.

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 28.4 on page 733 along with the value of the FUNCTION variable or Annotate macro that draws them.

Figure 28.4 *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 sometimes Z are used for numeric coordinates.
- XC and YC are used for character coordinates.
- GROUP, MIDPOINT, and SUBGROUP can be used when you annotate output from procedures such as GCHART. 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

Overview of 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 types of values as XSYS, YSYS, and ZSYS.

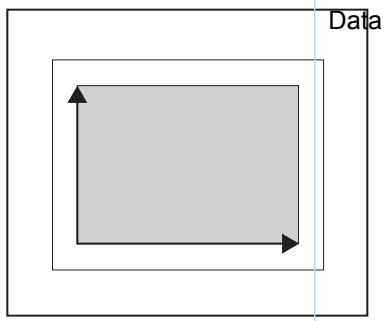
The SIZE variable can be used to specify the following:

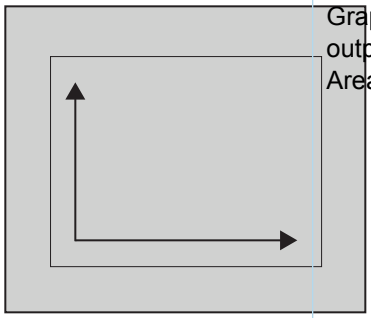
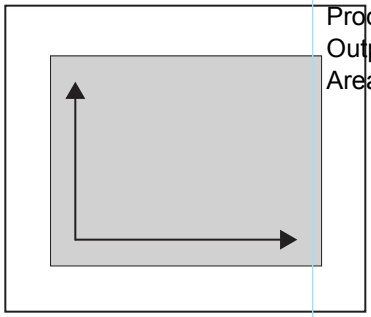
- size of a graphics element, such as the width of lines (for example, FRAME)
- radius of pie slices (for example, PIE, PIECNTR, and PIEXY)
- height of text (for example, LABEL and SYMBOL)

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. Table 28.3 on page 734 shows the areas on the graphics output and the coordinate systems that use them.

Table 28.3 *Drawing Areas and Their Coordinate Systems*

Drawing Area	Unit	Coordinate System	
		Absolute	Relative
	%	1	7
	Values	2	8

Drawing Area	Unit	Coordinate System	
		Absolute	Relative
 Graphics output Area	%	3	9
	Cells	4	A
 Procedure Output Area	%	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 736.
  - 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 28.4 on page 735 describes the coordinate system values for the XSYS, YSYS, ZSYS, and HSYS variables.

Table 28.4 *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' *

Type of Coordinates	Area	Units	Range	Value for XSYS, YSYS, ZSYS, HSYS
	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' *
	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'
none	Text font point size	none	0 to limit of graphics output area	'D'**

*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. In addition, coordinate systems 2 and 8 are not valid with radar charts in the GRADAR procedure.

**Coordinate system D is used only for text functions such as LABEL. For functions that do not create text, a warning appears in the log and the 4-coordinate system is used.

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. The rows and columns 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. The rows and columns are 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 “Overview” on page 79 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. (For details, see “BAR Function” on page 763.) These internal variables can also provide default coordinates if X, XC, Y, or YC contains a missing value.

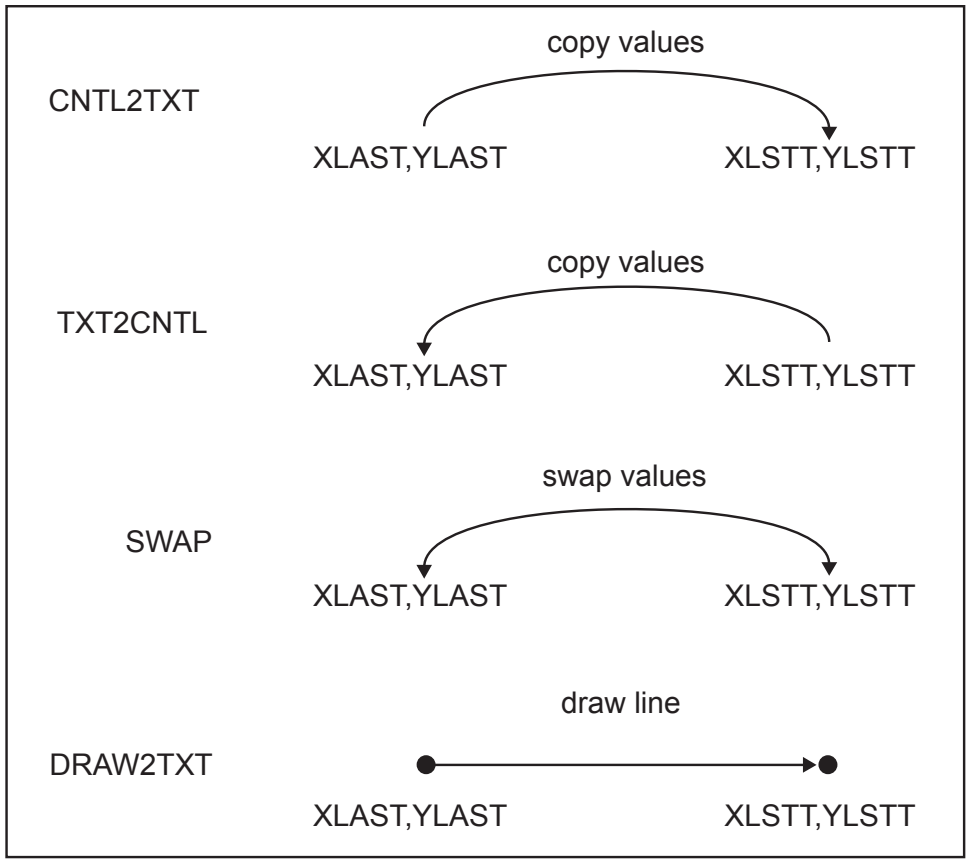
The internal coordinates are automatically updated by some of the Annotate functions. The text functions, LABEL and SYMBOL, update the (XLSTT,YLSTT) variables. The BAR, DRAW, MOVE, PIE, and POINT functions update the (XLAST,YLAST) variables.

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 28.5 on page 738.

Figure 28.5 *Programming Functions That Manipulate System Variables*



For a complete description, see "Annotate Internal Coordinates" on page 804.

Attribute Variables

Attribute variables control the appearance of the graphics elements. Each function uses only a subset of these variables. See Table 28.1 on page 727 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'. However, SIZE 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 control, see "About the Annotate Variables" on page 804.

Creating an Annotate Data Set

Overview of 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. 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 documentation use a DATA step with assignment statements. For more information about creating SAS data sets, see SAS Data Sets.

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 28.1 on page 725:

```
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 annotate=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 statement takes as input the annotate data set triangle created by the previous DATA step and creates the output shown in Figure 28.2 on page 726.

Using Annotate Macros in the DATA Step

A set of Annotate macros is provided in the SAS sample library.

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

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 "About the Annotate Macros" on page 854 and "Using Annotate Macros" on page 854.

Effect of Missing Values

Annotate data sets follow the same rules for missing values as any other SAS data set. (See "Missing Variable Values" in *SAS Programmer's Guide: Essentials* for information about 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 the color list that is defined by the COLORS= graphics option is used. If the COLORS= option is not specified, then the device's default color 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 "About the Annotate Variables" on page 804 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 might need to 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 stored the values in (XLAST,YLAST) or (XLSTT,YLSTT), then you can force Annotate to use the internal coordinates. You

do this by supplying missing values for the X and Y variables. See “Annotate Internal Coordinates” on page 804 for details about using the (XLAST,YLAST) and (XLSTT,YLSTT) internal coordinates.

Producing Graphics Output from Annotate Data Sets

Overview of Producing Graphics Output

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 without other procedure output, use the GANNO procedure or the GSLIDE procedure:

- The GANNO procedure produces graphics output consisting only of Annotate graphics. See Chapter 33, “GANNO Procedure,” on page 887 for information about 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 45, “GSLIDE Procedure,” on page 1475 for details.

Using the Annotate Variables for Web Output

Most of the annotate variables can be used in programs that generate output for the web. For more information about the annotate functions and variables, see "Annotate Function Dictionary" on page 759 and "Annotate Variable Dictionary" on page 803. For information about using annotate data sets in web output, see "When to Apply Annotate Data Sets to Web Presentations" on page 265.

Annotate Processing Details

Order in Which Graphics Elements Are Drawn

Overview of Drawing Order

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 showing 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 whether observations are processed 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 in which they appear in the Annotate data set. The graphics elements 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 838.

Order of Annotate Layers

When you annotate a graph, you create additional transparent layers that are interspersed with the existing layers (starting from the bottom):

1. the graph background (back wall)
2. the WHEN="B" layer for all annotations except the LABEL and SYMBOL functions
3. axis frame area (chart wall)
4. the WHEN="B" layer for the LABEL and SYMBOL functions
5. the graphical data (plot points, join lines, bars, pie slices, and so on)
6. axes (axis lines, tick values, tick marks, labels)
7. the WHEN="A" layer for all annotations except the LABEL and SYMBOL functions
8. the WHEN="A" layer for the LABEL and SYMBOL functions

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 association must occur 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. Specifying ANNOTATE= in an action statement in the procedure generates Annotate graphics for unique values of the BY variable.

See “BY Statement” on page 424 for details.

Using the LIFO Stack

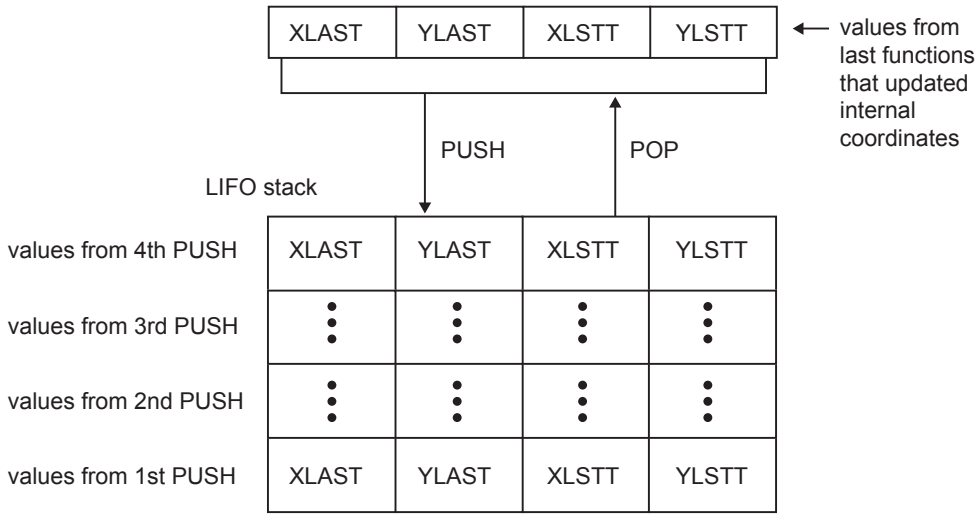
The FUNCTION variable supports several programming functions that manipulate the internal coordinates and provide other utility operations. Several of these functions use the LIFO stack to track and set variable values.

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 28.6 on page 745 illustrates how PUSH and POP functions work together.

Figure 28.6 *Using PUSH and POP to Store and Retrieve Coordinate Values*



See also "Internal Coordinates" on page 737.

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 28.1 on page 725 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.

List of Annotate Examples

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

- "Specifying Response Levels in a Block Map" in *SAS/GRAPH and Base SAS: Mapping Reference*
- "Example 1: Labeling Subgroups in a Vertical Bar Chart" on page 751
- "Example 2: Drawing a Circle of Stars" on page 754

Other examples that use Annotate data sets are as follows:

- "Example 1: Scaling Data-Dependent Output" on page 893 (and others in that chapter)
- "Mapping and Annotating Values from the GINSIDE Procedure" in *SAS/GRAPH and Base SAS: Mapping Reference*
- "Example 2: Displaying Annotate Graphics" on page 1484
- "Projecting an Annotate Data Set" in *SAS/GRAPH and Base SAS: Mapping Reference*
- Example Code 29.5 on page 779

Annotate Error Messages

If there is an error in your Annotate data set, one or more diagnostic messages are printed in the SAS log. A partial list of these messages is supplied here. Annotate data sets are checked for errors in this way:

- 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
  ```

Some common diagnostic messages are explained here.

A CALCULATED COORDINATE LIES OUTSIDE THE VISIBLE AREA
Explanation: The *x* or *y* coordinate is outside the display area (defined by HPOS= and VPOS= values).

User Action: Check for an invalid or misspecified coordinate system value, or *x* or *y* values outside displayed range.

A CALCULATED WINDOW COORDINATE LIES OUTSIDE THE WINDOW AREA
Explanation: the *x* or *y* coordinate is outside of the window area. This message might accompany the message for invalid coordinate system specification.

User Action: Check for an invalid or misspecified coordinate system value, or *x* or *y* values outside displayed range.

A PERCENTAGE VALUE LIES OUTSIDE 0 TO 100 BOUNDARIES
Explanation: The *x* or *y* value requested is negative or greater than 100%. This message is informational.

User Action: Check requested value for accuracy.

ANNOTATE MIDPOINT DATATYPE DOES NOT MATCH GCHART- INPUT WAS #
Explanation: The MIDPOINT variable in the Annotate data set is character, and the GCHART midpoint is numeric or vice versa.

User Action: Check for misspelling or wrong variable assignment, or check for quotation marks in the assignment statement.

ANNOTATE GROUP DATATYPE DOES NOT MATCH GCHART- INPUT WAS #
Explanation: The GROUP variable in the Annotate data set is character, and the GCHART group is numeric or vice versa.

User Action: Check for misspelling or wrong variable assignment, or check for quotation marks in the assignment statement.

ANNOTATE SUBGROUP DATATYPE DOES NOT MATCH GCHART- INPUT WAS #
Explanation: The SUBGROUP variable in the Annotate data set is character, and the GCHART subgroup is numeric or vice versa.

User Action: Check for misspelling or wrong variable assignment, or check for quotation marks in the assignment statement.

BOTH OLD AND NEW VARIABLE NAMES ENCOUNTERED IN ANNOTATE= DATA SET
Explanation: Variables named both MIDPOINT and MIDPNT or SUBGROUP and SUBGRP occur in the Annotate data set.

User Action: Determine which variable has the proper values for the Annotate data set and either delete the other variable or rename MIDPNT to MIDPOINT and SUBGRP to SUBGROUP.

CALCULATED COORDINATES LIE COMPLETELY OFF THE VISIBLE AREA
Explanation: Both the *x* and *y* coordinates supplied are outside the visible display area.

User Action: Check for improper or inappropriate coordinate system specification or coordinates out of range.

CANNOT HAVE MISSING GROUP VALUE IF GROUPS ARE PRESENT
Explanation: The GROUP variable in the Annotate data set contains a missing value.

User Action: If the GROUP= option is specified in the GCHART procedure, the Annotate GROUP variable cannot contain missing values. Remove the missing value from the request. Check reference system for data-dependent request.

CANNOT HAVE SUBGROUP AND X/Y MISSING IN GCHART STREAM
Explanation: Data coordinate system was requested and the X, Y, and SUBGROUP variables contain missing values.

User Action: The X, Y, or SUBGROUP variable must have a value if a data coordinate system is requested. Check stream for improper request.

CANNOT OMIT GROUP VARIABLE IF GCHART GROUPS ARE PRESENT
Explanation: You used a data coordinate system and specified GROUP= in the GCHART procedure, but the Annotate data set does not contain the GROUP variable.

User Action: Supply the GROUP variable in the Annotate data set.

CHARACTER VALUE SHOWN IS NOT ON THE HORIZONTAL AXIS
Explanation: The specified value of the XC variable is not on the *x* axis of the graph or chart. The observation is ignored.

User Action: Check for misspelling, for uppercase or lowercase conflict, or for exclusion in an axis specification.

CHARACTER VALUE SHOWN IS NOT ON THE VERTICAL AXIS
Explanation: The specified value of the YC variable does not occur on the *y* axis of the graph or chart. The observation is ignored.

User Action: Check for misspelling, for uppercase or lowercase conflict, or for exclusion in an axis specification.

CONFLICT BETWEEN PROCEDURE AXIS TYPE AND ANNOTATE DATA TYPE
Explanation: The axis type is character and the *x* and *y* coordinates are numeric or vice versa.

User Action: Check values for proper type matching.

DATA SYSTEM NOT SUPPORTED FOR THIS STATEMENT
Explanation: The data coordinate systems 1, 2, 7, 8 are not permitted for this statement.

User Action: Choose a different reference system for this observation.

DATA SYSTEM REQUESTED, BUT POINT IS NOT ON GRAPH
Explanation: The coordinate specified is not on displayed graph, and data coordinate system placement has been requested.

User Action: Check for improper specification of data value or graph axis parameters, or incorrect system specification. If this occurs, you might be able to use percent of the data area to position Annotate graphics.

G3D DATA SYSTEM REQUESTED, ALL SYSTEMS NOT DATA DEPENDENT
Explanation: Not all requested XSYS, YSYS, and ZSYS variable values are data values.

User Action: If one variable in G3D annotation is data-dependent, all variables must be data-dependent. Either specify all points in the data coordinate system or use another reference system value.

G3D DATA SYSTEM REQUESTED, VARIABLE CONTAINED MISSING VALUE
Explanation: The X, Y, or Z variable contained a missing value.

User Action: All values in G3D data placement requests must be specified. Remove the missing value from the request.

INTERNAL SYSTEM STACK OVERFLOW- TOO MANY PUSH FUNCTIONS
Explanation: The limit of stack positions has been exhausted. The maximum number of stack positions is system-dependent. Each PUSH operation uses one position; each POP frees one position for reuse.

User Action: Rewrite the program section to decrease the number of values stored in the stack.

INTERNAL SYSTEM STACK UNDERFLOW- TOO MANY POP FUNCTIONS
Explanation: The POP function has been issued with no values in the LIFO stack.

User Action: Check for unequal numbers of PUSH versus POP functions. They can be unequal, but you cannot move more values with the POP function than are stored with the PUSH function. At least one PUSH must occur before a POP can be issued.

LABEL FUNCTION REQUESTED, BUT TEXT VARIABLE NOT ON DATA SET
Explanation: A TEXT variable has not been found for the LABEL function.

User Action: If FUNCTION='LABEL', the TEXT variable must contain the string to be placed in the display area. Check for misspelling of variable name or specification of the wrong Annotate data set.

LINE VALUE SPECIFIED IS NOT WITHIN LIMITS- 0<=L<=3
Explanation: An invalid special line value has been specified.

User Action: The LINE value specified was not acceptable for FUNCTION='BAR' or the RECT macro. Check function for definition of line values or previous value used in DATA step prior to this observation.

LINE VALUE SPECIFIED IS NOT WITHIN LIMITS- 1<=L<=46
Explanation: The LINE value specified is not in the range 1 through 46.

User Action: Check for improper specification of data value. Line styles represented by the LINE values can be found in the “Specifying Line Types” on page 502.

MINIMUM VARIABLES NOT MET--AMBIGUITY PREVENTS SELECTION.
Explanation: The combinations of available X, Y, XC, YC, GROUP, MIDPOINT, and SUBGROUP variables do not identify the data-dependent values uniquely.

User Action: Check variable requirements and respecify.

MINIMUM VARIABLES NOT MET- MUST HAVE X/XC,Y/YC IN DATA SET
Explanation: The X, XC, Y, or YC variables have not been found in the Annotate data set.

User Action: The X or XC and Y or YC variables must be in the data set. This message represents a minimum validity check of the supplied Annotate data set.

POLYCONT ENCOUNTERED BEFORE POLY
Explanation: The POLYCONT function was encountered with no POLY function specification.

User Action: Probable sequencing error. Check for missing POLY command, improper ordering of polygon points, or interruption of POLY type commands by other valued functions. Also, check the value of WHEN for a mismatch.

“POLYCONT” INTERRUPTED
Explanation: A POLYCONT definition has been interrupted and resumed in the Annotate data set. This usually accompanies the error message `POLYCONT ENCOUNTERED BEFORE POLY`

User Action: Check data stream for proper order.

POSITION VALUE INVALID- MUST BE ONE OF “0123...9ABCDEF”
Explanation: The value of the POSITION variable is not in range '0' through '9' or 'A' through 'F' or '<', '+', or '>' in a LABEL command.

User Action: Check desired value in POSITION description and correct.

REQUESTED POLYGON CONTAINS TOO MANY VERTICES (OBSERVATIONS)
Explanation: The maximum allocation for polygon points is exhausted. The maximum number of vertices is limited by a device's memory.

User Action: Define polygon with fewer points or break polygon into sections.

SYSTEM VALUE INVALID- MUST BE ONE OF “0123...9ABC”
Explanation: The value supplied for the XSYS, YSYS, or HSYS variable is not valid.

User Action: Check the desired value and correct the data set.

TEXT STRING EXTENDS BEYOND BOUNDARY OF SYSTEM DEFINED
Explanation: The text string is too long.

User Action: Check for excessive SIZE value or shorten the string. This error could be caused by HSYS='4' and a small value of the VPOS graphics option.

USE THE XC VARIABLE FOR DATA VALUES WHEN TYPE IS CHARACTER
Explanation: The X variable is character type in the Annotate data set when it should be numeric.

User Action: If character data are being plotted, use the XC variable to specify any data-related points pertaining to character values. If data are not character, omit quotation marks in X data value assignment.

USE THE YC VARIABLE FOR DATA VALUES WHEN TYPE IS CHARACTER
Explanation: The Y variable is character type in the Annotate data set when it should be numeric.

User Action: If character data are being plotted, use the YC variable to specify any data-related points pertaining to character values. If data are not character, omit quotation marks in Y data value assignment.

VALUE SHOWN IS NOT A VALID FONT OR PATTERN TYPE
Explanation: The value of the STYLE variable is not a valid font or pattern.

User Action: Check the value supplied for misspelling, truncation, and support in the FUNCTION description.

VALUE SHOWN IS NOT A VALID FUNCTION
Explanation: The value in the FUNCTION variable is not recognized as an available function.

User Action: Check for misspellings or truncation of value. Truncation can be corrected by specifying a length of 8 bytes in the LENGTH statement in the DATA step that generates the data set.

VALUE SHOWN IS NOT A VALID SIZE FACTOR
Explanation: The SIZE value of the variable is negative or excessive.

User Action: Check request or calculation for positive value result.

VARIABLE SHOWN HAS IMPROPER LENGTH IN ANNOTATE= DATA SET
Explanation: The length is incorrect for variable indicated. Either the length of the character string exceeds the length for the variable specified in a LENGTH statement, or the variable was not specified in a LENGTH statement.

User Action: Make sure the variable length is defined in a length statement and that the length specified adequately covers the length of the character strings that are used.

VARIABLE SHOWN IS NOT OF THE PROPER DATA TYPE

Explanation: The data type does not match required type for variable listed. Either variable type is character where a numeric is required, or numeric where a character is required.

User Action: Specify proper type for variable as described in "About the Annotate Variables" on page 804.

Examples

Example 1: Labeling Subgroups in a Vertical Bar Chart

Features: Annotate Function
LABEL
Annotate Variables
MIDPOINT
POSITION
SUBGROUP

Sample library member: GANVBAR

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example shows how to annotate subgroup labels 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 units ordered 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 36, "GCHART Procedure," on page 983 for more information about creating bar charts.

Output 28.2 Bar Chart with Annotated Labels on Subgroups

Program

```
goptions reset=all border;

data sold;
   length type $ 10;
   input city $ units type $ ;
   datalines;
Atlanta  99 Printers
Atlanta 105 Plotters
Atlanta  85 Terminals
Paris   182 Printers
Paris   150 Plotters
Paris   157 Terminals
Sydney  111 Printers
Sydney  136 Plotters
Sydney  100 Terminals
;
run;

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

title "Orders Received";
footnote j=r "GANVBAR";
```

```
axis1 label=none major=none minor=none style=0
      value=none;
axis2 label=none;

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

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set SOLD.

```
data sold;
   length type $ 10;
   input city $ units type $ ;
   datalines;
Atlanta  99 Printers
Atlanta 105 Plotters
Atlanta  85 Terminals
Paris   182 Printers
Paris   150 Plotters
Paris   157 Terminals
Sydney  111 Printers
Sydney  136 Plotters
Sydney  100 Terminals
;
run;

data barlabel;
   length color style $ 8;
   retain color "white" when "a" style "arial"
      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 j=r "GANVBAR";
```

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 2: Drawing a Circle of Stars

Features:

Annotate Functions
- BAR
- CNTL2TXT
- FRAME
- LABEL
- MOVE
- PIECNTR
- PIEXY
- SYMBOL

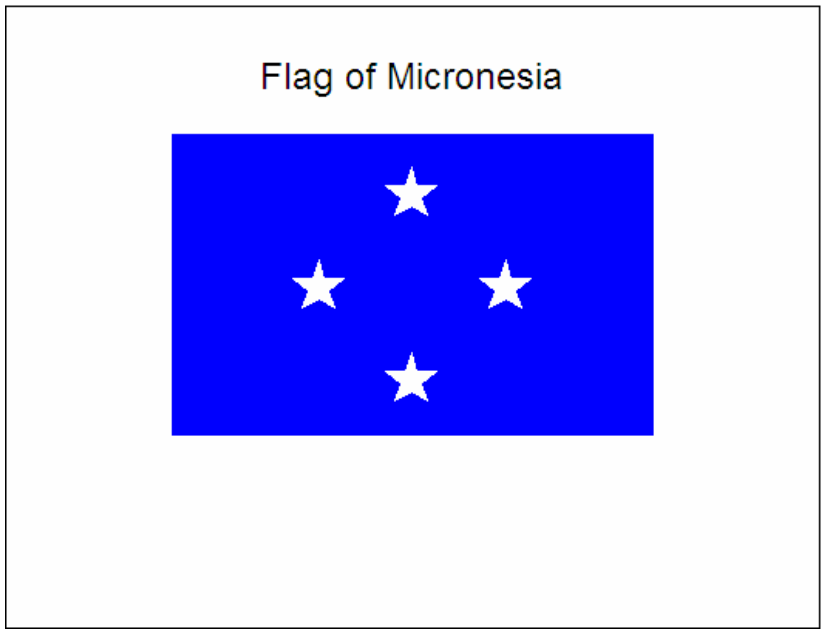
Annotate Variables
- COLOR
- HSYS, XSYS, YSY
- LINE
- STYLE
- TEXT
- X and Y
- XLAST and YLAST
- XLSTT and YLSTT

Sample library member: GANCIRCL

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example shows how to use an Annotate data set to draw a flag that consists 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.

Output 28.3 *Stars Positioned in a Circle with GANNO*



Program

```
goptions reset=all border;

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

   function="frame"; output;

   function="label"; x=50; y=90; text="Flag of Micronesia";
      style=""; size=6; output;

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

do star_ang=0 to 270 by 90;

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

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

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

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

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

```
   function="label"; x=50; y=90; text="Flag of Micronesia";
      style=""; 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 SYMBOL 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="symbol"; 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;
```

29

Annotate Function Dictionary

About the Annotate Functions

In an Annotate data set, the value of the FUNCTION variable specifies what action the observation performs. Annotate functions act in conjunction with Annotate variables that determine where and how to perform the action. Many of these variables are function-dependent, that is, what they do depends on the function that

they are used with. For example, with the LABEL function the STYLE variable specifies a font; with the BAR function, STYLE specifies a pattern.

This section describes all of the values of the FUNCTION variable. For each function it

- describes the function's action.
- notes whether the function updates the internal coordinate variables XLAST, YLAST and XLSTT, YLSTT.
- describes how other Annotate variables behave with the function. For a complete description of each variable, see "About the Annotate Variables" on page 804

For a summary of drawing and programming tasks performed by the FUNCTION variable, see Table 28.2 on page 731.

The variables that are available for use with each function are listed in Figure 28.3 on page 730 .

Dictionary

ARROW Function

Draws an arrow in the graphics output from the (XLAST, YLAST) coordinates to the (X,Y) coordinates specified in the function.

Updates: XLAST, YLAST

Tip: For best results, specify a graphics device driver in the GOPTIONS statement.

Syntax

FUNCTION='ARROW';

Associated Variables

ANGLE=*angle-value*

specifies the angle for the tip of the arrowhead. You can specify any number for the angle. If the angle that you specify is not between 0 and 180, the absolute value of mod(*angle-value*,180) is used. For example, the values -45, 45, and 225 all produce the same result.

Default 30

See "ANGLE Variable" on page 804

COLOR='*color*'
specifies the color of the arrow that is being drawn. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807.

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable.

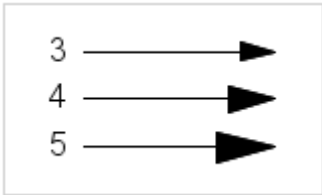
See "HSYS Variable" on page 813

HTML='*link-string*'
specifies the text that defines the drill-down link. The drill-down information is applied to the arrowhead.

See "HTML Variable" on page 814

LINE=*length*
specifies the length of the sides of the arrowhead as a percentage of the graphics area, regardless of the value for HSYS.

Figure 29.1 *Sample Filled Arrowhead Sizes Used with the SIZE Variable*

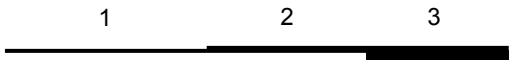


Default 1

See "LINE Variable" on page 821

SIZE=*line-thickness*
specifies the thickness of the line that is being drawn. The units depend on the value of the HSYS variable. For example, if HSYS='3', the SIZE variable is in units of percent of the graphics output area. If HSYS='4', the SIZE variable is in units of cells of the graphics output area.

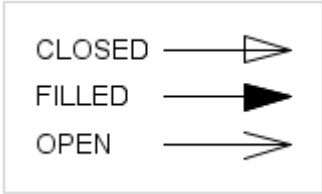
Figure 29.2 *Sample Line Thicknesses Used with the SIZE Variable*



See "SIZE Variable" on page 829

STYLE='CLOSED' | 'FILLED' | 'OPEN'
specifies the arrowhead style.

Figure 29.3 *Arrowhead Styles CLOSED, FILLED, and OPEN*



Default OPEN

See "STYLE Variable: Arrows" on page 832

WHEN='B' | 'A'
specifies when to draw the arrow in relation to other procedure output.

B
specifies that the arrow is drawn before the other procedure output is drawn.

A
specifies that the arrow is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the endpoint of a line drawn from (XLAST, YLAST) to (X,Y).

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. The XC variable can be used only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. The YC variable can be used only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

BAR Function

Draws a rectangle whose lower left corner is defined by the internal variables (XLAST, YLAST) and whose upper right corner is defined by the specified X, Y variable pair. You can define the color of the fill, the fill pattern, and the edge lines to be drawn.

Alias: BOX

Updates: XLAST, YLAST

Syntax

FUNCTION='BAR';

Associated Variables

COLOR='*color*'
specifies the color of either the interior of the bar or the outline of the bar. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807. The part of the bar affected depends on the value of the STYLE variable. If STYLE specifies a pattern or fill, the COLOR variable determines the color of the interior. If STYLE specifies an empty pattern, the COLOR variable determines the color of the outline of the bar.

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

HTML='*link-string*'
specifies the text that defines the drill-down link.

See "HTML Variable" on page 814

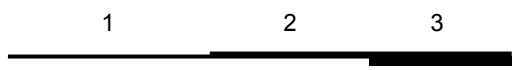
LINE=0–3
specifies the direction in which to adjust the outline of the bar. Use LINE values 1 and 2 to offset a particular bar from an axis or adjoining area.

Default 1

See "LINE Variable" on page 821

SIZE=*line-thickness*
specifies a line thickness for the rectangle

Figure 29.4 *Sample Line Thicknesses Used with the SIZE Variable*



See "SIZE Variable" on page 829

STYLE='SOLID' | 'EMPTY' | *style<density>*
specifies the pattern that fills the bar.

SOLID
specifies a solid color fill.

EMPTY
specifies no fill.

style<density>
specifies a fill style and density.

Default EMPTY

Restriction Java and ActiveX support only SOLID and EMPTY.

See "STYLE Variable: Patterns" on page 833

WHEN='B' | 'A'
specifies when to draw the bar in relation to other procedure output.

B
specifies that the bar is drawn before the other procedure output is drawn.

A
specifies that the bar is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
define the upper right corner of a bar (rectangle) whose lower left corner is (XLAST,YLAST). Figure 29.76 on page 765 illustrates the use of these coordinates.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. The XC variable can be used only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for Y or YC variable. The YC variable can be used with YSYS='2' only.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

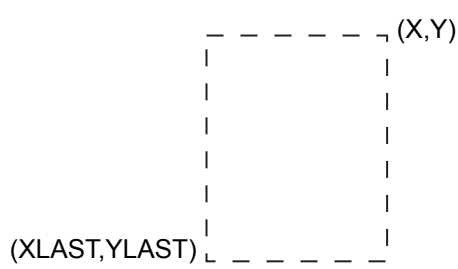
Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

Details

Figure 29.76 on page 765 shows how the XLAST, YLAST, and X, Y variables define the diagonal corners of the bar. With character data, the XC and YC variables are used in place of the X and Y variables. The values of the XLAST and YLAST variables are usually initialized with a MOVE function or another function that updates the XLAST and YLAST pair. When the XC variable is used, set XSYS='2'. When the YC variable is used, set YSYS='2'.

Figure 29.5 *Points Used to Construct a Bar*



CNTL2TXT Function

Copies the values of the internal coordinates stored in the variable pairs (XLAST, YLAST) to (XLSTT, YLSTT).

Updates: XLSTT, YLSTT

Syntax

FUNCTION='CNTL2TXT';

Details

You can use CNTL2TXT to calculate the position of labels on a graph. For example, the following DATA step uses CNTL2TXT to position a pie slice label in the center of the arc and just beyond the arc itself, as shown in Figure 29.79 on page 767.

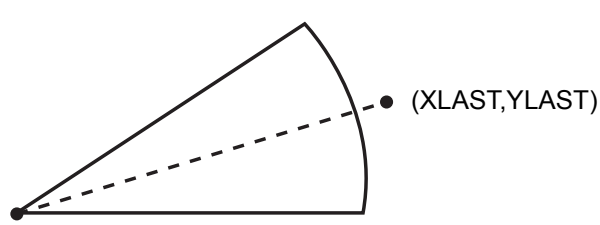
First, use the PIE function to draw the pie slice:

```
data pielabel;
   retain xsys ysys "3";
   length function style $ 8;
   function="pie"; size=20; x=30; y=30;
      style="empty"; rotate=45; output;
```

Then use the PIEXY function to calculate a point outside of the arc as shown in Figure 29.77 on page 766.

```
/* find a point that is half of the arc (rotate*.5) */
      /* and is 4 units beyond the radius (size=1.1) */
   function="piexy"; angle=rotate*.5; size=1.1; output;
```

Figure 29.6 *Position Calculated with the PIEXY Function*

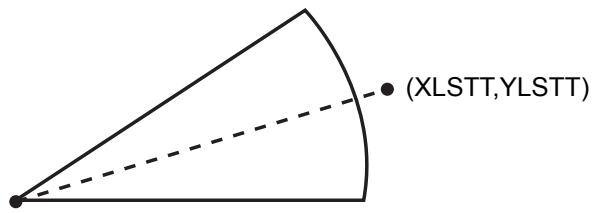


At this point, the XLAST and YLAST variables contain the coordinates of the point that is calculated by PIEXY. However, (XLAST, YLAST) cannot be used directly by text functions. Use CNTL2TXT to copy the coordinates in (XLAST, YLAST) to the XLSTT and YLSTT variables, which text functions can use. Figure 29.78 on page 767 shows the results.

```
function="cntl2txt";
```

```
output;
```

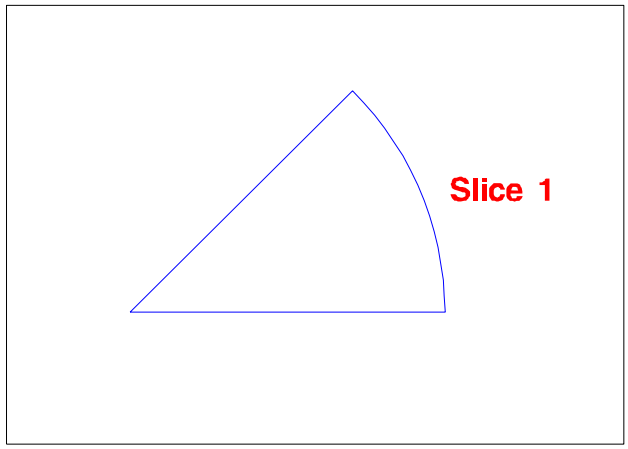
Figure 29.7 *Coordinates After Using the CNTL2TXT Function*



Now you can use the LABEL function to write the label as shown in Figure 29.79 on page 767. Specify missing values for the X and Y variables to force LABEL to use the XLSTT and YLSTT variables instead of the X and Y variables.

```
/* write the label "Slice 1" and position it to     */
      /* the right of the point stored in XLSTT and YLSTT */
   function="label"; text="Slice 1"; angle=0; rotate=0;
      position="6"; style="swissb"; size=4; x=.; y=.;
      output;
run;
   /* draw the Annotate graphics */
proc ganno anno=pielabel;
run;
quit;
```

Figure 29.8 *Labeled Pie Slice*



COMMENT Function

Inserts comments within the Annotate data set. The observations generated by the COMMENT function are ignored when the data set is processed.

Syntax

FUNCTION='COMMENT';

Associated Variables

TEXT='*text-string*'
specifies the comment to write to the data set.

See "TEXT Variable" on page 837

DEBUG Function

Writes the values of internal coordinates and Annotate variables to the SAS log before and after processing the next command (unless it is DEBUG) in the Annotate DATA step.

Syntax

FUNCTION='DEBUG';

DRAW Function

Draws a line in the graphics output from the (XLAST, YLAST) coordinates to the (X, Y) coordinates specified in the function.

Updates: XLAST, YLAST

Syntax

FUNCTION='DRAW';

Associated Variables

COLOR='*color*'
specifies the color of the line that is being drawn. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807.

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable.

See "HSYS Variable" on page 813

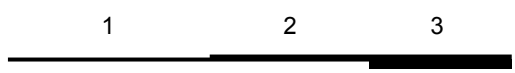
LINE=1–46
specifies the line type of the line that is being drawn.

See "LINE Variable" on page 821

"Specifying Line Types" on page 502 for an illustration of the line types.

SIZE=*line-thickness*
specifies the thickness of the line that is being drawn. The units depend on the value of the HSYS variable. For example, if HSYS='3', the SIZE variable is in units of percent of the graphics output area. If HSYS='4', the SIZE variable is in units of cells of the graphics output area.

Figure 29.9 *Sample Line Thicknesses Used with the SIZE Variable*



See "SIZE Variable" on page 829

WHEN='B' | 'A'
specifies when to draw the line in relation to other procedure output.

B
specifies that the line is drawn before the other procedure output is drawn.

A
specifies that the line is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the endpoint of a line drawn from (XLAST, YLAST) to (X,Y).

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. The XC variable can be used only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. The YC variable can be used only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

DRAW2TXT Function

Draws a line from (XLAST, YLAST) to (XLSTT, YLSTT) without updating any of those variables.

Syntax

FUNCTION='DRAW2TXT';

Associated Variables

COLOR='*color*'
specifies the line color. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807.

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable.

See "HSYS Variable" on page 813

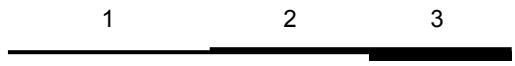
LINE=1–46
specifies the line type of the line that is being drawn.

See "LINE Variable" on page 821

"Specifying Line Types" on page 502 for an illustration of the line types.

SIZE=*line-thickness*
specifies the thickness of the line that is being drawn.

Figure 29.10 *Sample Line Thicknesses Used with the SIZE Variable*



See "LINE Variable" on page 821

"DRAW Function" on page 768 for details

WHEN='B' | 'A'
specifies when to draw the line in relation to generation of the procedure output.

B
specifies that the line is drawn before the other procedure output is drawn.

A
specifies that the line is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

Details

DRAW2TXT is useful for underlining text.

DRAW2TXT does not update the (XLAST, YLAST) or (XLSTT, YLSTT) coordinates; neither can it interrupt a POLYCONT sequence.

FRAME Function

Draws a border around the portion of the display area defined by the XSYS and YSYS variables. Can specify a background color for the framed area.

Note: The FRAME function is not supported by Java.

Syntax

FUNCTION='FRAME';

Associated Variables

COLOR='*color*'
specifies the frame color and, if the STYLE variable is specified, fills the interior of the frame. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807.

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable.

See "HSYS Variable" on page 813

HTML='*link-string*'
specifies the text that defines the drill-down link.

See "HTML Variable" on page 814

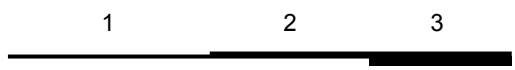
LINE=1–46
specifies the line type with which to draw the frame.

See "LINE Variable" on page 821

"Specifying Line Types" on page 502 for an illustration of the line types

SIZE=*line-thickness*
specifies the thickness of the line with which to draw the frame.

Figure 29.11 *Sample Line Thicknesses Used with the SIZE Variable*



See "SIZE Variable" on page 829

"DRAW Function" on page 768 for details.

STYLE='SOLID' | 'EMPTY' | *style<density>*
specifies the pattern that fills the area that is bounded by the frame.

SOLID
specifies a solid color fill.

EMPTY
specifies no fill.

style<density>
specifies a fill style and density.

Default EMPTY

Restriction Java and ActiveX support only SOLID and EMPTY.

See "STYLE Variable: Patterns" on page 833

the discussion of fill patterns for bars and blocks in the documentation for the VALUE= option for the "PATTERN Statement" on page 455.

WHEN='B' | 'A'
specifies when to draw the frame in relation to other procedure output.

B
specifies that the frame is drawn before the other procedure output is drawn.

A
specifies that the frame is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

XSYS='*coordinate-system*'
YSYS='*coordinate-system*'
define the area to be enclosed by the frame.

See "XSYS Variable" on page 843

"YSYS Variable" on page 849

Details

Use FRAME to simulate the CBACK= graphics option on devices (such as plotters) that do not support that option. For devices that do support the CBACK= graphics option, FRAME works in addition to that option. FRAME does not alter the (XLAST, YLAST) coordinates. See "CBACK Device Parameter and Graphics Option" on page 590 for more information about CBACK=.

IMAGE Function

Displays an image in the graphics output from the current (X,Y) coordinates to the (X, Y) coordinates that are associated with the IMGPATH variable.

Updates: XLAST, YLAST

Syntax

FUNCTION='IMAGE';

Associated Variables

HTML='*link-string*'
specifies the text that defines the drill-down link.

See "HTML Variable" on page 814

IMGPATH= '*external-file*'
specifies the image file to be displayed in the graphics output. The syntax of external file specifications varies across operating environments.

Note Copying and pasting the image works only if an absolute path is specified instead of a relative path, or if the file into which the image is being pasted is opened from the directory to which the image is relative.

See "IMGPATH Variable" on page 820

STYLE ='TILE' | 'FIT' | 'SINGLE'
specifies how the image is to be applied to fill the specified area of the graphics output.

TILE
replicates the image to fill the area.

FIT
stretches a single instance of the image to fill the area.

SINGLE
centers a single instance of the image on the specified coordinates.

Any part of the image that extends beyond the data area is clipped. SINGLE is valid starting with SAS 9.4M5.

Default TILE

See "STYLE Variable: Images" on page 832

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
specifies the coordinates that determine the size of the image displayed in the graphics output.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

Details

The following example shows how the IMAGE function adds a single stretched instance of an image to the graphics output. The image begins at the current coordinates and ends at the specified coordinates.

```
x=10; y=5; function="move"; output;
x=35; y=15; imgpath="/images/gifs/picture.gif";
style="fit";
function="image"; output;
```

For a list of the file types that you can use, see "Image File Types Supported by SAS/GRAPH" on page 384.

LABEL Function

Places text in the graphics output. Associated variables can control the color, size, font, base angle, and rotation of the characters displayed. You can also animate labels using the HTML= variable. When you specify the HTML= variable, you include the animation action and the parameters that control the

animation. The animated label feature is available only when you use one of the SVG device drivers, except SVGANIM.

Updates: XLSTT, YLSTT

Syntax

FUNCTION='LABEL';

Associated Variables

ANGLE=0–360
specifies the baseline angle of the character string with respect to the horizontal. The pivot point is at (X, Y), and the rotation is in a counterclockwise direction.

See "ANGLE Variable" on page 804

CBORDER='*color*' | 'CTEXT'
draws a border around the text of a specified color.

color
specifies the border color as any valid SAS/GRAPH color name. See "CBORDER Variable" on page 805.

CTEXT
specifies the text color as the border color.

CBOX='*color*' | 'CBACK'
draws a solid box of a specified color behind the text.

color
specifies the box color as any valid SAS/GRAPH color name. See "CBORDER Variable" on page 805.

CBACK
specifies the background color as the box color.

COLOR='*color*'
specifies the color of the text. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807.

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable.

See "HSYS Variable" on page 813

HTML='*link-string*'
specifies the text that defines the drill-down link.

See "HTML Variable" on page 814

HTML="ANIMATE=*animate-action*(*parameters*)"
animates the text, and specifies the text animation action and the parameters that control the animation.

See ""ANIMATE=*animate-action*(*parameters*)"" on page 815

POSITION='*text-position*' | '0'
controls the text string placement and alignment. *Text-position* can be one of the characters 1 through 9, A through F, <, +, or >. Invalid or missing values default to POSITION='5'. POSITION should always be a character variable of length 1.

See "POSITION Variable" on page 825

ROTATE=*rotation-angle*
specifies the rotation angle of each character in the string. It is equivalent to the ROTATE= option in the FOOTNOTE, NOTE, and TITLE statements.

See "ROTATE Variable" on page 828

SIZE=*height*
specifies the height of the text string. The SIZE variable units are based on the value of the HSYS variable.

See "SIZE Variable" on page 829

STYLE='*font-specification*' | 'NONE'
specifies the font with which to draw the text that is specified by the TEXT variable.

See "STYLE Variable: Fonts" on page 831

TEXT='*text-string*'
specifies the text to be written. *Text-string* can be up to 200 characters. Define the TEXT variable with sufficient length to contain all of the characters in your text string. If you need longer strings, use separate observations and POSITION='0' to continue the text.

See "TEXT Variable" on page 837

WHEN='B' | 'A'
specifies when to draw the text strings in relation to other procedure output.

B
specifies that the text strings are drawn before the other procedure output is drawn.

A
specifies that the text strings are drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the start point of the text string. You can also modify the placement of the text string with the POSITION variable.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. Use the XC variable only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. Use the YC variable only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

Details

Animating Annotation Text Labels

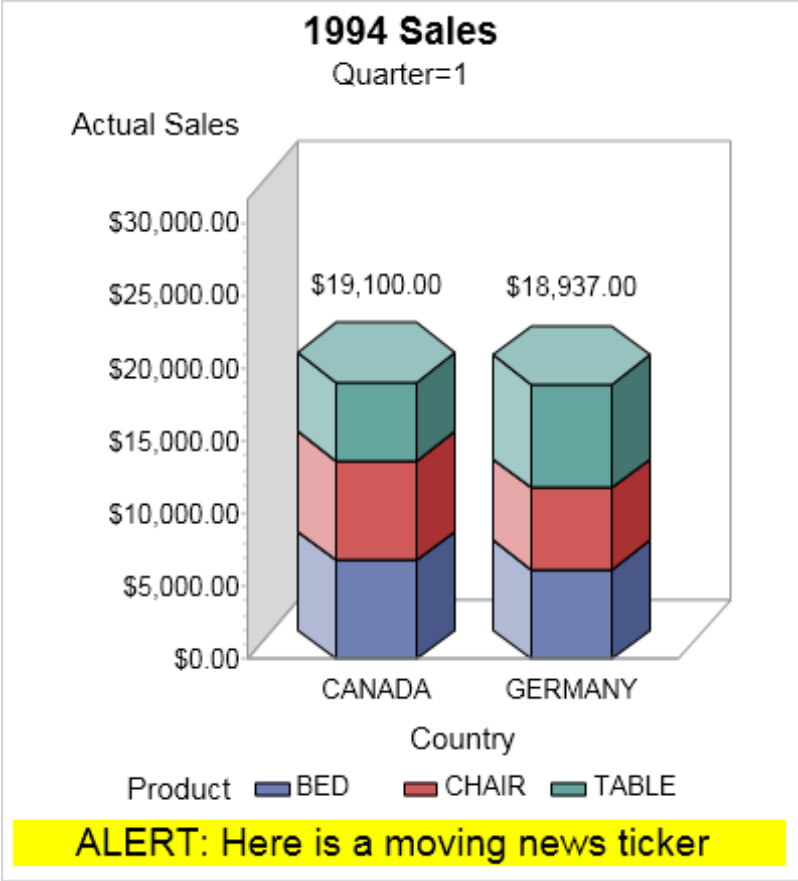
When you create text labels using the Annotate facility, you can animate those labels if you use one of the SAS/GRAPH SVG device drivers, except SVGANIM. Various animation actions are available, including moving text on a path, growing or shrinking text, moving text, and blinking text.

To animate labels, add the HTML variable to the LABEL function in the annotate data set. The HTML variable accepts attributes that specify the animation action and the parameters that control the animation.

In the following example, the HTML variable specifies that the text "ALERT: Here is a moving news ticker" moves continuously across the bottom of the screen. The code for this example can be found in Example Code 29.5.

The figure captures a still frame of the output.

Figure 29.12 *Example: Adding a Moving Text Annotation to Charts*



Here is the HTML variable statement that created the moving text.

```
html="animate=move('ALERT: Here is a moving news ticker',
  100, +0, 0, 5, repeat)";
```

The following items describe the parts of the statement:

- ANIMATE=MOVE(*parameters*) specifies the type of animation to perform. In this example, the MOVE animation is performed. The HTML variable specified must be enclosed in double quotation marks.
- *'ALERT: Here is a moving news ticker'* specifies the text that is moved. The text must be enclosed in single quotation marks. In addition, the text must match the text string that is provided for the TEXT variable in the LABEL function.
- *100* specifies the X coordinate that the text moves to in percentage units of the display or the SVG space.
- *+0* specifies the Y coordinate that the text moves to in percentage units of the display or the SVG space. The + indicates that the text moves the specified amount relative to the Y value that is specified for the label.
- *0, 5* specifies that the animation begins 0 seconds after the SVG page has been loaded and has a duration of 5 seconds.
- REPEAT specifies that the animation repeats continuously.

The attributes that determine the text color, background color, and so on, are specified separately in the data set.

Here are the main steps for producing the graph in the example.

1 In your annotation data set, after you specify FUNCTION='LABEL', add the HTML variable that specifies the animation action and associated parameters.

   For complete syntax, see "HTML Variable" on page 814.

2 Use the GOPTIONS statement to specify one of the SVG device drivers, except SVGANIM.

```
goptions dev=svgview;
```

3 Specify the ODS HTML5, ODS HTML, or the ODS LISTING destination.

> **TIP** If using one of the ODS HTML destinations, see "Browser Support for Viewing SVG Files" on page 169 to determine whether to use HTML5 or HTML.

4 After you have set the options and opened the ODS destination, proceed with your SAS code to create the graph. The graph code should be designed to display the annotation. For more information, see "Producing Graphics Output from Annotate Data Sets" on page 741.

SAS/GRAPH generates the output as an SVG graphic. To view the output, open the SVG file in a browser.

Note: If your browser does not render the graphic, see "Browser Support for Viewing SVG Files" on page 169.

The following code produces the output shown in the previous example.

Example Code 29.1 *Code for Adding a Moving Text Annotation to Charts*

```
/* Create a file reference for the ODS output */
filename odsout ".";

/* Specify the graphics options */
options nodate nonumber nofontembedding;
goptions reset=all device=svgview;
ods _all_ close;
ods html path=odsout file="animlabel.html";

/* Create the annotation data set */
/* that includes animation for the label */
data moving;
length color $ 8;
xsys='3'; ysys='3'; hsys='D'; position='6'; function='label';
x=0; y=5; color='black'; style='Arial'; size=18;
text='ALERT: Here is a moving news ticker';
html="animate=move('ALERT: Here is a moving news ticker',
  100, +0, 0, 5, repeat)";
output;
function='move'; x=0; y=3; hsys='3';
text=''; html='';  output;
```

```
function='bar'; x=100; y=5; hsys='3'; style='solid'; line=0;
  color='yellow';
output;
run;

/* Create the graph data by extracting information for */
/* Canada and Germany from sashelp.prdsale. */
data work.qsales;
set sashelp.prdsale(where=(country="CANADA" or country="GERMANY")
keep=Actual Country Product Quarter Year);
run;

/* Sort the data by quarter */
proc sort data=work.qsales;
by quarter;
run;

/* Generate the first set of graphs */
title1 "1993 Sales";
footnote h=2 ' ';
axis1 order=(0 to 30000 by 5000);
proc gchart data=work.qsales(where=(year=1993)) anno=moving;
vbar3d country / sumvar=actual subgroup=product sum raxis=axis1
shape=hexagon;
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;


/* Generate the second set of graphs */
title1 "1994 Sales";
proc gchart data=work.qsales(where=(year=1994)) anno=moving;
vbar3d country / sumvar=actual subgroup=product sum raxis=axis1
shape=hexagon;
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;

ods html close;
ods html; /* Not required in SAS Studio */
```

You can also use Base SAS Universal Printing to generate SVG output. See "Creating SVG Documents Using Universal Printing" in *SAS 9.4 Universal Printing*.

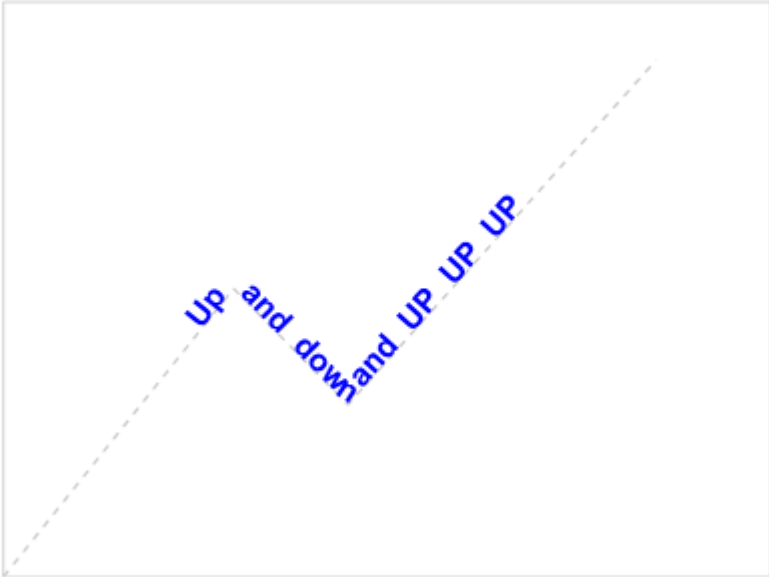
For another example of label text animation, see "Example: Moving Text on a Path" on page 781.

See Also

- Chapter 13, "Using SVG Graphics," on page 159
- Chapter 28, "Using Annotate Data Sets," on page 721
- "LABEL Function" on page 774

Example: Moving Text on a Path

This example moves text on a path that is defined in the annotate data set using the DRAW function. The path moves from left to right in a zig-zag motion, first up, then down, and back up again. The following figure shows a snapshot of the moving text.



In the figure, the path defined by the DRAW function is made visible so that you can see how the text moves along the defined text path. Here is the code for this example. In the output from this code, the path that is defined by the DRAW function is completely transparent.

```
filename odsout ".";

data annotp;
length function $8 color $ 10 style $ 12 text $ 50 html $ 100;
xsys='3'; ysys='3'; hsys='D';

/* Label observation to create moving text on a path */
function='label'; x=0; y=50; size=12; color='blue'; style='Arial/bold';
text='Up  and  down  and  UP  UP  UP';
html="animate=textpath('Up  and  down  and  UP  UP  UP', 0, 100, 0, 5,
repeat)";
output;

/* Move and Draw observations to create transparent line */
function='move'; x=0; y=0; color='AFFFFFF00'; html=''; text='';
size=.; style='';
hsys=''; position=''; output;
function='draw'; x=30; y=50; color='AFFFFFF00'; output;
function='draw'; x=45; y=30; color='AFFFFFF00'; output;
function='draw'; x=85; y=90; color='AFFFFFF00'; output;

run;

ods _all_ close;
ods html path=odsout file="movingtext.html";
```

```
options nofontembedding;
goptions reset=all dev=svgview vsize=3in hsize=4in;

proc gslide anno=annotp imagemap=imap;
run;
quit;

ods html close;
ods html; /* Not required in SAS Studio */
```

MOVE Function

Moves the drawing pointer to a specific location without drawing a line.

Updates: XLAST, YLAST

Syntax

FUNCTION='MOVE';

Associated Variables

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

WHEN='B' | 'A'
specifies when to perform the move in relation to other procedure output.

B
specifies that the move is performed before the other procedure output is drawn.

A
specifies that the move is performed after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the coordinates to which the pen is to be moved.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. Use the XC variable only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. Use the YC variable only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

Details

Use MOVE to prepare for a DRAW command, a BAR command, or programming functions.

PIE Function

Draws pie slices in the graphics output.

Updates: XLAST, YLAST to coordinates for center of the slice.

Syntax

FUNCTION='PIE';

Associated Variables

ANGLE=*starting-angle*
specifies the starting angle of the slice arc. The default is 0.00 (horizontal) if the ANGLE variable is not specified for the first slice. After the first slice, the default is the ending angle of the slice arc just drawn if ANGLE=. (missing). Therefore, you can specify consecutive pie slices more easily by omitting the start and end calculations that are otherwise required. If you want the next slice to start at an angle that is different from the ending angle of the previous slice, specify a value for the ANGLE variable.

See "ANGLE Variable" on page 804

COLOR='*color*'
specifies the color of the pie slice, if a pattern is specified in the STYLE variable. If you specify STYLE='EMPTY', the COLOR variable also specifies the outline color of the pie slices. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807.

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable.

See "HSYS Variable" on page 813

HTML='*link-string*'
specifies the text that defines the drill-down link.

See "HTML Variable" on page 814

LINE=0–3
specifies which slice line (or lines) to draw.

See "LINE Variable" on page 821

ROTATE=*rotation-angle*
specifies the angle of rotation or the delta angle of the slice arc. For example, if you specify these statements, the slice arc that is drawn begins at 90 degrees (vertical) and ends at 135 degrees (90+45):

```
function="pie"; angle=90; rotate=45; output;
```

The ANGLE variable is internally updated to the end value, 135 degrees. The value is modified only internally. If a second PIE is used and the ANGLE variable contains a missing value, the start angle is assumed to be the previous end, or 135 degrees. The arc continues from that point.

If you specify the previous statements and then specify these statements, the slice begins at 135 degrees (the end angle from the previous slice). The slice extends another 45 degrees to the end point, 180 degrees.

```
function="pie"; angle=.; rotate=45; output;
```

This action repeats for every missing angle in the sequence.

Default 0

See "ROTATE Variable" on page 828

SIZE=*radius*

specifies the radius of the circle being drawn. The SIZE variable uses units that are determined by the HSYS variable.

See "SIZE Variable" on page 829

STYLE='PSOLID' | 'PEMPTY' | P*density<style<angle>>*

specifies the value of the pattern that fills the pie slices.

PSOLID
specifies a solid color fill.

PEMPTY
specifies no fill.

P*style<density>*
specifies the a fill style and density.

Default PEMPTY

Restriction Java and ActiveX support only PSOLID and PEMPTY.

See "STYLE Variable: Patterns" on page 833

the discussion of fill patterns for pie and star charts in the documentation for the VALUE= option for the "PATTERN Statement" on page 455.

WIDTH='*line-thickness*'

specifies the thickness of the outline around the pie slice.

See "WIDTH Variable" on page 839

WHEN='B' | 'A'

specifies when to draw the pie slice in relation to other procedure output.

B
specifies that the pie slice is drawn before the other procedure output is drawn.

A
specifies that the pie slice is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
define the center of the slice. The pivot point for all slices is the point referenced by X, Y, and Z (with the G3D procedure only). The first PIE command that is issued sets the center at the (X,Y) value. If subsequent values for X and Y are missing, the coordinates of the center point are used.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. Use the XC variable only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. Use the YC variable only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

See Also

"CNTL2TXT Function" on page 766

PIECNTR Function

Sets new center and radius values for later use by the PIEXY function but does not draw an arc.

Updates: XLAST, YLAST

Syntax

FUNCTION='PIECNTR';

Associated Variables

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable.

See "HSYS Variable" on page 813

SIZE=*radius*
specifies the new radius of the pie slice. The new radius is used by a subsequent PIEXY function.

Tip The HSYS variable determines the SIZE variable units.

See "SIZE Variable" on page 829

WHEN='B' | 'A'
specifies when to draw the pie slice in relation to other procedure output.

B
specifies that the pie slice is drawn before the other procedure output is drawn.

A
specifies that the pie slice is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
define the center and radius of the slice. All slices are referenced from that center.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. Use the XC variable only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. Use the YC variable only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

PIEXY Function

Calculates a point on the outline of the slice arc.

Updates: XLAST, YLAST

Syntax

FUNCTION='PIEXY';

Associated Variables

ANGLE=*rotation-angle*
specifies the angle of rotation when moving around the perimeter of a pie. The ANGLE variable determines the angle at which the point is located relative to 0 (the three o'clock position).

Default 0

See "ANGLE Variable" on page 804

SIZE=*radius-multiplier*
determines the distance from the center of the slice to the point that is being calculated. The point's distance is the current value of the SIZE variable multiplied by the radius (that is, the SIZE variable) of the previously drawn slice.

To position a graphics element inside the pie slice, set the SIZE variable to less than 1; to position it outside of the pie slice, set the SIZE variable to greater than 1. For example, if you specify these statements, the point calculated is 1.1 times the radius. (The radius is taken from the SIZE variable that is used with the previous FUNCTION='PIE' or FUNCTION='PIECNTR' observation.)

```
function="piexy"; size=1.1; output;
```

See "SIZE Variable" on page 829

WHEN='B' | 'A'

specifies when to update the internal coordinate pair (XLAST, YLAST) in relation to other procedure output.

B

specifies that XLAST and YLAST are updated before the other procedure output is drawn.

A

specifies that XLAST and YLAST are updated after the other procedure output is drawn.

See "WHEN Variable" on page 838

Details

PIEXY does not draw anything but places the calculated coordinates of the point in the internal coordinate pair (XLAST, YLAST). Then you can use XLAST and YLAST with other functions to perform other graphics actions, such as labeling pie slices. If you need to use the calculated position for a text function, use the SWAP or CNTL2TXT to put (XLAST, YLAST) into (XLSTT, YLSTT).

PIEXY assumes that a pie slice has been drawn or that FUNCTION='PIECNTR' has been used. Erroneous results can occur if a slice has not been drawn and PIEXY is invoked.

Figure 29.84 on page 789 shows a pie slice that is drawn with the PIE function. Figure 29.85 on page 790 shows a point beyond the arc that was calculated using the PIEXY function.

Figure 29.13 *Pie Slice Drawn with the PIE Function*

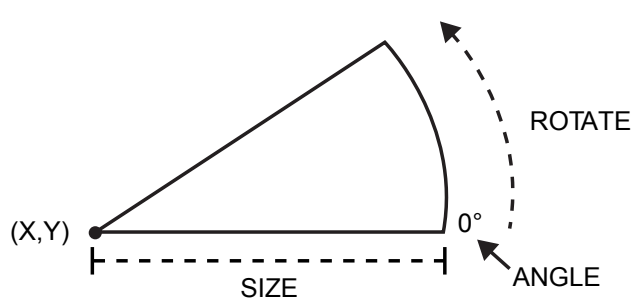
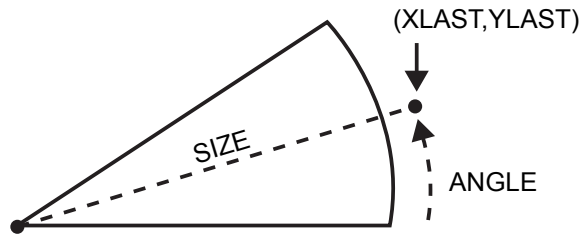


Figure 29.14 Point Calculated with the PIEXY Function



See Also

"CNTL2TXT Function" on page 766

POINT Function

Places a single point at the (X, Y) coordinates in the color that you specify. The point is one visible pixel in size.

Updates: XLAST, YLAST

Syntax

FUNCTION='POINT';

Associated Variables

COLOR='*color*'
specifies the color of the point to be drawn. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807.

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates when used with HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

WHEN='B' | 'A'
specifies when to draw the point in relation to other procedure output.

B
specifies that the point is drawn before the other procedure output is drawn.

A
specifies that the point is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the coordinates of the point that is to be drawn.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. Use the XC variable only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. Use the YC variable only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

POLY Function

Specifies the beginning point of a polygon. Associated variables can define the fill pattern and color, as well as the line type that outlines the polygon.

Syntax

FUNCTION='POLY';

Associated Variables

COLOR='*color*'
specifies the color of the interior of the polygon, if a pattern is specified for the STYLE variable. The outline color is specified with the POLYCONT function. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807.

HTML='*link-string*'
specifies the text that defines the drill-down link.

See "HTML Variable" on page 814

LINE=1–46
specifies the line type that outlines the polygon.

See "LINE Variable" on page 821

"Specifying Line Types" on page 502 for an illustration of the line types.

STYLE='MSOLID' | 'MEMPTY' | M*density<style<angle>>*
specifies the value of the pattern that fills the polygon.

MSOLID
specifies a solid color fill.

MEMPTY
specifies no fill.

M*density<style<angle>>*
specifies a fill style and density.

For example, if STYLE='MSOLID' for the POLY function, the fill area that is drawn by the POLYCONT sequence uses a solid fill. If STYLE='M5N15', the fill area uses a shaded fill of parallel lines. The *fill-pattern* value M5N15 specifies that the lines use the heaviest density, are parallel, and are drawn at a 15-degree angle from the horizontal.

Default MEMPTY

Restriction Java and ActiveX support only MSOLID and MEMPTY.

See "STYLE Variable: Patterns" on page 833

the discussion of fill patterns for maps in the documentation for the VALUE= option for the "PATTERN Statement" on page 455.

WHEN='B' | 'A'
specifies when to begin the polygon in relation to other procedure output.

B
specifies that the polygon begins before the other procedure output is drawn.

A
specifies that the polygon begins after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=depth-coordinate
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the initial point of the polygon that is being created.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. Use the XC variable only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. Use the YC variable only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

Details

Use POLY with POLYCONT to define and fill areas in the graphics output. POLY and POLYCONT do not update the (XLAST, YLAST) coordinates.

See Also

"POLYCONT Function" on page 794

POLYCONT Function

Continues drawing a polygon begun with the POLY function. POLYCONT specifies each successive point in the polygon definition.

Syntax

FUNCTION='POLYCONT';

Associated Variables

COLOR='*color*'
specifies the polygon outline color. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807. You can specify an outline color only with the first POLYCONT command in the sequence; all subsequent POLYCONT commands ignore the COLOR variable. If you do not specify a color, the POLYCONT function uses the interior color that was specified with the POLY function.

WHEN='B' | 'A'
specifies when to draw the polygon in relation to other procedure output.

B
specifies that the polygon is drawn before the other procedure output is drawn.

A
specifies that the polygon is drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify a point on the outline of the polygon that is being created.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X and XC variable.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y and YC variable.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

Details

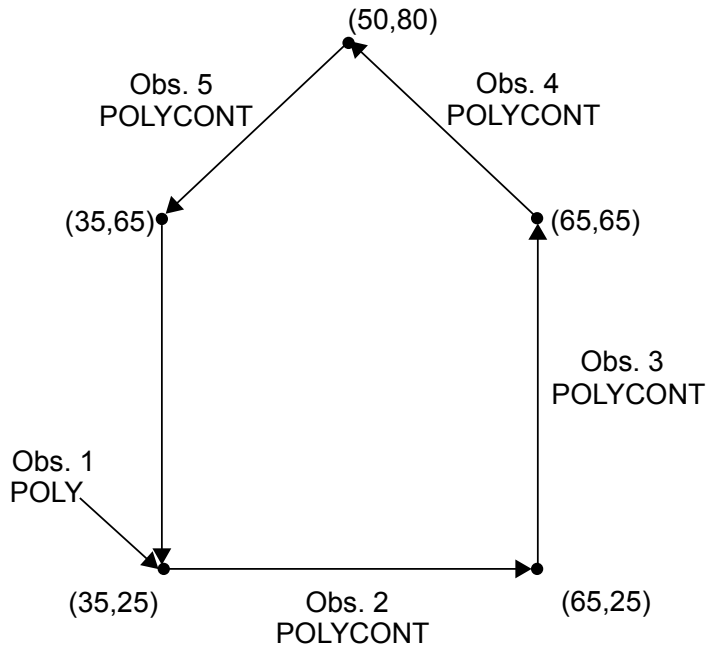
The polygon definition is terminated by a new POLY command or by any of these functions:

BAR	LABEL	PIEXY
DRAW	MOVE	POINT
DRAW2TXT	PIE	SYMBOL
FRAME	PIECNTR	

Use POLY and POLYCONT together to draw a polygon. The (X, Y) observation from the POLY function and the last (X, Y) observation from POLYCONT are assumed to connect. Thus, you are not required to respecify the first point. For example, these statements draw a pentagon like the one in Figure 29.86 on page 796:

```
data house;
   retain xsys ysys "3";
   length function $ 8;
      /* start at the lower left corner */
   function="poly"; x=35; y=25; output;
      /* move to the lower right corner */
   function="polycont"; x=65; y=25; output;
      /* move to the upper right corner */
   function="polycont"; x=65; y=65; output;
      /* move to the center top*/
   function="polycont"; x=50; y=80; output;
      /* move to the upper left corner and complete the figure */
   function="polycont"; x=35; y=65; output;
run;
proc ganno anno=house;
run;
quit;
```

Figure 29.15 Pentagon Produced with the POLY and POLYCONT Functions



Missing values for the X and Y variables that are specified with POLYCONT are interpreted differently from how they are interpreted with the other functions. Other functions use the missing values to request a default value. POLYCONT interprets a missing value as a discontinuity (that is, a hole) in the polygon. If you are not using the data coordinate system and you specify an X or Y value of –999 in a POLYCONT observation, the default of (XLAST, YLAST) is used. Missing values indicate holes and are handled identically in the Annotate facility and the GMAP procedure. See "Displaying Map Areas and Response Data" in *SAS/GRAPH and Base SAS: Mapping Reference* for more information about handling missing values.

POP Function

Removes the (XLAST, YLAST) and (XLSTT, YLSTT) values from the LIFO stack and updates the internal coordinate pairs with the retrieved values.

Updates: (XLAST, YLAST) and (XLSTT, YLSTT)

Syntax

FUNCTION='POP';

Comparisons

Use POP when you want to access the values of (XLAST, YLAST) and (XLSTT, YLSTT) that you most recently stored with the PUSH function. See the PUSH function for a description of the LIFO stack.

PUSH Function

Adds current (XLAST, YLAST) and (XLSTT, YLSTT) values to the LIFO stack.

Syntax

FUNCTION='PUSH';

Comparisons

The LIFO (last-in-first-out) stack is a storage area where you can keep internal coordinate values for later use by utility functions without recalculating those values. LIFO stacks manage the stored data so that the last data stored in the stack is the first data removed from the stack.

Use the stack 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 these values from the stack with 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.

SWAP Function

Exchanges values of (XLAST, YLAST) with (XLSTT, YLSTT) and vice versa.

Updates: (XLAST, YLAST) and (XLSTT, YLSTT)

Syntax

FUNCTION='SWAP';

Comparisons

Use SWAP when you want to use both the (XLAST, YLAST) and (XLSTT, YLSTT) coordinates for text and nontext functions, respectively.

SYMBOL Function

Places symbols in the graphics output. Associated variables can specify the color, font, and height of the symbols displayed.

Updates: XLSTT, YLSTT

Syntax

FUNCTION='SYMBOL';

Associated Variables

CBORDER='*color*' | 'CTEXT'
draws a border of a specified color around the text.

color
specifies the border color as any valid SAS/GRAPH color name. See "CBORDER Variable" on page 805.

CTEXT
specifies the text color as the border color.

CBOX='*color*' | 'CBACK'
draws a solid box of a specified color behind the text.

color
specifies the box color as any valid SAS/GRAPH color name. See "CBORDER Variable" on page 805.

CBACK
specifies the background color as the box color.

COLOR='*color*'
specifies the symbol color. *Color* can be any SAS/GRAPH color name. See "COLOR Variable" on page 807. The COLOR variable behaves in the same way as the COLOR= option in the SYMBOL statement.

See "SYMBOL Statement" on page 470

GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See "GROUP Variable" on page 811

"MIDPOINT Variable" on page 822

"SUBGROUP Variable" on page 835

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable.

See "HSYS Variable" on page 813

HTML='*link-string*'
specifies the text that defines the drill-down link.

See "HTML Variable" on page 814

SIZE=*height*
specifies the height of the symbol that is being drawn, using units determined by the HSYS variable. The SIZE variable is equivalent to the HEIGHT= option in the SYMBOL statement.

See "SIZE Variable" on page 829

"SYMBOL Statement" on page 470

STYLE='*font-specification*' | 'NONE'
specifies the font that is used to draw the symbol that is specified by the TEXT variable. When the STYLE variable is used with the SYMBOL function, it behaves the same as the FONT= option in the SYMBOL statement. By default, no font is specified and the symbol that is specified by the TEXT variable is taken from the special symbol table. If you use STYLE to specify a symbol font, such as Marker, the string that is assigned by the TEXT variable is the character code for a symbol. If you use STYLE to specify a text font, such as Swiss, the string assigned by the TEXT variable is displayed as text.

See "STYLE Variable: Arrows" on page 832

"STYLE Variable: Fonts" on page 831

TEXT='*special-symbol*' | '*text-string*';
specifies the symbol or text string to be displayed. *Special-symbol* can be up to eight characters long. Values for *special-symbol* are those described in the VALUE= option of the SYMBOL statement.

For ActiveX, the following values are supported: plus, X, star, square, diamond, triangle, dot, circle, ", #, $, %, =. If a symbol is not supported, a plus sign (+) is drawn instead.

For Java, the following values are supported: plus, X, star, square, diamond, triangle, dot (draws a circle), circle, *, +, >. If a symbol is not supported, a plus sign (+) is drawn instead.

If you also specify a text font with the STYLE variable, you can specify a text string that is displayed as the symbol. The maximum length for *text-string* is 200 characters.

When the TEXT variable is used with the SYMBOL function, it behaves the same as the VALUE= option in the SYMBOL statement.

See "TEXT Variable" on page 837

"SYMBOL Statement" on page 470

WHEN='B' | 'A'
specifies when to draw the symbols in relation to other procedure output.

B
specifies that the symbols are drawn before the other procedure output is drawn.

A
specifies that the symbols are drawn after the other procedure output is drawn.

See "WHEN Variable" on page 838

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate*
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the point at which the symbol is placed.

Restriction The Z= variable is used with the G3D procedure only.

See "X Variable" on page 840

"Y Variable" on page 846

"Z Variable" on page 850

"XC Variable" on page 841

"YC Variable" on page 847

XSYS='*coordinate-system*'
specifies the coordinate system for the X or XC variable. Use the XC variable only with XSYS='2'.

See "XSYS Variable" on page 843

YSYS='*coordinate-system*'
specifies the coordinate system for the Y or YC variable. Use the YC variable only with YSYS='2'.

See "YSYS Variable" on page 849

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable

Restriction The ZSYS= variable is used with the G3D procedure only.

See "ZSYS Variable" on page 851

Details

SYMBOL is similar to the LABEL function with these exceptions:

- SYMBOL draws symbols. If you do not specify a font, SYMBOL can use the symbols found in Table 25.65 on page 492.

- The text cannot be rotated or angled.
- The text string cannot be longer than eight characters.
- The text string is always centered with respect to *x* and *y*.

TXT2CNTL Function

Copies the values (XLSTT, YLSTT) to (XLAST, YLAST), replacing previous values of (XLAST, YLAST).

Syntax

FUNCTION='TXT2CNTL';

Comparisons

TXT2CNTL allows nontext functions to use the ending position of a text string as a starting or ending point.

30

Annotate Variable Dictionary

About the Annotate Variables

When an Annotate data set is processed, the Annotate facility looks at the values of specific variables in order to draw graphics. This section describes all of the Annotate variables in alphabetical order. Not all variables are used with all functions. Refer to the description of the individual functions in "About the Annotate Functions" on page 759 for more information about how each variable is used with each function. For a summary of Annotate variables and their uses, see Table 28.1 on page 727.

Annotate Internal Coordinates

The Annotate facility maintains two sets of internal coordinates that are stored in the variable pairs (XLAST, YLAST) and (XLSTT, YLSTT). One set of variables (XLAST, YLAST) stores coordinate values that are generated by nontext functions and the other set (XLSTT, YLSTT) stores coordinates generated by text functions. These two variable pairs supply default values when the X or Y variable contains a missing value.

Both pairs are initially set to 0 and remain 0 until a function updates the values. You cannot assign explicit values to these variables, but you can manipulate their values with some of the Annotate functions.

Dictionary

ANGLE Variable

Specifies the angle at which the graphics output is drawn.

Type: numeric

Default: function dependent

Syntax

ANGLE=*angle-value*;

Variable Values

angle-value
specifies the angle at which the graphics output is to be drawn.

Details

The ANGLE variable is function dependent.

Function	What the ANGLE variable specifies
ARROW	the angle of the tip of the arrowhead. You can specify any number value. If the angle that you specify is not between 0 and 180, the absolute value of mod(*angle-value*,180) is used. For example, the values -45, 45, and 225 all produce the same result. The default value is 30.
LABEL	the baseline angle of the character string with respect to the horizontal. With the LABEL function, the pivot point is at (X,Y) and the direction of rotation is counterclockwise. The valid values are from 0 to 360. The default value is 0.
PIE	the starting angle of the slice arc, measured counterclockwise. The valid values are from –360 to 360. The default for the first PIE function is ANGLE=0 (horizontal, or 3:00 position), or is the ending point of the arc of the previous slice. Specify a value for the ANGLE variable if you want the next slice to start at an angle that is different from the edge of the previous slice. Also, specify a value if you want the first slice to start at an angle other than horizontal.
PIEXY	the angle that works with the SIZE variable to establish the new XLAST, YLAST point relative to the last pie element established with the PIE or PIECNTR functions. The angle is measured counterclockwise starting at the 3:00 position. The default value is 0.

CBORDER Variable

Draws a colored border around text or symbols.

Type: character

Length: 8 for color codes and up to 64 for color names

See: CBOX

Syntax

CBORDER='*color*' | 'CTEXT';

Variable Values

color
specifies the color that outlines the box. The *color* value can be any SAS/GRAPH color name. See Chapter 23, “Using Colors in SAS/GRAPH Programs,” on page 363 for more information about specifying colors.

Specifying a null value for the *color* value (CBORDER=' ') cancels the CBORDER variable.

CTEXT
draws the border in the same color as the text or symbol. The text color is determined by (1) the COLOR variable or (2) the CTEXT=graphics option or (3) the first color in the color list.

Details

Using the CBORDER Variable

Once you have specified CBORDER, it remains in effect for all subsequent observations that use the LABEL or SYMBOL function and draws a border around all text or symbols. To turn off the border for subsequent text or symbols, specify CBORDER=' '.

To fill the area defined by CBORDER, use the CBOX variable in conjunction with CBORDER.

Functions

You can use the CBORDER variable with these functions:

LABEL
SYMBOL

CBOX Variable

Draws a solid box behind the text or symbol and fills the box with the specified color.

Type:	character
Length:	8 for color codes and up to 64 for color names
Note:	The size of the box that is drawn behind some symbols might change when UTF-8 encoding is used in the SAS session.

See: CBORDER

Syntax

CBOX='*color*' | 'CBACK';

Variable Values

color
specifies the color that fills the box. *Color* is any SAS/GRAPH color name. See Chapter 23, “Using Colors in SAS/GRAPH Programs,” on page 363 for more information about specifying colors.

Specifying a null value for *color* (CBOX=' ') cancels the CBOX variable.

CBACK
fills the box with the same color as the background color of the graph. The background color is either (1) the color specified by the CBACK= graphics option or (2) the default background color for the device.

Details

Using the CBOX Variable

Once you have specified CBOX, it remains in effect for all subsequent observations that use the LABEL or SYMBOL function.

The color of the text or symbol within the box is controlled by the COLOR variable.

By default, the solid box has no border. To add a colored border to the box, use the CBORDER variable in conjunction with CBOX.

Functions

You can use the CBOX variable with these functions:

LABEL
SYMBOL

COLOR Variable

Specifies the color used by the function.

Type: character

Length: 8 for color codes and up to 64 for color names

Default: 1. COLORS= graphics option, if specified; 2. first color in device's default color list

Syntax

COLOR='*color*';

Variable Values

color
specifies any SAS/GRAPH color name. See Chapter 23, “Using Colors in SAS/GRAPH Programs,” on page 363 for more information about specifying colors.

Details

The COLOR variable is function dependent.

Function	What the COLOR variable specifies
BAR	The color that outlines and, as an option, fill the bar if a pattern is specified in the STYLE (patterns) on page 833 variable. If no pattern is specified, the *color* value is applied only to the outline of the bar.
ARROW	The color of the arrow.
DRAW, DRAW2TXT	The color of the line.
FRAME	The color of the outline of the frame. If a fill pattern is specified, *color* also determines the color of the inside of the frame.
LABEL	The color of the text.
PIE	The color for the pie slice if a pattern is specified with the STYLE (patterns) on page 833 variable. If no pattern is specified, *color* determines the color of the outline of the pie slice.
POINT	The color of the point.
POLY	The fill color for the interior of the polygon if a pattern is specified with the STYLE variable. If the STYLE variable is missing or EMPTY, *color* is ignored. Use the POLYCONT function to specify the outline color.

Function	What the COLOR variable specifies
POLYCONT	The color that outlines the polygon when used with the first POLYCONT function. COLOR is ignored for subsequent POLYCONT functions in the POLYCONT sequence.
SYMBOL	The color that draws the symbol.

FUNCTION Variable

Specifies a graphics command or programming function for the Annotate facility to perform.

Type: character

Length: 8

Default: LABEL

Syntax

FUNCTION='*function-name*';

Variable Values

function-name

specifies the name of an Annotate function. The *function-name* value can be any of the following.

BAR

draws and fills a rectangle (optional).

CNTL2TXT
DRAW2TXT

copies (XLAST, YLAST) to (XLSTT, YLSTT), overwriting the previous values of (XLSTT, YLSTT).

COMMENT

places comments in your data set. The observation is ignored when the data set is processed.

DEBUG

writes the values of all Annotate variables to the SAS log before and after the next observation.

DRAW

draws a line in the graphics output.

FRAME

draws a border around the area defined by XSYS and YSYS and specifies a background color for the framed area.

IMAGE
displays an image in the graphics output from the current (X,Y) coordinates to the coordinates that are associated with the IMGPATH variable.

LABEL
draws text and is the default for the FUNCTION variable.

MOVE
moves to the specified point (does not draw a line).

PIE
draws a pie slice, arc, or circle that can be filled.

PIECNTR
sets new center and radius values. The PIEXY function can use this information in a later observation.

PIEXY
returns the coordinates of a point on a pie slice. Other functions can use this information in a later observation.

POINT
draws a point.

POLY
begins drawing a polygon (first vertex). Use the POLYCONT function in successive observations to supply the remaining vertices.

POLYCONT
continues drawing a polygon.

POP
gets values from the LIFO stack and changes the current value of (XLAST, YLAST) and (XLSTT, YLSTT) to those values.

PUSH
puts the current values for (XLAST, YLAST) and (XLSTT, YLSTT) in the LIFO stack.

SWAP
exchanges the values of (XLAST, YLAST) and (XLSTT, YLSTT).

SYMBOL
draws a symbol.

TXT2CNTL
copies the values (XLSTT, YLSTT) to (XLAST, YLAST), overwriting the previous values of (XLAST, YLAST).

For a list of the symbols that can be used with the SYMBOL variable, see Table 25.65 on page 492.

All other variables in the observation that contain the function act as parameters for the action. For a detailed description of each function and the Annotate variables that can be used in conjunction with it, see “About the Annotate Functions” on page 759.

GROUP Variable

Positions graphics elements on the bars of a vertical or horizontal bar chart drawn using the GROUP= option in the GCHART procedure.

Type:	Numeric or character; must match the type of the GROUP= variable used in the GCHART procedure.
Length:	Should match the length of GROUP= variable in the GCHART procedure.
Default:	none
Restriction:	Used only with vertical or horizontal bar charts produced by the GCHART procedure.

Syntax

GROUP=*group-value*;

Variable Values

group-value
references value(s) of the variable that is identified by the GROUP= option in the GCHART procedure either as a variable name or as an explicit data value. *Group-value* can be one of the following:

group-variable
the name of a group variable.

group-data-value
a specific numeric data value.

'***group-data-value***'
a specific character data value.

To annotate all the bars in a horizontal or vertical bar chart, specify a variable name. To annotate a bar chart for a specific value of the GROUP variable, specify a specific value.

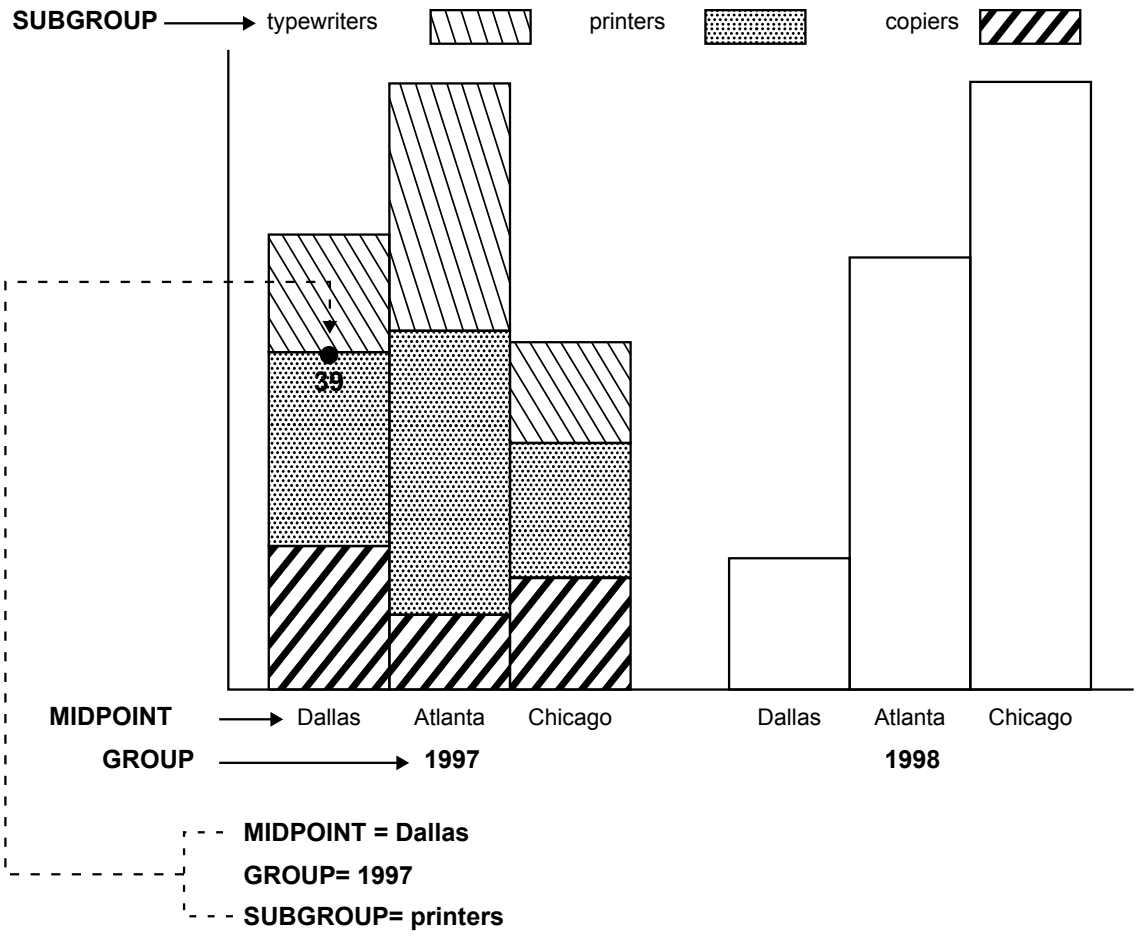
Details

Using the GROUP Variable

Using the GROUP variable is similar to using the X and Y variables with data system coordinates to position graphics elements in a vertical or horizontal bar chart.

Figure 30.87 on page 812 shows how the GROUP variable works with the SUBGROUP and MIDPOINT variables to label the bars of a vertical bar chart.

Figure 30.1 Using the GROUP Variable to Position a Label in a Bar Chart



The label showing the number of units that were sold in Dallas in the year 1997 is positioned by the values that are assigned to these Annotate variables:

- GROUP=YEAR (where YEAR is a variable in the GCHART data set)
- MIDPOINT=CITY (where CITY is a variable in the GCHART data set)
- SUBGROUP=ITEM (where ITEM is a variable in the GCHART data set).

Functions

You can use the GROUP variable only with the data coordinate systems 1, 2, 7, and 8, and with these functions:

BAR	MOVE	POINT
DRAW	PIE	SYMBOL
LABEL	PIECNTR	

HSYS Variable

Defines the coordinate system and area of the output used by the SIZE variable to display the Annotate graphics. In addition, you can use the HSYS variable with Java or ActiveX to control the marker size and line size for the BAR, DRAW, DRAW2TXT, POLY, and SYMBOL functions.

Type: character

Length: 1

Default: 4

Syntax

HSYS='*coordinate-system*';

Variable Values

coordinate-system

specifies a value that represents a coordinate system. Values can be 1 through 9 and A through C as shown in the following table:

Absolute Systems	Relative Systems	Coordinate System Units
1	7	percentage of data area
2	8	data values
3	9	percentage of graphics output area
4	A	cell in graphics output area
5	B	percentage of procedure output area
6	C	cell in procedure output area
D		point size (text only)

These values are also used by the XSYS and YSYS variables. See “Coordinate Systems” on page 734 for a description of the areas and coordinate systems.

Details

About the Coordinate System

The coordinate system that you specify with the HSYS variable affects how the function interprets the value of the SIZE variable. For example, if you use HSYS='3' and SIZE=10 with the DRAW function, the thickness of the line is 10% of the graphics output area. If you use HSYS='1' and SIZE=10 with DRAW, the thickness of the line is 10% of the data area.

For text only, HSYS='D' specifies that text sizes are in points. For example, if you use HSYS='D' and SIZE=10 for the LABEL function, the label text uses a 10-point font.

If you use HSYS='D' with a function that does not create text, a warning appears in the log and the HSYS='4' coordinate system is used.

Functions

You can use HSYS with these functions, all of which also use the SIZE variable:

DRAW	LABEL	SYMBOL
DRAW2TXT	PIE	
FRAME	PIECNTR	

HTML Variable

Defines a link in the HTML file that is created for a drill-down graph. This link is associated with an area of the graph and contains valid HTML syntax that can link to a report or another graph. The report or graph is displayed when the user clicks on the area.

Type:	character
Length:	no limit
Default:	none
Tip:	When used with an SVG graphics device, the variable can create particular drill-down events. When used with the LABEL function and an SVG graphics device, this variable can be used to animate the label text.

Syntax

HTML='*link-string*';

HTML="ANIMATE=*animate-action*(*parameters*)";

HTML=*enhanced-link-options*;

Variable Values

link-string

specifies the text that defines the drill-down link. For more information about drill-down graphs and how to specify the link string, see “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 255. For using the HTML variable for data tips, see “Adding Custom Data Tips with the HTML= Option” on page 253.

Restriction The ACTXIMG device does not support nested quotation marks in drill-down link URLs for annotations. If you need to nest quotation marks in your annotation drill-down link URLs, use the PNG device instead.

"ANIMATE=*animate-action*(*parameters*)"

specifies an animation action and animation control parameters that are required to animate a label. Use this option with the LABEL function and an SVG graphics device to animate label text. *Animate-action* specifies an animation action while *parameters* specifies the parameters that control the animation.

Note: "ANIMATE=*animate-action*(*parameters*)" must be enclosed in double quotation marks. If you use single quotation marks, an error is generated.

When you animate the label text, the output is saved as an SVG graphic.

animate-action can be one of the following:

BLINK('*text-string*', *begin*, *speed*, REPEAT | *repeat-value*)

causes the text in the label observation to blink at a specified rate.

Provide all of the following parameters in the order listed here.

'*text-string*'

specifies the value of the label to blink.

This parameter must exactly match the text variable in the corresponding label observation in the annotate data set. In addition, '*text-string*' must be enclosed in single quotation marks. If you use double quotation marks, an error is generated.

begin

specifies after how many seconds to begin the blinking action after the SVG page has been loaded.

speed

specifies the blinking speed in seconds.

REPEAT | *repeat-value*

repeats the blink action.

REPEAT

repeats indefinitely.

repeat-value

repeats *repeat-value* times.

Examples This example causes the text label ‘blinking text’ to blink when the page is loaded into the browser. The text blinks at a speed of one second, and repeats indefinitely.

```
HTML="animate=blink('blinking text', 0, 1, repeat)"
```

This example causes the text label ‘text blinking for 10 times’ to blink when the page is loaded into the browser. The text blinks 10 times at a speed of 2 seconds, and then stops blinking.

```
HTML="animate=blink('text blinking for 10 times',
   0, 2, 10)"
```

SIZE('*text-string*', *begin*, *duration*, REPEAT | *repeat-value*)
causes the text in the label observation to increase or decrease in size by modifying the font size of the text over a time interval.

Provide all of the following parameters in the order listed here.

'*text-string*'
specifies the value of the label to resize.

This parameter must exactly match the text variable in the label observation in the annotate data set. In addition, '*text-string*' must be enclosed in single quotation marks. If you use double quotation marks, an error is generated.

font-size
specifies the new font size in pixels. The text changes to this size during the specified duration.

begin
specifies after how many seconds to begin the sizing action after the SVG page has been loaded.

duration
specifies the duration in seconds during which the text sizing should occur.

REPEAT | *repeat-value*
repeats the size action.

REPEAT
repeats indefinitely.

repeat-value
repeats *repeat-value* times.

Examples This example causes the text label ‘growing text’ to increase to an 80-pixel font size. The increase occurs three seconds after the page is loaded into the browser over a duration of five seconds. The action repeats indefinitely.

```
HTML="animate=size('growing text', 80, 3, 5, repeat)"
```

This example causes the text label ‘growing text’ to decrease to 0 pixel font size. The increase occurs five seconds after the page is loaded into the browser over a duration of three seconds, and then the animation stops.

```
HTML="animate=size('shrinking text', 0, 5, 3, norepeat)"
```

MOVE('*text-string*', *toX*, *toY*, *begin*, *duration*, REPEAT | *repeat-value*)
moves the text in the label observation to a specified coordinate in the SVG output within a specified time interval.

Provide all of the following parameters in the order listed here.

'*text-string*'
specifies the value of the label to move.

This parameter must exactly match the text variable in the label observation in the annotate data set. In addition, '*text-string*' must be enclosed in single quotation marks. If you use double quotation marks, an error is generated.

toX
specifies the X coordinate that the text moves to in percentage units of the display or the SVG space.

You can specify either a relative or absolute value.

Note: The *toX* and *toY* coordinates can be specified in either a relative or absolute value from the original X and Y coordinates of the annotated text string. To indicate a relative move, add + or – prior to the coordinate. The symbol causes the text to move the specified amount in a positive or negative direction relative from the original text coordinate. To move the text to an absolute position, specify the percentage-based coordinates without the + or –.

toY
specifies the Y coordinate that the text moves to in percentage units of the display or the SVG space.

You can specify either a relative or absolute value. (See the note for toX.)

begin
specifies after how many seconds to begin the move action after the SVG page has been loaded.

duration
specifies the duration in seconds during which the text move should occur.

REPEAT | *repeat-value*
repeats the move action.

REPEAT
repeats indefinitely.

repeat-value
repeats *repeat-value* times.

Examples This example moves the text label ‘moving text’ to the bottom right corner of the screen when the page is loaded into the browser. The text moves over a duration of four seconds, and repeats indefinitely.

```
HTML="animate=move('moving text', 100, 0, 0, 4, repeat)"
```

This example moves the text label across the bottom of the screen when the page is loaded into the browser. The text moves over a duration of five seconds, and repeats indefinitely. The X value of *100* is interpreted as an absolute value and moves the text 100% across the display. The Y value of *+0* is relative to the original Y coordinate. The 0 value has the effect of not moving the text vertically.

```
HTML="animate=
   move('ALERT: SVG can create a moving news ticker!',
      100, +0, 0, 5, repeat)"
```

TEXTPATH('*text-string*', *from-offset*, *to-offset*, *begin*, *duration*, REPEAT | *repeat-value*)

causes the text in the label observation to move along a path at a specified speed. The path that the text follows must also be defined in the annotate data set.

Provide all of the following parameters in the order listed here.

'*text-string*'

specifies the value of the label to move.

This parameter must exactly match the text variable in the label observation in the annotate data set. In addition, '*text-string*' must be enclosed in single quotation marks. If you use double quotation marks, an error is generated.

from-offset

specifies the starting position of the text string relative to the path. This value is expressed as a percentage of the path.

to-offset

specifies the ending position of the text string relative to the path. This value is expressed as a percentage of the path.

begin

specifies after how many seconds to begin the move action after the SVG page has been loaded.

duration

specifies the duration in seconds during which the move should occur.

REPEAT | *repeat-value*

repeats the move action.

REPEAT

repeats indefinitely.

repeat-value

repeats *repeat-value* times.

Restriction There can be only one TEXTPATH action per SVG file.

Requirement The line that the text follows must be added to the annotate data set using the MOVE and DRAW functions. The line annotation must immediately follow the label observation in the data set.

Example This example causes the text label 'moving text on a path' to move from the 0% to 100% offset of the path. The animation occurs when the page is loaded into the browser. The text moves across the annotated line over a duration of five seconds, and repeats indefinitely.

```
HTML="animate=textpath('moving text on a path', 0,
   100, 0, 5, repeat)"
```

Example "Example: Moving Text on a Path" on page 781

Restriction Available only when used with the LABEL function and an SVG graphics device.

See Chapter 13, "Using SVG Graphics," on page 159

enhanced-link-options

When you generate output as an SVG graphic, you have additional options for enhancing your drill-down links. The SVG graphics devices provide the following *enhanced-link-options*.

ONMOUSEOVER=*events*

specifies particular events that occur when the mouse pointer is positioned over a linked shape.

ONCLICK=*events*

specifies particular events that occur when a linked shape is clicked.

For the complete syntax with examples, see "Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" on page 262.

See Chapter 13, "Using SVG Graphics," on page 159

Details

Setting the Length of the HTML Variable

Use a LENGTH statement to set the length of the HTML variable to the longest string that you need for the link string. Be sure to set the HTML value to a null if you continue writing observations to the annotate data set after you are done assigning links. For example, the following code defines link information for two squares, but then sets the HTML variable to null when drawing a frame. Otherwise, the background area within the frame uses the link information from the last defined HTML value and become a hot zone in the graph.

```
data squares;
     length function style color $ 8
            html text $ 15;
     xsys="3"; ysys="3";
        /* draw a green square */
     color="green";
     function="move"; x=10; y=65; output;
     function="bar";  x=30; y=95; style="solid";
        html="href=green.gif"; output;
        /* draw a red square */
     color="red";
     function="'move"; x=60; y=65; output;
     function="bar";  x=80; y=95;
        html="href=red.gif"; output;
        /* draw a blue frame */
     function="frame"; color="blue"; style="empty";
        /* set null link for background area in frame */
        html=""; output;
run;
```

Functions

You can use the HTML variable with these functions:

- ARROW

- BAR
- FRAME
- IMAGE
- LABEL
- PIE
- POLY
- SYMBOL

IMGPATH Variable

Specifies an image to be displayed from the current (X,Y) coordinates to the (X,Y) coordinates that are associated with this variable.

Type: character

Length: 255

Syntax

IMGPATH='*external-file*';

Variable Values

external-file
specifies the full path or full filename of an external image file. The format of the external file specification varies between operating environments. You can also specify the URL for a graphic on a web page. For example:

```
'http://mywebsite.com/myImage.png'
```

Note Copying and pasting the image works only if an absolute path is specified instead of a relative path, or if the file into which the image is being pasted is opened from the directory to which the image is relative.

Details

The IMGPATH variable can be used only with the "IMAGE Function" on page 773.

The manner in which the specified image is to be displayed is determined by the "STYLE Variable: Images" on page 832.

For a list of the file types that you use, see "Image File Types Supported by SAS/GRAPH" on page 384.

LINE Variable

Controls the drawing of a line by determining either the type of line to draw or the relative position of the line.

Type: numeric

Default: 1

Syntax

LINE=*line-type*;

Variable Values

line-type
specifies the line type. The line type that can be used depends on the function. See "Details" on page 821.

Details

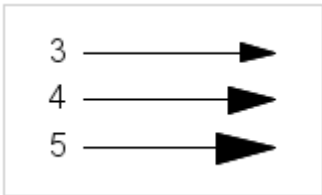
Syntax and Behavior of the LINE Variable

The syntax and behavior of the LINE variable are function dependent.

LINE Variable in the ARROW Function

In the ARROW function, the valid values are positive numbers greater than 1. The value of the LINE variable specifies the length of the sides of the arrowhead. The units for the LINE variable are always a percentage of the graphics area, regardless of the HSYS= value.

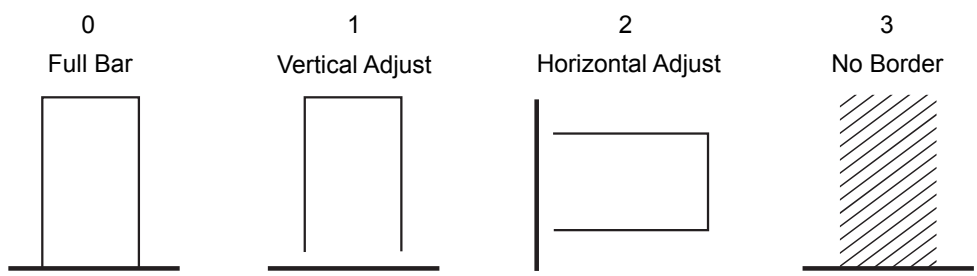
Figure 30.2 *Sample LINE Values for Filled Arrowheads*



LINE Variable in the BAR Function

In the BAR function, valid values for the LINE variable can be 0, 1, 2, or 3. These values determine how the outline of the bar is to be drawn, as shown in the following figure. A value of 0 draws the outline all the way around the bar. A value of 1 draws the outline only on the vertical sides of the bar. A value of 2 draws the outline only on the horizontal sides of the bar. A value of 3 draws no outline.

Figure 30.3 *LINE Values for Bars*



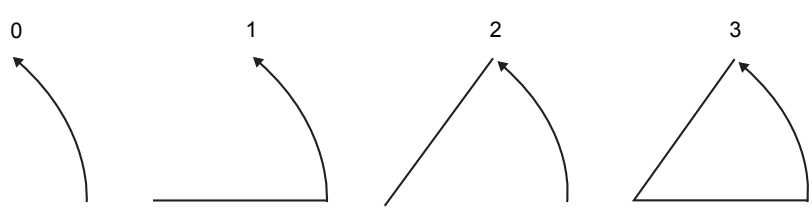
LINE Variable in the DRAW, DRAW2TXT, FRAME, and POLY Functions

Valid values are whole numbers from 1 to 46. A value of 1 specifies a solid line. The remaining values specify different segmented lines, as illustrated in Figure 25.52 on page 503.

LINE Variable in the PIE Function

Valid values are 0, 1, 2, or 3. The value specifies which lines of a pie slice are to be drawn for the current arc, as shown in Figure 30.90 on page 822.

Figure 30.4 *LINE Values Used with the PIE Function*



MIDPOINT Variable

Positions graphics elements on the bars of a vertical or horizontal bar chart drawn by the GCHART procedure.

Type: Numeric or character; must match the type of the midpoint variable in the GCHART procedure.

Length: Should match the length of the midpoint variable in the GCHART procedure.

Default: none

Restriction: Used only with vertical or horizontal bar charts produced by the GCHART procedure.

Syntax

MIDPOINT=*midpoint-value*;

Variable Values

midpoint-value
references midpoint data value(s) in the GCHART procedure either as a variable name or as an explicit data value. *Midpoint-value* can have one of the following forms:

midpoint-variable
the name of a midpoint variable.

midpoint-data-value
a specific numeric data value.

'***midpoint-data-value***'
a specific character data value.

Generally, specify a variable name if you want to annotate all of the bars in a horizontal or vertical bar chart. To annotate a bar chart for a specific value of the MIDPOINT variable, specify a specific value.

Details

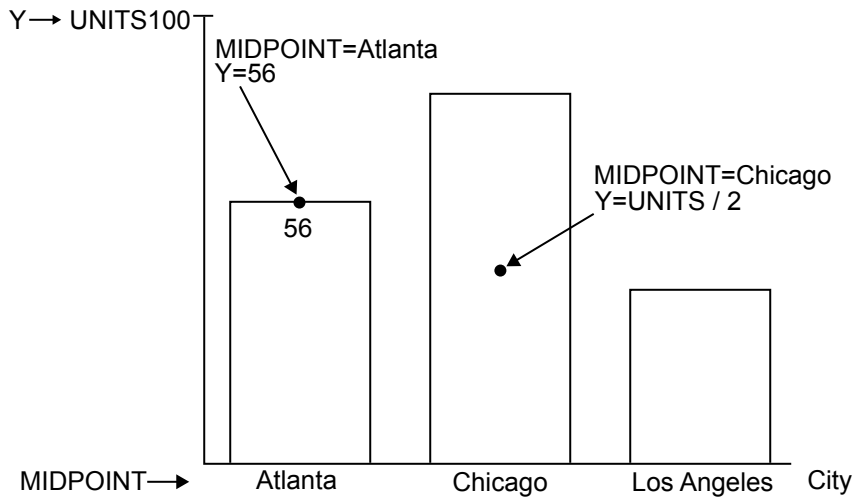
Using the MIDPOINT Variable

Using the MIDPOINT variable is similar to using the X and Y variables to position graphics elements in a vertical or horizontal bar chart when using data system coordinates. For example, suppose you produce a vertical bar chart in which the chart variable CITY produces a bar for each city in a data set. The height of each bar is determined by the value of the SUMVAR= variable, UNITS.

You can label these bars by assigning the chart variable CITY to the Annotate MIDPOINT variable. The MIDPOINT variable provides the *x* coordinate for the label. By default, Annotate assigns the statistic variable, in this case the SUMVAR= variable, UNITS, to the Annotate Y variable, which provides the *y* coordinate for the label.

Figure 30.91 on page 824 shows how the values of the MIDPOINT and Y variables position the label that shows the number of units sold in Atlanta. The value, which is calculated and printed by the LABEL function, is 56.

Figure 30.5 *Using the MIDPOINT Variable to Position a Label in a Bar Chart*



The labels in this figure are positioned by the values that are assigned to these Annotate variables:

- MIDPOINT=CITY (where CITY is the chart variable); the MIDPOINT variable provides the horizontal coordinate in the vertical bar chart.
- Y=UNITS (where UNITS is the SUMVAR= variable); the Y variable provides the vertical coordinate. By specifying Y=units/2, you can vertically center the label in the bar.

Note: In a horizontal bar chart, the MIDPOINT variable controls the *y* coordinate and the statistic variable controls the *x* coordinate.

CAUTION

Be careful when using MIDPOINT and X and Y variables in the same data set. Using the MIDPOINT and X variables in an Annotate data set that is used to annotate a VBAR chart can cause unexpected results. The same is true when using the MIDPOINT and Y variables in the same data set that is used to annotate an HBAR chart.

When annotating a VBAR chart, the Annotate facility uses the MIDPOINT variable as the horizontal coordinate if it exists in the Annotate data set and ignores the X variable. Consequently, you should use the MIDPOINT variable as the horizontal coordinate for all observations in an Annotate data set if you use it for one. A similar behavior occurs if you use both the MIDPOINT and Y variables in an Annotate data set that is used to annotate HBAR charts. The MIDPOINT variable is always used, regardless of whether it has a missing value, and the Annotate facility ignores the Y variable. In this case, as well, use the MIDPOINT variable for the vertical coordinate for all observations in an Annotate data set if you use it for one.

Functions

You can use the MIDPOINT variable only with the data coordinate systems 1, 2, 7, and 8, and with these functions:

BAR MOVE POINT

DRAW	PIE	SYMBOL
LABEL	PIECNTR	

POSITION Variable

Controls placement and alignment of a text string specified by the LABEL function.

Type: character

Length: 1

Default: 5

Syntax

POSITION=*text-position* | '0";

Variable Values

text-position

specifies the placement of the text string in relation to the position that is defined by the X and Y variables. *Text-position* can be one of the characters 1 through 9, A through F, <, +, or >. These characters represent the positions that are described in the following table.

Vertical Position of Text Relative To Specified Location	Horizontal Alignment		
	Right	Center	Left
One cell above location	1	2	3
Centered on location	<	+	>
Text baseline half cell below location	4	5	6
One cell below location	7	8	9
Half cell above location	A	B	C
Half cell below location	D	E	F

These positions are illustrated in Table 30.81 on page 826.

'0'

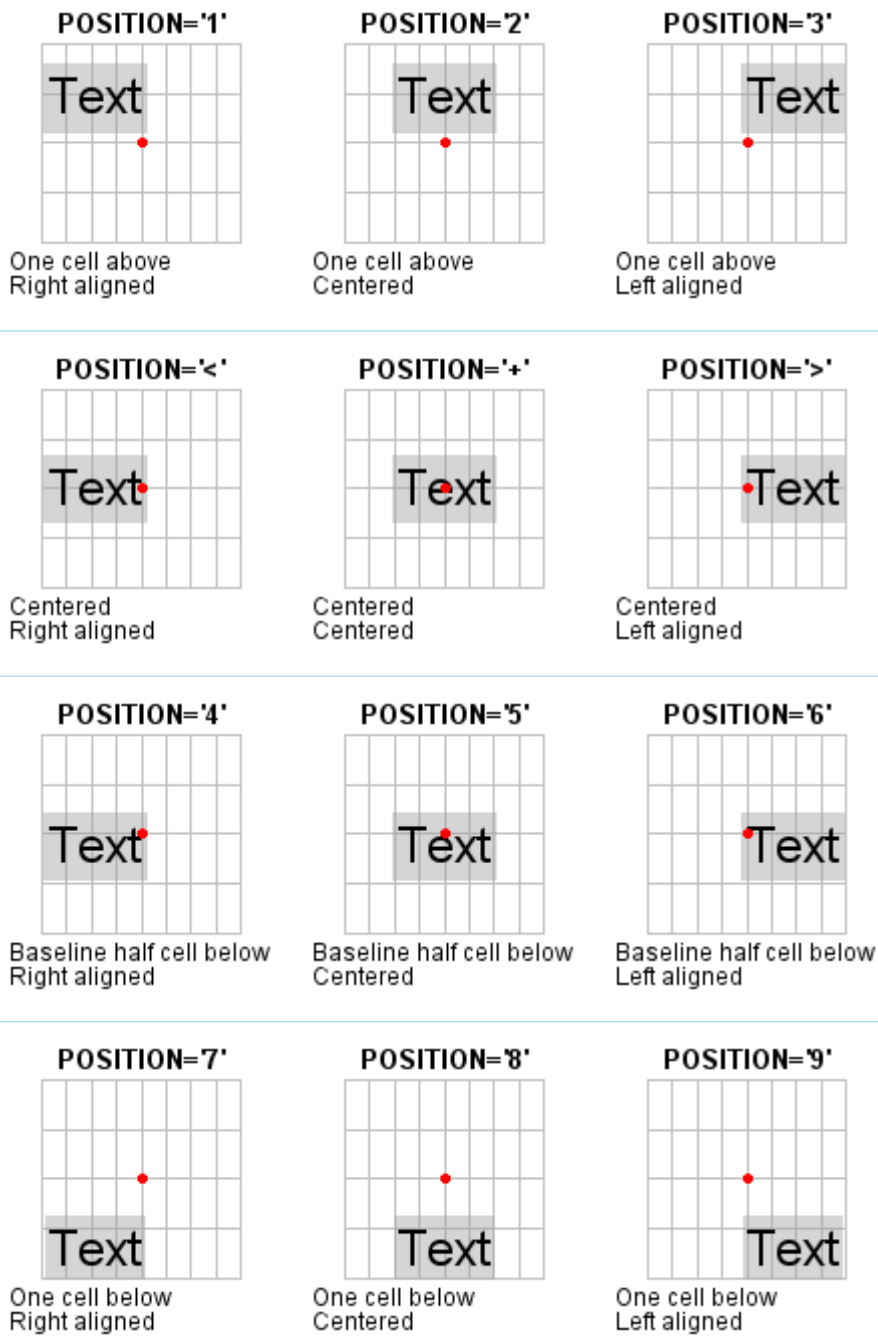
specifies a pause in the string in order to change an attribute, such as the color of the text.

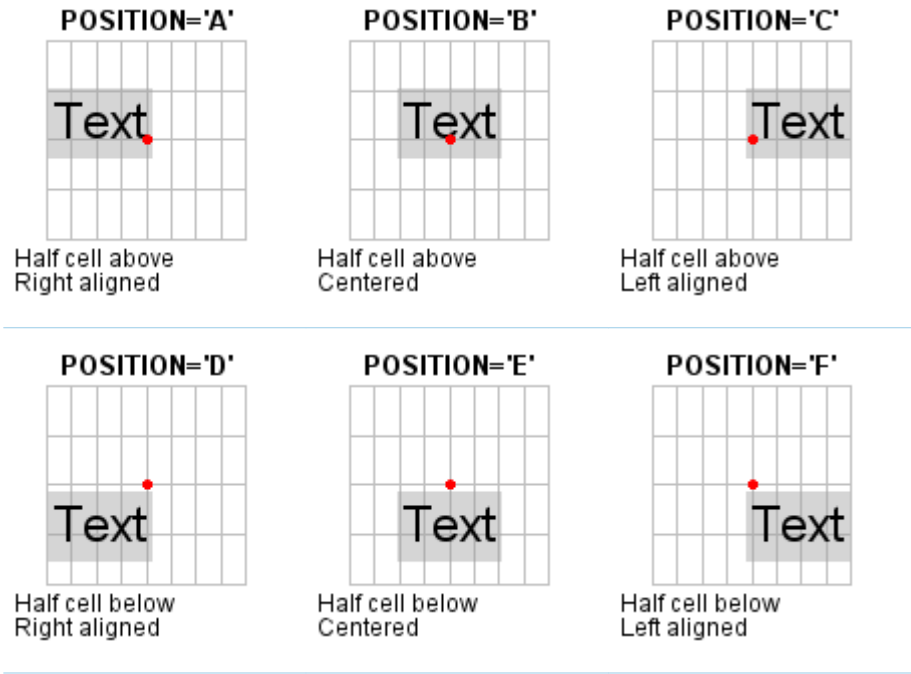
Details

Positioning Text Strings

Table 30.81 on page 826 shows the effect of the POSITION= variable values on text string, "Text." In each image, The red dot represents the location specified for the text string. The gray box surrounding the text string indicates the bounding box for the text.

Table 30.1 *Effect of POSITION Values on Text Strings*



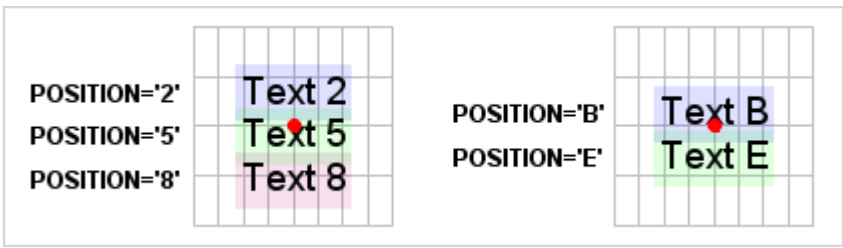


The <, +, and > positions are recommended for labels that are numbers. The + position is also recommended for annotating font-based markers. The <, +, and > positions are especially useful when you are labeling a horizontal bar chart. You can use <, +, or > if the numbers in your font are significantly smaller than the text and you are having trouble centering labels. If the numbers in your font are the same or nearly the same height as the text, you can use positions 4, 5, and 6 to center the labels.

Stacking Text Strings

To stack text strings, specify a different position value of each string. Figure 30.92 on page 827 shows two ways to stack text. The text bounding boxes are shaded with different colors so that you can see how they overlap when text strings are stacked.

Figure 30.6 *Combining POSITION Values to Stack Text*



Note: You cannot stack <, +, and > positions as you can 4, 5, and 6 positions.

Changing Attributes in the Middle of a Text String

0 is a special value to use when you want to pause and then continue a text string. With this value, you can change colors, fonts, and so on, in the middle of a line, while retaining the exact position of the text at the pause. When POSITION='0', the combined text string is left-justified beginning at the point that is defined by the X and Y variables. However, you must define missing values for X for the continuation string. The following Annotate data set changes the font in the middle of the string. The result is shown in Figure 30.93 on page 828.

```
data anno;
   length style $ 8 text $ 12;
   xsys="3"; ysys="3"; hsys="3"; x=5; y=50;
      style="swissb"; size=10; text="This is the";
      position="0"; output;
   x=.; style="swissbi"; text=" ITALIC font";
       output;
run;
```

Figure 30.7 *Using POSITION='0' to Change the Attributes of a Text String*

ROTATE Variable

Specifies the angle at which to rotate the graphics element.

Type: numeric

Default: 0.00

Syntax

ROTATE=*rotation-angle*;

Variable Values

rotation-angle
specifies the angle at which the graphics element is to be rotated. This variable is function dependent. See "Details" on page 829.

Details

The ROTATE variable is function dependent:

Function	What the variable does
PIE	Specifies the sweep of the generated arc that begins at the angle that is specified by the ANGLE variable that is used with the PIE function.
LABEL	Rotates the individual text characters with respect to the baseline.

SIZE Variable

Determines the size of the graphics element with which it is used.

Type: numeric

Length: 8

Default: 1.00 (2 when HSYS=3)

Interaction: For the LABEL function, the value of the HTEXT= goption is used as the default. However, the value of the GUNIT= goption affects the default value that is used by the SIZE= variable.

Syntax

SIZE=*size-factor*;

Variable Values

size-factor
specifies the size of the graphics element. This variable is function dependent. See "Functions" on page 830.

Details

About the SIZE Variable

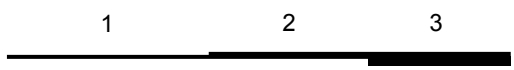
The SIZE variable uses the coordinate system that is specified by the "HSYS Variable" on page 813, which specifies the type of coordinate system used to generate the graph.

For lines, as the thickness of the line increases, it might be impossible to center around a given coordinate. For example, if you specify a thickness of value 2 and HSYS='4', the first line is drawn at the (X, Y) coordinates. The second is drawn slightly above the first. The exact amount varies by device, but it is always one pixel in width. A thickness of value 3 produces one line above, one line at, and one line below the (X, Y) coordinate position.

The SIZE variable is equivalent to the HEIGHT= option in the SYMBOL statement. See "SYMBOL Statement" on page 470 for details.

See Figure 30.94 on page 830 for examples of line thicknesses.

Figure 30.8 *Sample Line Thicknesses Used with the SIZE Variable*



Functions

The SIZE variable is function dependent.

Function.	What the variable does
ARROW	Determines the thickness of the arrow being drawn.
DRAW, DRAW2TXT, or FRAME	Determines the thickness of the line being drawn.
LABEL	Specifies the height of the text.
PIE or PIECNTR	Determines the radius of the pie.
PIEXY	Sets the radius multiplier.
SYMBOL	Selects the height of the symbol.

STYLE Variable: Fonts

Specifies a font for text or symbols produced by the LABEL or SYMBOL functions.

Type:	character
Length:	Depends on specification.
Default:	default device-resident font
Restriction:	Partially supported by ActiveX and not supported by Java

Syntax

STYLE='*font-specification*' | 'NONE';

Variable Values

font-specification
specifies a font. You can specify a GRSEG catalog entry that is supplied by SAS (for example, CENTB) or a system font that is available in your operating environment. A device-resident font can be specified by using either of these forms:

- HW*xxxnn*
- *font-name*

NONE
specifies the default device-resident font.

Details

When the STYLE variable is used with the SYMBOL function, it behaves the same as the FONT= option in the SYMBOL statement. By default, no font is specified and the symbol that is specified by the TEXT variable is taken from the special symbol table. If you use STYLE to specify a symbol font, such as Marker, the string that is assigned by the TEXT variable is the character code for a symbol. If you use STYLE to specify a text font, such as Swiss, the string assigned by the TEXT variable is displayed as text. See the FONT= option of the SYMBOL statement for details.

Note: Java does not support the STYLE variable. However, you can use special symbols from the MARKER font by using the SYMBOL function.

See Chapter 22, “Specifying Fonts in SAS/GRAPH Programs,” on page 349 for more information about specifying fonts.

If the value of the STYLE variable is missing, SAS/GRAPH software searches for a font specification in this order:

1. the font specified by the FTEXT= graphics option
2. the device-resident font, if the device supports one
3. the SIMULATE font.

STYLE Variable: Images

Determines the appearance of images specified with the IMGPATH variable and the IMAGE function.

Type: character

Default: 'TILE'

Syntax

STYLE='TILE' | 'FIT' | 'SINGLE';

Variable Values

TILE
uses copies of the image to fill the image area.

FIT
stretches one instance of the image to fill the image area.

SINGLE
centers a single instance of the image on the specified coordinates. Any part of the image that extends beyond the data area is clipped. SINGLE is valid starting with SAS 9.4M5.

Details

This version of the STYLE variable can be used only with the “IMAGE Function” on page 773.

STYLE Variable: Arrows

Specifies the type of arrowhead for arrows.

Type: character

Length: 8

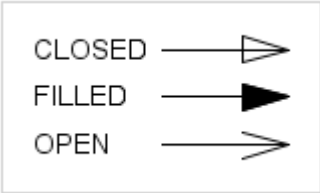
Default: OPEN

Syntax

STYLE='CLOSED' | 'FILLED' | 'OPEN';

Details

The following figure shows an example of each arrowhead style.



STYLE Variable: Patterns

Specifies a pattern for bars, pies, frames, and rectangles

Type:	character
Length:	8
Default:	EMPTY | PEMPTY | MEMPTY
Restriction:	Partially supported by Java and ActiveX

Syntax

STYLE='*fill-pattern*';

Variable Values

fill-pattern
specifies a pattern to use with the graphics element. The value for *fill-pattern* is function dependent as shown in the following table.

Function	Fill Pattern [1, 2]	Description
BAR, FRAME	SOLID | S	Fill with a solid color.
	EMPTY | E	No fill.

Function	Fill Pattern [1, 2]	Description
	style <density>	*style* R for right-slanted fill lines, L for left-slanted fill lines, or X for crossing fill lines. *density* Whole numbers 1 through 5 specify increasing thickness for the fill lines.
PIE	PSOLID | PS	Fill with a solid color.
	PEMPTY | PE	No fill.
	P*density* *<style<angle>>*	*density* Whole numbers 1 through 5 specify increasing thickness for the fill lines. *style* N, the default, can specify parallel fill lines; X can specify crossed fill lines. *angle* can specify the angle of the fill lines. Values range from 0 to 360. The angle is measured counterclockwise from the horizontal. The default is 0°, which draws horizontal lines. For example, if STYLE='P5N15', a pie slice with a fill of parallel lines is produced. The fill uses the heaviest density to draw the lines, and the parallel lines are drawn at a 15-degree angle from perpendicular to the radius of the pie slice. **Note:** Java and ActiveX support only PSOLID and PEMPTY values and default to PEMPTY if any other value is used. An illustration of these pattern styles is provided in the definition of the VALUE= option of the PATTERN statement.
POLY	MSOLID | MS	Fill with a solid color.
	MEMPTY | ME	No fill.
	M*density* *<style<angle>>*	*density* Whole numbers 1 through 5 specify increasing thickness for the fill lines. *style* N, the default, can specify parallel fill lines; X can specify crossed fill lines. *angle* Can specify the angle of the fill lines. Values range from 0 to 360. The angle is measured counterclockwise from the

Function	Fill Pattern [1, 2]	Description
		vertical. The default is 0°, which draws vertical lines. **Note:** Java and ActiveX support only MSOLID and MEMPTY and default to MEMPTY if any other value is used. An illustration of these pattern styles is provided in the definition of the VALUE= option of the PATTERN statement.

1 Java and ActiveX support only SOLID and EMPTY. EMPTY is the default if any other value is used.

2 An illustration of these pattern styles is provided in the VALUE= option description for the PATTERN statement on page 455.

SUBGROUP Variable

Positions graphics elements within subgrouped bars of a vertical or horizontal bar chart produced by the GCHART procedure.

Type: Numeric or character; must match the type of the SUBGROUP variable used in the GCHART procedure.

Length: Should match the length of the SUBGROUP= variable in the GCHART procedure.

Default: none

Restriction: The bar charts must have been produced using the SUBGROUP= option.

Syntax

SUBGROUP=*subgroup-value*;

Variable Values

subgroup-value
references value(s) of the SUBGROUP= variable in the GCHART procedure either as a variable name or as an explicit data value. *Subgroup-value* can have one of the following forms:

subgroup-variable
the name of a subgroup variable.

subgroup-data-value
a specific numeric data value.

subgroup-data-value
a specific character data value.

Generally, specify a variable name if you want to annotate all of the bars in a horizontal or vertical bar chart. To annotate a bar chart for a specific value of the SUBGROUP variable, specify a specific value.

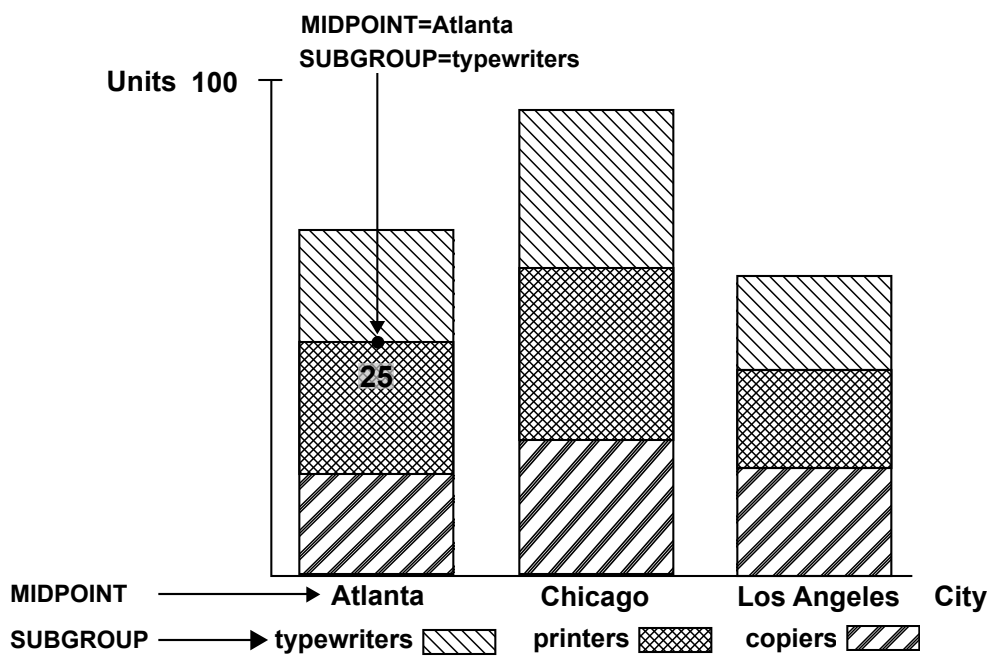
Details

Using the SUBGROUP Variable

The SUBGROUP variable positions the graphics elements in subgroup segments in vertical and horizontal bar charts. Using the SUBGROUP variable is similar to using the X and Y variables with data system coordinates. For example, in a vertical bar chart that produces a bar for each city in a data set, you can easily label the subgroups in each bar by setting *subgroup-variable* to the GCHART variable by which the bar is being subgrouped. This variable provides the *y* coordinate of the label. Therefore, you do not specify a competing value for y, but instead specify *y=.* or *y=y*.

The MIDPOINT variable works well with the SUBGROUP variable to provide the *x* coordinate. In this example, if you set the MIDPOINT variable to the GCHART variable that contains the names of the cities, the MIDPOINT variable provides your *x* coordinate. Rather than providing the X and Y variables, you would use the SUBGROUP and MIDPOINT variables. Figure 30.95 on page 836 shows how the SUBGROUP variable works with the MIDPOINT variable to label the bars of a vertical bar chart.

Figure 30.9 *Using the SUBGROUP Variable to Position a Label in a Bar Chart*



The label showing the number of printers sold in Atlanta is positioned by the values that are assigned to these Annotate variables:

- MIDPOINT=CITY (where CITY is a variable in the GCHART data set)
- SUBGROUP=ITEM (where ITEM is a variable in the GCHART data set)

Functions

You can use the SUBGROUP variable only with the data coordinate system 1, 2, 7, or 8, and with these functions:

BAR	MOVE	POINT
DRAW	PIE	SYMBOL
LABEL	PIECNTR	

TEXT Variable

Specifies the text or symbol to be placed on the graphics output.

Type:	character
Length:	up to 200
Default:	blank string

Syntax

TEXT='*text-string*' | '*special-symbol*';

Variable Values

text-string
specifies the text that is used as a label (LABEL or COMMENT function) or symbol (SYMBOL function). The maximum length for *text-string* is 200 characters.

special-symbol
specifies the name of a special symbol. The maximum length for *special-symbol* is eight characters. See Table 25.65 on page 492.

Details

Defining the TEXT Variable

Define the TEXT variable with sufficient length to contain all of the characters in your text string. If you need longer strings, use separate observations and POSITION='0' to continue the text.

Use a LENGTH statement to set the length of the TEXT variable if the length of a text string is longer than one character.

Functions

You can use the TEXT variable with these functions:

COMMENT
LABEL
SYMBOL

WHEN Variable

Specifies when the function is performed in relation to generating other graphics output for the procedure or in relation to generating other Annotate graphics.

Type: character

Length: 1

Default: B

Syntax

WHEN='B' | 'A';

Variable Values

B

draws the annotation before the graph is drawn.

Notes A missing value is equivalent to specifying B.

The frame of some plot types is drawn before the annotations when WHEN="B". If you use the Annotate facility to draw a background and you want your graph frame to be visible, then you can use the BAR function to draw a frame. The following annotate statements draw a white graph frame:

```
xsys="3"; ysys="3"; when="b";
function="move"; x=0; y=0; output;
function="bar"; style="solid"; color="white"; x=100; y=100; output;
```

A

draws the annotation after the graph is drawn.

Tip To prevent your graph from occluding your annotations, specify WHEN="A". This is particularly important for interactive presentations, where the back wall of the graph might be made visible by default.

Details

The WHEN Variable and Annotate Data Set Processing

Normally, observations in an Annotate data set are processed sequentially. If you use the WHEN variable, all those observations with a WHEN value of B are processed first. The procedure output is then processed (if one is to be produced), and finally the observations with a WHEN value of A are processed.

Functions

You can use the WHEN variable with these functions:

BAR	MOVE	POLY
DRAW	PIE	POLYCONT
DRAW2TXT	PIECNTR	SYMBOL
FRAME	PIEXY	
LABEL	POINT	

WIDTH Variable

Determines the thickness of a line.

Type:	numeric
Length:	8
Default:	1

Syntax

WIDTH=*line-thickness*;

Variable Values

line-thickness
specifies the thickness of the line that is to be drawn.

Details

The WIDTH variable can be used only with the PIE function.

The WIDTH variable always specifies a width in pixels. The coordinate system that you specify with the HSYS variable does not affect the WIDTH variable.

Note: The WIDTH variable is not supported by Java when your graph contains a depth axis (for example, graphs that are created by the SCATTER statement of the G3D procedure).

Note: For ActiveX output, the maximum line thickness is ten pixels. If you specify a greater value, then the value is reduced to 10.

X Variable

Identifies the *x* coordinate of where a graphics element is to be drawn.

Type: numeric

Default: value of XLAST or XLSTT

Syntax

X=*horizontal-coordinate*;

Variable Values

horizontal-coordinate
specifies the X coordinate of where the graphics element is to be drawn.

Details

Using the X Variable

Specify a corresponding vertical coordinate when using the X variable. This vertical coordinate can be specified with the Y, YC, MIDPOINT, or SUBGROUP variables, depending on the type of graph that you are annotating.

The X variable uses the units that are specified in the XSYS variable. If you use XSYS='2' and the data axis is entered as character, use the XC variable instead of the X variable.

If the value of the X variable is missing for a function that requires it, the value of the XLAST variable or the XLSTT variable is used. The XLAST variable is used with nontext functions, and the XLSTT variable is used with text functions.

Functions

You can use the X variable with these functions:

ARROW	LABEL	POINT
BAR	MOVE	POLY
DRAW	PIE	POLYCONT
IMAGE	PIECNTR	SYMBOL

Note: The X or XC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the horizontal coordinate.

XC Variable

Identifies the *x* coordinate of a graphics element when the coordinate value is character.

Type:	character
Length:	Should match that of the plot variable in the procedure. The maximum length is 32 characters in SAS 9.4M2 and in earlier releases. Starting with SAS 9.4M3, the maximum length is 256 characters.
Default:	the value of XLAST or XLSTT
Restriction:	Used only with output from the GCHART and GPLOT procedures. Ignored if the axes are numeric.

Syntax

XC='*character-type-horizontal-coordinate*';

Variable Values

character-type-horizontal-coordinate
specifies the X coordinate of the graphics element to be drawn when the coordinate value is a character.

Details

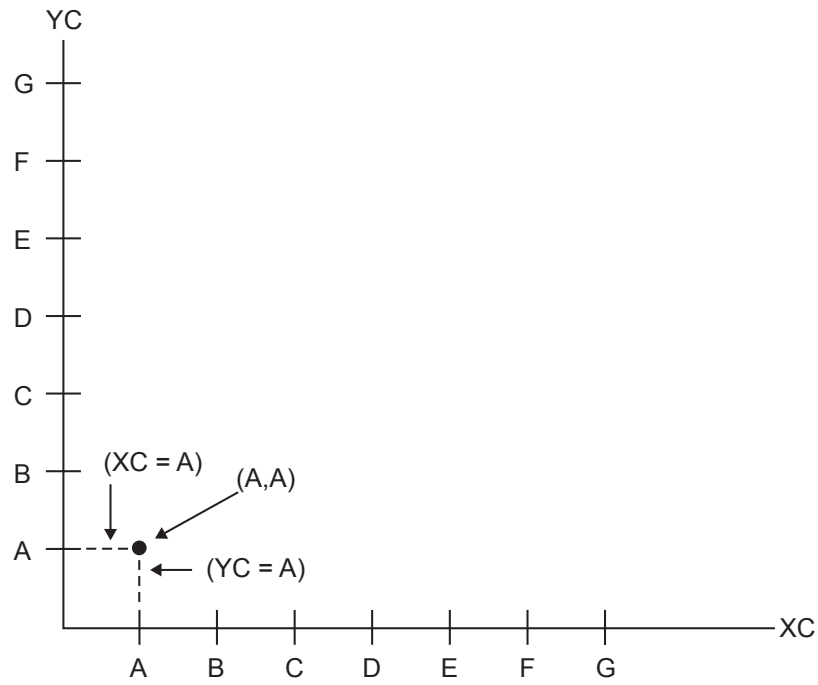
Using the XC Variable

The XC variable is the character equivalent of the X variable. Use XC when the axis values are character. You must also specify a value of 2 (absolute data values) for

the XSYS variable. (See also "XSYS Variable" on page 843.) If you use a value other than 2 for the XSYS variable, the graphics output is not displayed properly.

Figure 30.96 on page 842 illustrates the XC variable.

Figure 30.10 *Using the XC and YC Variables with Character Data*



Note: The X or XC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the horizontal coordinate.

CAUTION
Do not use the X and XC variables in the same data set. Using both X and XC variables in the same data set can cause unpredictable results.

Functions

You can use the XC variable with these functions:

BAR	PIE	POLYCONT
DRAW	PIECNTR	SYMBOL
LABEL	POINT	
MOVE	POLY	

XLAST, YLAST Variables

Track the last values specified for the X and Y variables when X and Y are used with nontext functions.

Comparisons

The coordinate values that are stored in the (XLAST, YLAST) variables are automatically updated by these nontext functions: BAR, DRAW, MOVE, PIE, and POINT. These values are then available for use by other nontext functions that follow in the DATA step. (The DRAW2TXT graphics function uses XLAST and YLAST but does not update them.)

Because (XLAST, YLAST) are updated internally, you cannot specify values for them. However, their values can be manipulated by these programming functions:

CNTL2TXT	PUSH
PIECNTR	SWAP
PIEXY	TXT2CNTL
POP	

XLSTT, YLSTT Variables

Track the last position for the X and Y variables when X and Y are used with text-handling functions.

Comparisons

The coordinate values stored in the (XLSTT, YLSTT) variables are automatically updated by the LABEL and SYMBOL text functions. These values are then available for use by other text functions that follow in the DATA step.

Because (XLSTT, YLSTT) are updated internally, you cannot specify values for them. However, their values can be manipulated by these programming functions:

CNTL2TXT	PUSH
DRAW2TXT	SWAP
POP	TXT2CNTL

XSYS Variable

Defines the coordinate system and area of the output used by the X and XC variables to display the Annotate graphics.

Type:	character
Length:	1
Default:	4

Syntax

XSYS='*coordinate-system*';

Variable Values

coordinate-system
specifies a value that represents a coordinate system. Values can be 1 through 9 and A through C. See the following table.

Absolute Systems	Relative Systems	Coordinate System Units
1	7	Percentage of data area
2	8	Data values
3	9	Percentage of graphics output area
4	A	Cell in graphics output area
5	B	Percentage of procedure output area
6	C	Cell in procedure output area

These values are also used by the HSYS and YSYS variables. See "Coordinate Systems" on page 734 for a description of the areas and coordinate systems.

Details

Using the XSYS Variable with the X and XC Variables

The coordinate system that you specify with the XSYS variable affects how the function interprets the value of the X or XC variable.

Note: Not all coordinate systems can be used with all Annotate variables. For any restrictions, see the individual variables in this section.

Functions

You can use the XSYS variable with these functions:

ARROW	LABEL	POINT
BAR	MOVE	POLY

DRAW	PIE	POLYCONT
FRAME	PIECNTR	SYMBOL

The behavior of the XSYS variable is function dependent for the BAR, DRAW, and FRAME functions.

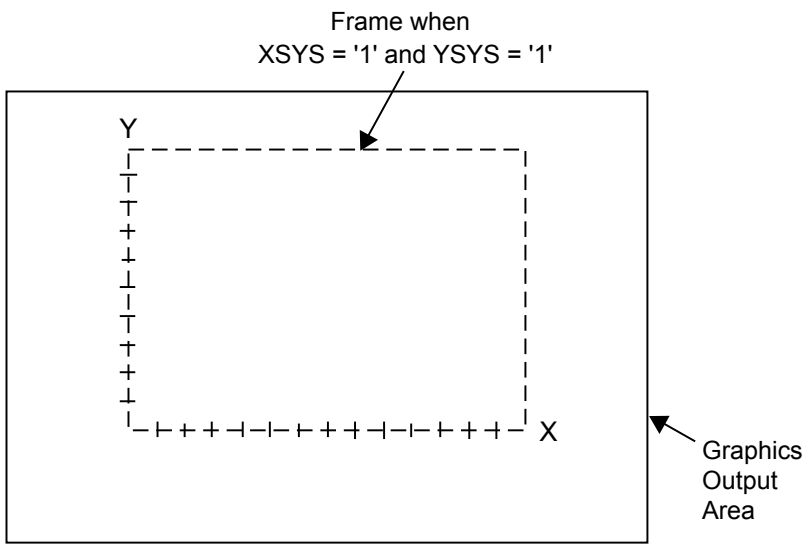
XSYS Variable Behavior In the BAR and DRAW Functions

The coordinate system that you specify with the XSYS variable affects how the function interprets the value of the X or XC variable. If XC is used, XSYS='2' must also be used.

XSYS Variable Behavior In the FRAME Function

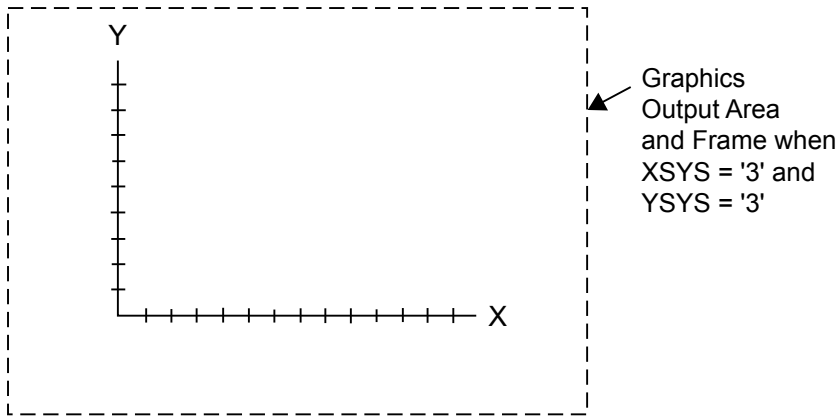
The XSYS and YSYS variables define the area enclosed by the frame. To draw a frame that encloses the axis area, use XSYS='1' and YSYS='1', as shown in the following figure.

Figure 30.11 *Frame Created When XSYS='1' and YSYS='1'*



To draw a frame that encloses the entire graphics output area, specify XSYS='3' and YSYS='3', as shown in the following figure.

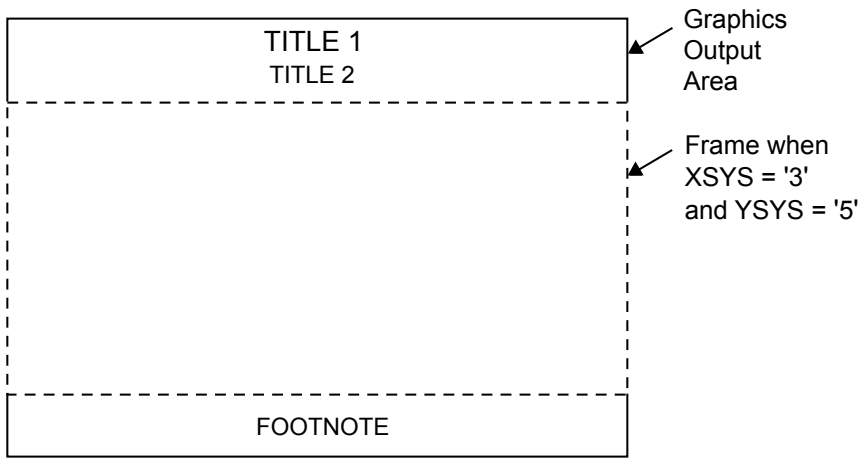
Figure 30.12 Frame Created When XSYS='3' and YSYS='3'



To limit the size of the frame to the size of the procedure output area, specify a value of 5 for XSYS and YSYS.

Note that the values of XSYS and YSYS can differ. You can specify a frame that occupies the entire width of the graphics output area and only the vertical width of the procedure output area. To do this, specify XSYS='3' and YSYS='5', as shown in the following figure.

Figure 30.13 Frame Created When XSYS='3' and YSYS='5'



Y Variable

Identifies the *y* coordinate of where a graphics element is to be drawn.

Type: numeric

Default: value of YLAST or YLSTT

Syntax

Y=*vertical-coordinate*;

Variable Values

vertical-coordinate
specifies the Y coordinate of where the graphics element is to be drawn.

Details

Using the Y Variable

Specify a corresponding horizontal coordinate when using the Y variable. You can specify the horizontal coordinate with the X, XC, MIDPOINT, or SUBGROUP variable, depending on the type of graph that you are annotating.

The Y variable uses the units specified in the YSYS variable. If you use YSYS='2' and the axis data is type character, use the YC variable instead of the Y variable.

If the value of the Y variable is missing for a function that requires it, the value of the YLAST variable or the YLSTT variable is used. The YLAST variable is used for nontext functions, and the YLSTT variable is used for text functions.

Functions

You can use the Y variable with these functions:

ARROW	LABEL	POINT
BAR	MOVE	POLY
DRAW	PIE	POLYCONT
IMAGE	PIECNTR	SYMBOL

Note: The Y or YC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the vertical coordinate.

YC Variable

Identifies the *y* coordinate of a graphics element when the coordinate value is character.

Type: character

Length: Should match that of the plot variable in the procedure. The maximum length is 32 characters in SAS 9.4M2 and in earlier releases. Starting with SAS 9.4M3, the maximum length is 256 characters.

Default: YLAST | YLSTT

Restriction: Used only with output from the GCHART and GPLOT procedures. Ignored if the axes are numeric.

Syntax

YC='*character-type-vertical-coordinate*';

Variable Values

character-type-vertical-coordinate
specifies the Y coordinate of the graphics element to be drawn when the coordinate value is a character.

Details

Using the YC Variable

The YC variable is the character equivalent of the Y variable. Use YC when the axis values are character. You must also specify a value of 2 (absolute data values) for the YSYS variable. (See "YSYS Variable" on page 849.) If you use a value other than 2 for the YSYS variable, the graphics output is not displayed properly.

See Figure 30.96 on page 842 for an illustration of the YC variable.

Note: The X or XC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the horizontal coordinate.

CAUTION
Do not use Y and YC variables in the same data set. Using both Y and YC variables in the same data set can cause unpredictable results.

Functions

You can use the YC variable with these functions:

BAR	PIE	POLYCONT
DRAW	PIECNTR	SYMBOL
LABEL	POINT	
MOVE	POLY	

YSYS Variable

Defines the coordinate system and area of the output used by Y and YC to display the Annotate graphics.

Type: character

Length: 1

Default: 4

Syntax

YSYS='*coordinate-system*';

Variable Values

coordinate-system

specifies a value that represents a coordinate system. Values can be 1 through 9 and A through C. See the following table.

Absolute Systems	Relative Systems	Coordinate System Units
1	7	Percentage of data area
2	8	Data values
3	9	Percentage of graphics output area
4	A	Cell in graphics output area
5	B	Percentage of procedure output area
6	C	Cell in procedure output area

These values are also used by the HSYS and XSYS variables. See "Coordinate Systems" on page 734 for a description of the areas and coordinate systems.

Details

Using the YSYS Variable with the Y and YC Variables

The coordinate system that you specify with the YSYS variable affects how the function interprets the value of the Y or YC variable.

Note: Not all coordinate systems can be used with all Annotate variables. For any restrictions, see the individual variables in this section.

Functions

The YSYS variable is function dependent, as defined in the "XSYS Variable" on page 843.

You can use the YSYS variable with these functions:

ARROW	LABEL	POINT
BAR	MOVE	POLY
DRAW	PIE	POLYCONT
FRAME	PIECNTR	SYMBOL

Z Variable

Identifies the *z* coordinate of where a graphics element is to be drawn.

Type:	numeric
Length:	8
Default:	none
Restriction:	For Java or ActiveX, you can use the Z variable with GMAP, GCHART, GCONTOUR, GPLOT, and G3D, for example, to add annotations above the plane of the map. For other devices, the Z variable is used only with output from the G3D procedure.

Syntax

Z=depth-coordinate;

Variable Values

depth-coordinate
specifies the Z coordinate of where the graphics element is to be drawn.

Note The Z variable uses the units that are specified in the ZSYS variable.

Details

You can use the Z variable with these functions:

ARROW	LABEL	POINT
BAR	MOVE	POLY
DRAW	PIE	POLYCONT
IMAGE	PIECNTR	SYMBOL

ZSYS Variable

Defines the coordinate system and area of the output used by Z variable to display the Annotate graphics.

Type: character

Length: 1

Default: 2

Syntax

ZSYS='*coordinate-system*';

Variable Values

coordinate-system
specifies a value that represents a coordinate system. Values can be 1, 2, 7, or 8 as shown in the following table:

Absolute Systems	Relative Systems	Coordinate System Units
1	7	Percentage of data area
2	8	Data values

See "Coordinate Systems" on page 734 for a description of the areas and coordinate systems.

Details

Using the ZSYS Variable with the Z Variable

The coordinate system that you specify with the ZSYS variable affects how the function interprets the value of the Z variable.

Note: Not all coordinate systems can be used with all Annotate variables. For any restrictions, see the individual variables in this section.

Functions

You can use the ZSYS variable with these functions:

ARROW	LABEL	POINT
BAR	MOVE	POLY
DRAW	PIE	POLYCONT
IMAGE	PIECNTR	SYMBOL

31

Annotate Macro Dictionary

About the Annotate Macros

You can use Annotate macros within a SAS DATA step to simplify the process of creating Annotate observations. With a macro, you specify a function and assign variable values in one step without having to write explicit variable assignment statements. You can mix assignment statements and macro calls in the same DATA step.

This section describes all of the Annotate macros including the complete syntax and a description of the parameters. For more information about accessing and using macros, and for a summary of operations performed by the Annotate macros, see "Using Annotate Macros" on page 854.

Using Annotate Macros

Macro-Call Syntax

The general syntax of an Annotate macro call is

```
%macro-name(parameter1, parameter2, ...);
```

In general, the macro name represents a function and the parameters contain the values for the variables that can be used with the function. All macros except DCLANNO, SYSTEM, and SEQUENCE output one or more observations.

The Annotate macro parameters are positional parameters, which must be specified in the proper order in the macro call. The parameters are either numeric or character. Numeric parameters can be numeric constants or numeric variable names that have been initialized to the appropriate value. Most character parameters must be expressed as literal character strings without quotation marks. Exceptions are the text values that are used with the COMMENT and LABEL macros, which can be expressed as character strings enclosed in quotation marks or as character variable names.

When generating observations, an Annotate macro assigns the macro parameter values to the corresponding Annotate variables. The resulting observations are equivalent to those that you would create manually using assignment statements. For example, the following two statements are equivalent:

```
%LABEL(10, 15, "Graph", black, 0, 0, centb, 8);

function="label"; x=10; y=15; text="Graph"; color="black"; style="centb";
position="8"; output;
```

Making the Macros Available

Before you can use the Annotate macros, you must run the %ANNOMAC autocall macro to compile the Annotate macros as follows:

```
%annomac;
```

When executed, the %ANNOMAC macro writes a message to the SAS log indicating that the Annotate macros are available. The message also describes how to get help for using the macros. The macros remain available for the duration of your SAS session. When you start a new SAS session, you must run the %ANNOMAC macro again to make the Annotate macros available in the new session.

Annotate Macro Task Summary

The following table summarizes the tasks performed by the Annotate macros.

Table 31.1 *Tasks with Annotate Macros*

Tasks	Macro to use
Assign values of (XLSTT,YLSTT) to (XLAST,YLAST)	%TXT2CNTL;
Begin drawing a polygon	%POLY(x, y, color, style, line);
Continue drawing a polygon	%POLYCONT(x, y, color);
Copy (XLAST,YLAST) to (XLSTT,YLSTT)	%CNTL2TXT;
Declare all variables	%DCLANNO;
Draw an arrow	%ARROW(x1, y1, x2, y2, color, line, size, angle, CLOSED | FILLED | OPEN);
Draw a bar	%BAR(x1, y1, x2, y2, color, line, style);
Draw a circle	%CIRCLE(x, y, size, color);
Draw a frame	%FRAME(color, line, size, style);
Draw a line from (XLAST,YLAST) to (XLSTT,YLSTT)	%DRAW2TXT(color, line, size);
Draw a line from previous point	%DRAW(x, y, color, line, size);

Tasks	Macro to use
Draw a line	%LINE(x1, y1, x2, y2, color, line, size);
Draw a pie slice or arc	%SLICE(x, y, angle, rotate, size, color, style, WHOLE | NONE | LEAD | TRAIL | BOTH);
Draw a rectangle	%RECT(x 1,y 1,x 2,y 2, color, line, size);
Draw text	%LABEL(x, y, text, color, angle, rotate, size, style, position);
Exchange the values of (XLAST,YLAST) and (XLSTT,YLSTT)	%SWAP;
Move to a point near a pie slice	%PIEXY(angle, size);
Move to a point without drawing	%MOVE(x, y);
Put values into a stack	%PUSH;
Retrieve values from a stack	%POP;
Scale and move input	%SCALET(ptx, pty, x0, y0, x1, y1, x0, vy0, vx1, vy1);
Scale input	%SCALE(ptx, pty, x0, y0, x1, y1, x0, vy0, vx1, vy1);
Set the coordinate system for the observation	%SYSTEM(xsys, ysys, hsys);
Set when to draw an observation	%SEQUENCE(BEFORE | AFTER);
Write a comment to the data set	%COMMENT(text);

Dictionary

%ANNOMAC Annotate Macro

Compiles Annotate macros and makes them available for use.

Variables written out: none directly

Note: Starting with SAS 9.4M6, it is not necessary to run the %ANNOMAC macro before using the %CENTROID macro.

Syntax

%ANNOMAC;

Comparisons

In a SAS session, you must submit the %ANNOMAC macro before you can use the Annotate macros.

%ARROW Annotate Macro

Draws an arrow from (X1, Y1) to (X2,Y2).

Updates: XLAST, YLAST

Variables written out: ANGLE, COLOR, FUNCTION, LINE, SIZE, STYLE, X, Y

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%ARROW (*x1*, *y1*, *x2*, *y2*, *color*, *line*, *size*, *angle*, CLOSED | FILLED | OPEN);

Parameters

x1*, *y1
specify coordinates for the start point of the arrow. Values can be coordinate numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

x2*, *y2
specify coordinates for the end point of the arrow. Values can be coordinate numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

color
specifies the color of the line using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

line
specifies the length of the sides of the arrowhead as a percentage of the graph area. The value can be a number, a numeric constant, or a numeric variable.

See "LINE Variable" on page 821 for valid *line* values.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the ARROW function for valid *size* values.

angle
specifies the angle of the tip of the arrowhead. The value can be a number, a numeric constant, or a numeric variable.

See "ANGLE Variable" on page 804 for the ARROW function for valid *angle* values.

CLOSED | FILLED | OPEN
specifies the arrowhead style.

See "STYLE Variable: Arrows" on page 832 for more information about the values.

Details

The point from which the line is drawn is usually set with the MOVE macro.

%BAR, %BAR2 Annotate Macros

Draws a rectangle using two sets of X/Y coordinates, which specify diagonal corners. You can specify the rectangle's line type, line color, fill type, and fill color.

Updates:	XLAST, YLAST
Variables written out:	COLOR, FUNCTION, LINE, STYLE, X, Y
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%BAR (*x1*, *y1*, *x2*, *y2*, *color*, *line*, *style*);

%BAR2(*x1*, *y1*, *x2*, *y2*, *color*, *line*, *style*, *width*);

Parameters

x1*, *y1
specify the location of the first corner of the bar. Values can be numeric coordinates, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

x2*, *y2
specify the location of second corner of the bar, which is drawn diagonal to the first corner. Values can be numeric coordinates, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

color
specifies the outline color and optional fill color using a character string without quotation marks.

See "COLOR Variable" on page 807

line
specifies which of the outlines of the bar are to be drawn. The value can be a number, a numeric constant, or a numeric variable.

See "LINE Variable" on page 821 for the BAR function for valid *line* values.

style
specifies the fill pattern for the bar using a character string without quotation marks.

See "STYLE Variable: Patterns" on page 833 for the BAR function for valid *style* values.

width
specifies the width of the outline and optional fill lines. The value can a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the DRAW function for valid *width* values.

%CENTROID Macro Autocall Macro

Retrieves the centroid positions of map data set polygons.

Variables written out:	X, Y, ID variables

Requirement: In SAS 9.4M5 and in earlier releases, you must run the %ANNOMAC macro before using the %CENTROID annotate macro. See "Making the Macros Available" on page 855. Starting with SAS 9.4M6, it is not necessary to run the %ANNOMAC macro before using the %CENTROID macro.

Note: Starting with SAS 9.4M6, the %CENTROID autocall macro is available with Base SAS. The macro is no longer part of SAS/GRAPH, nor is SAS/GRAPH required to be installed to use this macro.

Tip: Use this macro as an aid in adding labels to a map. It can also be used in conjunction with the Base SAS GEODIST function when computing distances. See the "GEODIST Function" in *SAS Functions and CALL Routines: Reference* for more information.

Syntax

%CENTROID (*input-data-set*, *output-data-set*, *list-of-id-variables* <, SEGONLY=*n*>);

Parameters

input-data-set
specifies a map data set. The input map data set must be sorted by the ID variables.

output-data-set
contains the ID variables and the X and Y variables.

list-of-id-variables
specifies the variables, each of which is assigned the centroid coordinates of each observation in the input-data-set. There is one observation for each unique set of ID values. If you specify more than one ID variable, then separate each variable with a space.

SEGONLY=*n*
specifies that only the *n*th segment of each map area is used to calculate the centroid. For example, if you specify SEGONLY=1, then only the first segment of each area is used.

Note: Be sure to end the list of ID variables with a comma before specifying a SEGONLY= value.

%CIRCLE Annotate Macro

Draws an empty circle with the center at (*x*, *y*).

Updates: XLAST, YLAST

Variables written out: ANGLE, FUNCTION, ROTATE, SIZE, STYLE, X, Y

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%CIRCLE (*x*, *y*, *size*, *color*);

Parameters

x, y
specify coordinates for the center of the circle. Values can be coordinate numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

size
specifies the radius of the circle. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for valid *size* values.

color
specifies the color of the circle using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

See Also

"%SLICE Annotate Macro" on page 878 to draw a filled circle

%CNTL2TXT Annotate Macro

Copies the values of the internal coordinates (XLAST, YLAST) to the text coordinate (XLSTT, YLSTT).

Updates:	XLSTT, YLSTT
Variables written out:	FUNCTION
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%CNTL2TXT;

Comparisons

The %CNTL2TXT macro is useful when you are calculating the position of labels on a graph. For an example, see "CNTL2TXT Function" on page 766.

%COMMENT Annotate Macro

Inserts a comment into an Annotate data set.

Variables written out: FUNCTION, TEXT

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%COMMENT (*text-string*);

Parameters

text-string
specifies the text to insert in the Annotate data set. The value can be a character string enclosed in quotation marks or the name of a character variable.

See "TEXT Variable" on page 837

%DCLANNO Annotate Macro

Automatically sets the correct length and data type for all Annotate variables except the TEXT variable.

Variables written out: None

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%DCLANNO;

%DRAW Annotate Macro

Draws a line from (XLAST, YLAST) to the specified coordinate.

Updates:	XLAST, YLAST
Variables written out:	COLOR, FUNCTION, LINE, SIZE, X, Y
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%DRAW (*x*, *y*, *color*, *line*, *size*);

Parameters

x*, *y
specify coordinates for the end point of the line. Values can be coordinate numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

color
specifies the color of the line using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

line
specifies the line type (continuous or segmented). The value can be a number, a numeric constant, or a numeric variable.

See "LINE Variable" on page 821 for the DRAW function for valid *line* values.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the DRAW function for valid *size* values.

Details

The point from which the line is drawn is usually set with the MOVE macro.

%DRAW2TXT Annotate Macro

Draws a line from the coordinate (XLAST, YLAST) to the text coordinate (XLSTT, YLSTT).

Variables written out: COLOR, FUNCTION, LINE, SIZE

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%DRAW2TXT (*color*, *line*, *size*);

Parameters

color
specifies the color of the line using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

line
specifies the line type (continuous or segmented). The value can be a number, a numeric constant, or a numeric variable.

See "LINE Variable" on page 821 for the DRAW function for valid *line* values.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the DRAW function for valid *size* values.

%FRAME Annotate Macro

Draws a border around the portion of the display area defined by the reference system and can fill the area.

Variables written out: COLOR, FUNCTION, LINE, SIZE, STYLE

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%FRAME (*color*, *line*, *size*, *style*);

Parameters

color
specifies the outline color and the optional fill color using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

line
specifies a line type (continuous or segmented) for the frame outline and fill lines. The value can be a number, a numeric constant, or a numeric variable.

See "LINE Variable" on page 821 for the DRAW function for valid *line* values.

size
specifies the width of the frame outline and fill lines. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the DRAW function for valid *size* values.

style
specifies the fill pattern for the frame using a character string without quotation marks.

See "STYLE Variable: Patterns" on page 833 for the FRAME function for valid *style* values.

Details

See "%SYSTEM Annotate Macro" on page 881 for information about setting the reference system.

%LABEL Annotate Macro

Places a text label at the specified coordinates.

Updates:	XLSTT, YLSTT
Variables written out:	ANGLE, COLOR, FUNCTION, POSITION, ROTATE, SIZE, STYLE, TEXT, X, Y
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%LABEL (*x*, *y*, *text-string*, *color*, *angle*, *rotate*, *size*, *style*, *position*);

Parameters

x, y
specifies the location of the text string. Values can be coordinate numbers, numeric constants, or numeric variables. The position of the text string relative to *x*, *y* is determined by the *position* parameter.

See "X Variable" on page 840 and "Y Variable" on page 846

text-string
specifies the text of the label. The value can be a character variable name or a character string enclosed in quotation marks.

See "TEXT Variable" on page 837

color
specifies the color of the text string using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

angle
specifies the angle of the text string with respect to the horizontal. The value can be a number, a numeric constant, or a numeric variable. The *x*, *y* coordinates specify the pivot point, and the *position* parameter positions the text relative to *x*, *y*.

See "ANGLE Variable" on page 804 for the LABEL function for valid *angle* values.

rotate
specifies the rotation angle of each character in the text string. The value can be a number, a numeric constant, or a numeric variable.

See "ROTATE Variable" on page 828

size
specifies the size of the text string. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the LABEL function for valid *size* values.

style
specifies the text font, using a character string without quotation marks.

See "STYLE Variable: Fonts" on page 831

position
specifies the placement and alignment of the text string relative to the *x*, *y* coordinates, using a text string without quotation marks.

See "POSITION Variable" on page 825

%LINE Annotate Macro

Draws a line between two sets of coordinates.

Updates:	XLAST, YLAST
Variables written out:	COLOR, FUNCTION, LINE, SIZE, X, Y
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%LINE (*x1*, *y1*, *x2*, *y2*, *color*, *line*, *size*);

Parameters

x1*, *y1
specify the coordinates of the start of the line. Values can be numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

x2*, *y2
specify the coordinates of the end of the line. Values can be numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

color
specifies the color of the line using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

line
specifies the line type, which can be continuous or segmented. The value can be a number, a numeric constant, or a numeric variable.

See "LINE Variable" on page 821 for the DRAW function for valid *line* values.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the DRAW function for valid *size* values.

%MAPLABEL Annotate Macro

Creates an output data set that can be used with the ANNO= option for PROC GMAP.

Variables written out: FUNCTION, STYLE, COLOR, SIZE, HSYS

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%MAPLABEL (*map-dataset*, *attr-dataset*,*output-dataset*,*label-var*,*id-list*,*font=font_name*,*color=n*,
size=n,*hsys=n*);

Parameters

map-dataset
the name of the map to be annotated.

attr-dataset
the name of the data set containing the text to be shown on each ID value.

output-dataset
the name of the Annotate data set created by the macro.

label-var
the name of the label variable to place on the map (the text for annotate).

id-list
the list of ID variables that you would issue in PROC GMAP to create the map. These values need to be on both the map and attribute data sets. If you also supply the SEGMENT variable, then every polygon will get a value. Without the SEGMENT variable, only one label per ID set will be shown over the collection of polygons. For example, Hawaii with SEGMENT gets a label on each island, whereas without SEGMENT, there is only one label centered on the entire set of islands.

font
specifies a font name for the STYLE variable.

color
specifies a value for the COLOR variable.

size
specifies a value for the SIZE variable. Defaults to 2.

hsys
specifies a value for the HSYS variable. Defaults to 3.

%MOVE Annotate Macro

Moves to the (*x*, *y*) coordinate.

Updates: XLAST, YLAST

Variables written out: FUNCTION, X, Y

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%MOVE (*x*, *y*);

Parameters

x*, *y
specify new coordinates for the next annotation. Values can be numeric coordinates, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

%PIEXY Annotate Macro

Calculates a point in relation to the latest pie slice.

Updates: XLAST, YLAST

Variables written out: ANGLE, FUNCTION, SIZE, X, Y

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%PIEXY (*angle*, *size*);

Parameters

angle

specifies the angle used to calculate the point, relative to the center of the latest pie slice. The value can be a number, a numeric constant, or a numeric variable.

See "ANGLE Variable" on page 804 for the PIEXY function for valid *angle* values.

size

specifies the radius multiplier that works with the *angle* parameter to determine the location of the point. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the PIEXY function for valid *size* values.

Details

This macro is useful when you want to label a pie chart or a circle.

When you use this macro, the Annotate facility expects a slice to have been previously drawn. If a slice has not been drawn or if the "PIECNTR Function" on page 786 has not been processed, you can get erroneous results.

%POLY, %POLY2 Annotate Macros

Begins drawing a polygon at the specified coordinates and determines the color, fill pattern, and line type of the polygon.

Variables written out:	FUNCTION, COLOR, LINE, STYLE, X, Y,
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%POLY (*x*, *y*, *color*, *style*, *line*);

%POLY2(*x*, *y*, *color*, *style*, *line*, *width*);

Parameters

x, y

specify the starting point for a new polygon. Values can be numeric coordinates, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

color
specifies the optional polygon fill color using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

Tip Use the "%POLYCONT Annotate Macro" on page 871 to specify the color of the polygon outline.

See "COLOR Variable" on page 807

style
specifies the fill pattern for the polygon, using a character string without quotation marks.

See "STYLE Variable: Patterns" on page 833 for the POLY function for valid *style* values.

line
specifies the polygon's line type, which can be continuous or segmented. The value can be a number, a numeric constant, or a numeric variable.

See "LINE Variable" on page 821 for the POLY function for valid *line* values.

width
specifies the width of the polygon's outline and optional fill lines. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the POLY function for valid *width* values.

See Also

"POLY Function" on page 791

%POLYCONT Annotate Macro

Continues drawing the polygon to the next specified coordinates.

Variables written out:	COLOR, FUNCTION, X, Y
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%POLYCONT (*x*, *y*, *color*);

Parameters

x, y
specify the end point of the next line in the polygon. Values can be numeric coordinates, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

color
specifies the color of the polygon outline using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

Details

The first invocation of the %POLYCONT macro in the polygon-drawing sequence determines the outline color of that polygon. Subsequent color specifications for that polygon in later invocations of the %POLYCONT macro are ignored.

The polygon fill color and line type are specified in the initial "%POLY, %POLY2 Annotate Macros" on page 870.

%POP Annotate Macro

Removes the coordinates (XLAST, YLAST) and (XLSTT, YLSTT) from the LIFO system stack and updates the internal coordinate pairs with these retrieved values.

Updates:	XLAST, YLAST, XLSTT, YLSTT
Variables written out:	FUNCTION
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%POP;

Comparisons

Use the %POP macro when you want to access the values of (XLAST on page 842, YLAST) and (XLSTT on page 843, YLSTT variables that you previously stored with the "%PUSH Annotate Macro" on page 873 or function.

%PUSH Annotate Macro

Enters the coordinates (XLAST, YLAST) and (XLSTT, YLSTT) in a LIFO system stack.

Updates: XLAST, YLAST, XLSTT, YLSTT

Variables written out: FUNCTION, internal coordinates

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%PUSH;

Comparisons

The last-in, first-out (LIFO) stack provides a way to save previously calculated coordinates. It enables you to retain coordinate values for later use by utility functions without recalculating those values. In order to save coordinate values in the stack, you must explicitly push them onto the stack. See "Using the LIFO Stack" on page 744 for a description of the LIFO stack.

%RECT Annotate Macro

Draws a rectangle with diagonal corners at two specified points.

Updates: XLAST, YLAST

Variables written out: COLOR, FUNCTION, LINE, SIZE, X, Y

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%RECT (*x1*, *y1*, *x2*, *y2*, *color*, *line*, *size*) ;

Parameters

x1*, *y1
specify the coordinates of the first corner of the rectangle. Values can be numeric coordinates, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

x2*, *y2
specify the coordinates of the second corner of the rectangle, which is drawn diagonal to the first corner. Values can be numeric coordinates, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

color
specifies the color of the rectangular line using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

line
specifies the rectangle's line type, which can be continuous or segmented. The value can be a number, a numeric constant, or a numeric variable.

See "LINE Variable" on page 821 for the DRAW function for valid *line* values.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829 for the DRAW function for valid *size* values.

Details

The rectangle is drawn such that the first corner is diagonal to the second corner.

The %RECT macro produces rectangles that do not have fill patterns. Use the %BAR macro to generate filled rectangles. For more information, see "%BAR, %BAR2 Annotate Macros" on page 858..

%SCALE Annotate Macro

Scales input coordinates relative to the origin (0, 0) based on the relationship between two ranges of minima and maxima.

Variables written out:	X, Y
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%SCALE (*ptx*, *pty*, *x1*, *y1*, *x2*, *y2*, *vx1*, *vy1*, *vx2*, *vy2*);

Parameters

ptx*, *pty
specifies the coordinates to scale. Values can be numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

x1*, *y1
specifies the minima of the first range. Values can be numbers, numeric constants, or numeric variables.

x2*, *y2
specifies the maxima of the first range. Values can be numbers, numeric constants, or numeric variables.

vx1*, *vy1
specifies the minima of the second range. Values can be numbers, numeric constants, or numeric variables.

vx2*, *vy2
specifies the maxima of the second range. Values can be numbers, numeric constants, or numeric variables.

Details

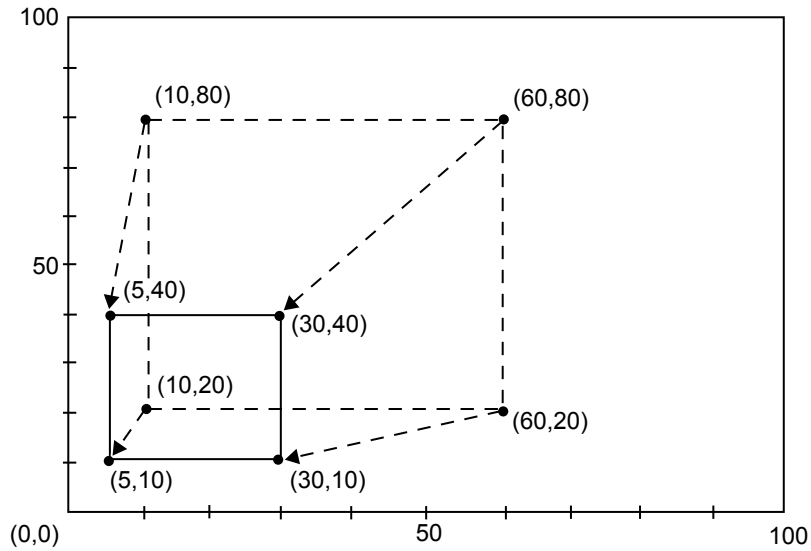
The %SCALE macro reduces or enlarges Annotate graphics elements that use two-dimensional, numeric coordinates. The %SCALE macro does not affect graphics elements that are drawn with text functions.

The difference between the %SCALE and %SCALET macros is that the %SCALE macro always places the origin at (0, 0) and plots the new coordinates with respect to that origin. The %SCALET macro plots the new coordinates with respect to the minima of the second range. For details, see "%SCALET Annotate Macro" on page 876.

The following example uses the %SCALE macro to reduce *x* and *y* coordinates by 50%, as shown in Figure 31.100 on page 876.

```
%SCALE(x, y, 0, 0, 100, 100, 0, 0, 50, 50);
```

Figure 31.1 *Using the %SCALE Macro to Reduce the Size of a Box*



%SCALET Annotate Macro

Scales input coordinates based on the relationship between two ranges of minima and maxima. The scaled coordinates are plotted relative to the minima of the second range.

Variables written out: X, Y

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%SCALET (*ptx*, *pty*, *x1*, *y1*, *x2*, *y2*, *vx1*, *vy1*, *vx2*, *vy2*);

Parameters

ptx*, *pty
specifies the coordinates to scale. Values can be numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

x1*, *y1
specifies the minima of the original range. Values can be numbers, numeric constants, or numeric variables.

x1*, *y2
specifies the maxima of the original range. Values can be numbers, numeric constants, or numeric variables.

vx1*, *vy1
specifies the minima of the second range using numeric values. Values can be numbers, numeric constants, or numeric variables. These coordinates are also used as the origin against which the scaled point is plotted.

vx2*, *vy2
specifies the maxima of the second range. Values can be numbers, numeric constants, or numeric variables.

Details

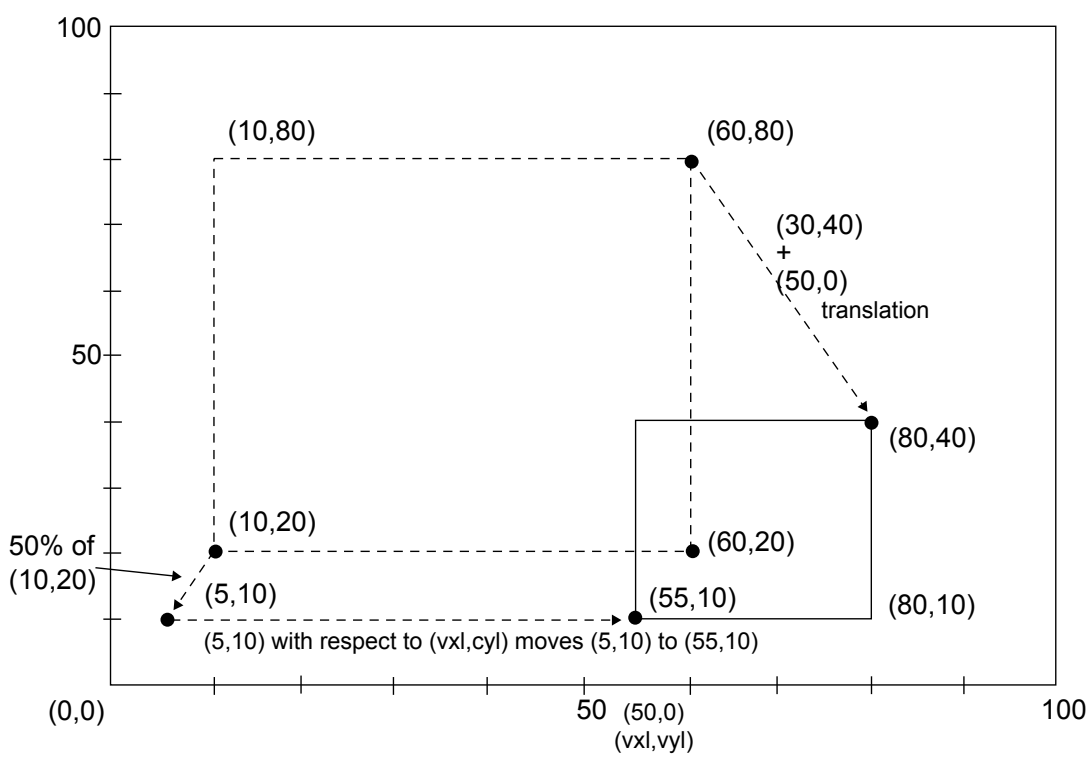
The %SCALET macro reduces or enlarges Annotate graphics elements that use two-dimensional numeric coordinates. The %SCALET macro does not affect graphics elements that are drawn with text functions.

The difference between the %SCALET and %SCALE on page 874 macros is that the SCALET macro plots the new coordinates with respect to minima of the second range (*vx1*, *vy1*). The %SCALE macro plots the new coordinates with respect to the origin (0, 0).

The following example uses the %SCALET macro reduces *x* and *y* coordinates by 50% and plots the new coordinates with respect to (50, 0), as shown in Figure 31.101 on page 877. :

```
%SCALET(x, y, 0, 0, 100, 100, 50, 0, 100, 50);
```

Figure 31.2 *Using the %SCALET Macro to Reduce the Size of a Box*



%SEQUENCE Annotate Macro

Specifies when to draw Annotate graphics elements, relative to the procedure's graphics output or relative to the other Annotate graphics drawn.

Variables written out: WHEN

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%SEQUENCE (BEFORE | AFTER);

Parameters

BEFORE
specifies that the annotation is drawn before the other procedure output is drawn.

AFTER
specifies that the annotation is drawn after the other procedure output is drawn.

See Also

"WHEN Variable" on page 838

%SLICE Annotate Macro

Draws a arc, pie slice, or circle, with available line types, colors, and fill types.

Updates: XLAST, YLAST

Variables written out: ANGLE, COLOR, FUNCTION, LINE, ROTATE, SIZE, STYLE, X, Y

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%SLICE (*x*, *y*, *angle*, *rotate*, *size*, *color*, *style*, WHOLE | NONE | LEAD | TRAIL | BOTH);

Parameters

x, y
specify the center point of the arc. Values can be numbers, numeric constants, or numeric variables.

See "X Variable" on page 840 and "Y Variable" on page 846

angle
specifies the starting point of the arc. The value can be a number, a numeric constant, or a numeric variable.

See "ANGLE Variable" on page 804 for the PIE function for valid *angle* values.

rotate
specifies the sweep of the arc. The value can be a number, a numeric constant, or a numeric variable.

Interaction When LINE=WHOLE on page 879 is specified, the %SLICE macro always specifies ROTATE=360 for the PIE function in the Annotate data set.

See "ROTATE Variable" on page 828 for the PIE function for valid *rotate* values.

size
specifies the radius of the arc. The value can be a number, a numeric constant, or a numeric variable.

See "SIZE Variable" on page 829

color
specifies the color of the arc outline and optional fill using a character string without quotation marks. Use an asterisk (*) to specify the previous value of the *color* parameter.

See "COLOR Variable" on page 807

style
specifies the fill pattern for the slice or circle, using a character string without quotation marks.

See "STYLE Variable: Patterns" on page 833 for the PIE function for valid *style* values.

WHOLE | NONE | LEAD | TRAIL | BOTH
specifies which lines of a pie slice are to be drawn. The value can be one of the following, which are shown in Figure 31.102 on page 880:

WHOLE
draws the entire 360 degree outside arc, starting at the specified angle, with no radius lines.

NONE
draws the outside arc from the specified starting angle through the specified rotation with no radius lines.

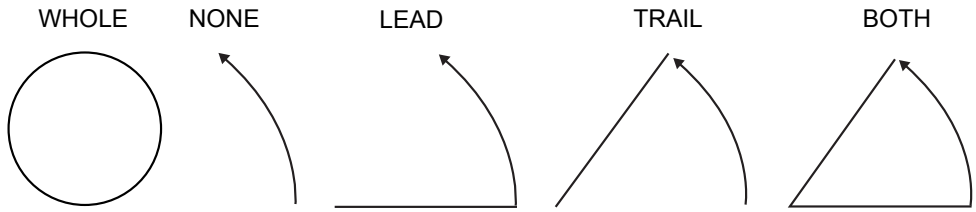
LEAD
draws the outside arc from the specified starting angle through the specified rotation and a radius line from the lead point to the center.

TRAIL
draws the outside arc from the specified starting angle through the specified rotation and a radius line from the trail point to the center.

BOTH
draws the outside arc from the starting angle through the specified rotation, a lead radius line, and a trail radius line.

The following figure illustrates each of the line values.

Figure 31.3 *LINE Values Used with the %SLICE Macro*



In the Annotate data set, these values are converted to the PIE function LINE values shown in the following table.

Macro %SLICE LINE Value	PIE Function LINE Value
WHOLE	0 [1]
NONE	0
LEAD	1
TRAIL	2
BOTH	3

1 When LINE=WHOLE is specified, LINE=0 and ROTATE=360 are specified for the PIE function in the Annotate data set.

See LINE Variable on page 821 for the PIE function.

%SWAP Annotate Macro

Exchanges control between (XLAST, YLAST) and text (XLSTT, YLSTT) coordinates.

Updates: XLAST, YLAST, XLSTT, YLSTT

Variables written out: FUNCTION

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%SWAP;

%SYSTEM Annotate Macro

Defines the Annotate reference systems and the XSYS, YSYS, and HSYS variables.

Variables written out: HSYS, XSYS, YSYS

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See "Making the Macros Available" on page 855.

Syntax

%SYSTEM (*xsys*, *ysys*, *hsys*);

Parameters

xsys*, *ysys*, *hsys
specifies values that represent a coordinate system and an area of the output.

Default %SYSTEM (4, 4, 4)

See "XSYS Variable" on page 843

"YSYS Variable" on page 849

"HSYS Variable" on page 813

Details

Note: Not all coordinate systems are valid with all Annotate variables or all SAS/GRAPH procedures. See "About the Annotate Functions" on page 759 for any restrictions that apply to the variable that you want to use.

The ZSYS variable on page 851 cannot be set through this macro. Use an explicit variable assignment instead:

```
zsys="value"; output;
```

See Coordinate Systems on page 734 for a description of the areas and coordinate systems.

%TXT2CNTL Annotate Macro

Assigns the values of the text (XLSTT, YLSTT) coordinates to the control (XLAST, YLAST) coordinates.

Updates:	XLAST, YLAST
Variables written out:	FUNCTION
Requirement:	You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 855.

Syntax

%TXT2CNTL;

Comparisons

Use the %TXT2CNTL macro when you want nontext functions to use the ending position of a text string as a starting or ending point.

PART 7

SAS/GRAPH Procedures

32

SAS Mapping Procedures

Procedures Documented in Another SAS Publication

The following mapping procedures are documented with related subject matter in *SAS/GRAPH and Base SAS: Mapping Reference*: GEOCODE, GINSIDE, GMAP, GPROJECT, GREDUCE, GREMOVE, and MAPIMPORT. The non-mapping SAS/GRAPH procedures are documented in this reference document.

Chapter 33

GANNO Procedure

Overview: GANNO Procedure

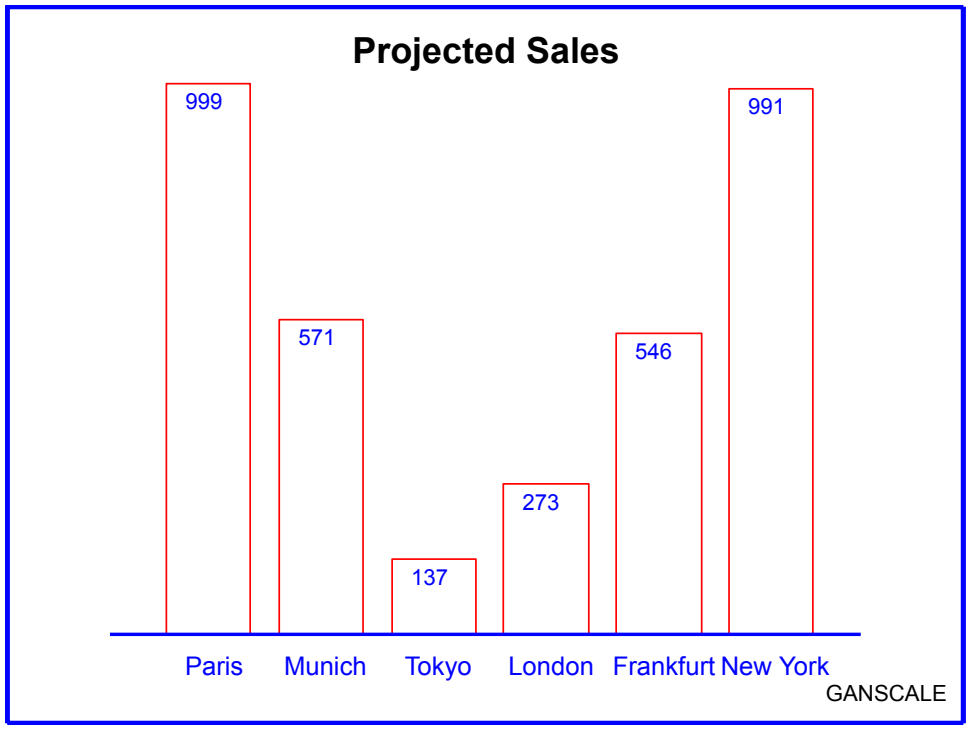
What Does the GANNO Procedure Do?

The GANNO procedure displays graphs created by Annotate data sets. The procedure can also be used to scale data-dependent graphics to fit the graphics output area. Note that the GANNO procedure ignores all currently defined title and footnote statements and some graphics option specifications, including BORDER. To include titles, footnotes, and graphics options along with your Annotate data set, use the GSLIDE procedure instead of the GANNO procedure. For more information about the Annotate facility, see Chapter 28, “Using Annotate Data Sets,” on page 721.

By default, both the GANNO and GSLIDE procedures scale graphics output from the data set to fill the entire graphics area. However, if you are using a data coordinate system and the data values are so large that some of the graphics elements do not fit in the graphics output area and are not displayed, you can use the GANNO procedure with the DATASYS option. This causes the procedure to scale the output to fit the available space. The GSLIDE procedure does not have this capability.

Figure 33.103 on page 888 displays output from an Annotate data set.

Figure 33.1 *Displaying Annotate Graphics with the GANNO Procedure*



The program for this graph is in "Example 1: Scaling Data-Dependent Output" on page 893 .

Syntax: GANNO Procedure

Requirement: An input Annotate data set is required.

Note: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, "Plotting a Cloud Analytic Services (CAS) In-Memory Table," on page 153.

PROC GANNO ANNOTATE=*Annotate-data-set*
<DATASYS>
<DESCRIPTION="*description*">
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>
<NAME="*name*">;

PROC GANNO Statement

Identifies the Annotate data set and draws the graphics output defined by that data set. It can also scale the output to accommodate data-dependent coordinate values and specify an output catalog.

Syntax

PROC GANNO ANNOTATE=*Annotate-data-set*
<DATASYS>
<DESCRIPTION="*description*">
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>
<NAME="*name*">;

Summary of Optional Arguments

DATASYS
indicates that absolute or relative data-dependent coordinates occur in the Annotate data set and scales the coordinates to fit the graphics output area.

DESCRIPTION="*description*"
specifies a description of the output.

GOUT=<*libref.*>*output-catalog*
specifies the SAS catalog in which to save the graphics output produced by the GANNO procedure.

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Required Argument

ANNOTATE=*Annotate-data-set*
specifies a data set that includes Annotate variables that identify graphics commands and parameters.

Alias ANNO=

See Chapter 28, “Using Annotate Data Sets,” on page 721

Optional Arguments

Options in the GANNO statement affect all graphs produced by that statement. You can specify as many options as you want and list them in any order.

DATASYS

indicates that absolute or relative data-dependent coordinates occur in the Annotate data set and scales the coordinates to fit the graphics output area. Use the DATASYS option only with Annotate data sets in which the coordinate system variables XSYS, YSYS, and HSYS specify the values 1, 2, 7, or 8.

Use the DATASYS option when graphics elements that were created with data-dependent variables do not fit in the graphics output area. This happens when the coordinate values generated by the data exceed a range of 0 to 100.

If you omit the DATASYS option, the GANNO procedure attempts to draw each graphics element according to the data values assigned to it, without scaling the values. If the range of data values is too large, some graphics elements will not be displayed.

See "Using the DATASYS Option to Scale Graphs" on page 892

Example "Example 1: Scaling Data-Dependent Output" on page 893

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

Alias DES=

Default OUTPUT FROM PROC GANNO

GOUT=<*libref*.>*output-catalog*

specifies the SAS catalog in which to save the graphics output produced by the GANNO procedure. If you omit the libref, the SAS/GRAPH software looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

See "Specifying the Catalog Name and Entry Name for Your GRSEGs " on page 138

Example "Example 2: Storing Annotate Graphics" on page 896

IMAGEMAP=*output-data-set*

creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is

not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255 and "Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" on page 262

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
  - If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

    Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

  - The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GANNO

Examples "Example 2: Storing Annotate Graphics" on page 896

"Example 3: Using the NAME= Option to Produce Multiple Graphs" on page 899

Details

Using the DATASYS Option to Scale Graphs

If your Annotate data set specifies a coordinate system that is based on data values (that is, XSYS, YSYS, and HSYS are assigned the values 1, 2, 7, or 8), the data values determine the size and location of the graphics elements on the output.

If the procedure that specifies the annotation generates axes (such as GPLOT or GCHART), by default the axes are scaled to accommodate the full range of data values and to fit in the procedure output area. Because all values are included in the axes, the graph displays all the Annotate output that is dependent on data values.

However, if the annotation displays with the GSLIDE or GANNO procedure, which do not generate axes, the data values might generate coordinate values that exceed the limits of the graphics output area.

In this case, you can use the DATASYS option to tell the procedure that the Annotate data set contains data-dependent coordinates and to scale the output accordingly. For an illustration of this process, see "Example 1: Scaling Data-Dependent Output" on page 893 .

When you use the DATASYS option, the GANNO procedure reads the entire input data set before drawing the graph and creates an output environment that is data dependent. That is, the environment is based on the minimum and maximum values that are contained in the data set. It then scales the data to fit this environment so that all graphics elements can be drawn.

Although the DATASYS option enables you to generate graphs using one of the data-dependent coordinate systems, it requires that the procedure scan the entire data set to determine the minimum and maximum data values. You can save this extra pass of the data set by using data-dependent values only in procedures that generate axes. Annotate coordinate system 5 (percent of the procedure output area) is recommended for use with the GANNO procedure. This coordinate system works equally well with the GSLIDE procedure if you decide to display the annotation with titles and footnotes.

Examples: GANNO Procedure

Example 1: Scaling Data-Dependent Output

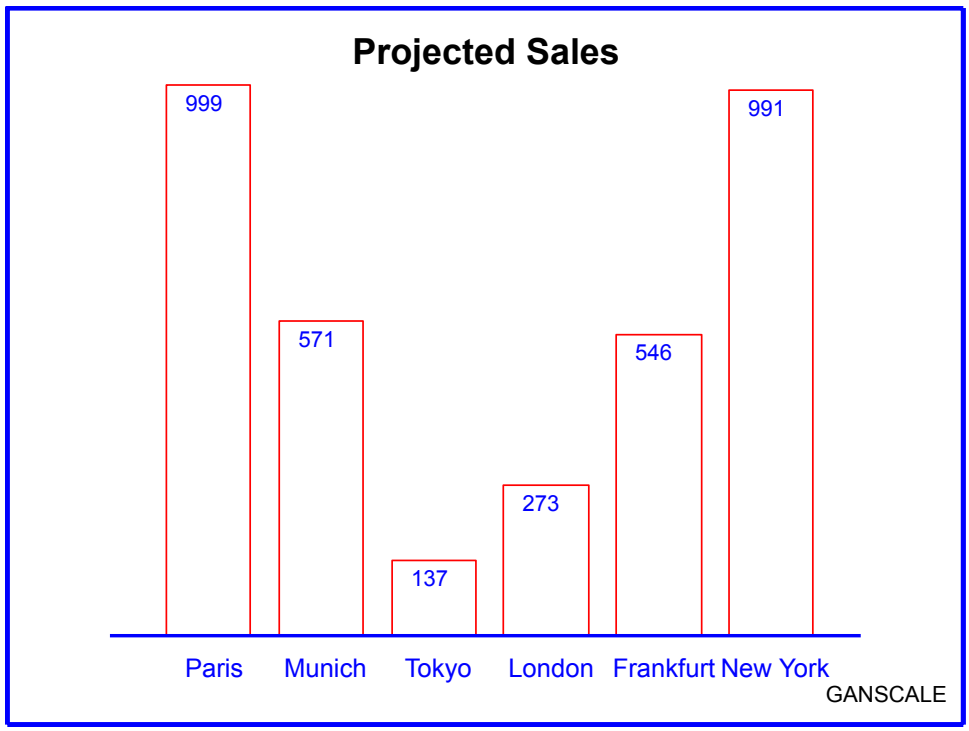
Features: PROC GANNO statement options ANNOTATE= and DATASYS

Sample library member: GANSCALE

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses an Annotate data set to scale data-dependent output with the DATASYS option and create a vertical bar chart of sales for each of six sites. The values that determine the height of each bar range from 137 to 999. The range of values is so large that the GANNO procedure cannot fit all of the bars in the output area without scaling the output. This program uses the DATASYS option to scale the data values so that the bars fit in the graphics output area.

Output 33.1 *Scaled GANNO Output*



Program

```
goptions reset=all border;

data wrldtotl;
   length sitename $ 10;
   input sitename $ 1-10 mean 12-15;
   datalines;
Paris      999
Munich     571
Tokyo      137
London     273
Frankfurt  546
New York   991
;
run;

data wrldanno;
   length function color $ 8 text $ 20;
   retain line 0 xsys ysys "2" hsys "3" x 8;
   set wrldtotl end=end;

   function="move"; x=x+8; y=20; output;
   function="bar"; y=y+(mean); x=x+9;
      style="empty"; color="red"; output;

   function="label"; y=0; x=x-4; size=3.5;
      position="E"; style="Arial";
      color="blue"; text=sitename; output;

   function="move"; y=y+(mean)-3; output;
   function="label"; x=x-1; text=left(put(mean,3.));
      position="5"; style="Arial"; size=3; output;

   if end then do;
      function="move"; x=10; y=20; output;
      function="draw"; x=90; y=20; line=1;
         size=.5; color="blue"; output;
      function="label"; x=50; y=95; text="Projected Sales";
         xsys="3"; ysys="3"; position="5"; style="Arial/bold";
         size=5; color=" "; output;
         x=92; y=5; size=3; style="Arial"; text="GANSCALE"; output;
      function="frame"; color="blue"; when="b";
         style="empty"; output;
   end;
run;

proc ganno annotate=wrldanno
   datasys;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set WRLDTOTL. WRLDTOTL contains sales data for six sites. SITENAME contains the names of the sites. MEAN contains the average sales for each site.

```
data wrldtotl;
   length sitename $ 10;
   input sitename $ 1-10 mean 12-15;
   datalines;
Paris      999
Munich     571
Tokyo      137
London     273
Frankfurt  546
New York   991
;
run;
```

Create the Annotate data set, WRLDANNO. XSYS and YSYS specify coordinate system 2 (absolute data values) for X and Y. HSYS specifies coordinate system 3 (percent of the graphics output area) for SIZE. The SET statement processes every observation in WRLDTOTL.

```
data wrldanno;
   length function color $ 8 text $ 20;
   retain line 0 xsys ysys "2" hsys "3" x 8;
   set wrldtotl end=end;
```

Draw the bars. The MOVE function defines the lower left corner of the bar. The BAR function draws the bar. Bar height (Y) is controlled by MEAN.

```
function="move"; x=x+8; y=20; output;
function="bar"; y=y+(mean); x=x+9;
   style="empty"; color="red"; output;
```

Label the bar with the name of site.

```
function="label"; y=0; x=x-4; size=3.5;
   position="E"; style="Arial";
   color="blue"; text=sitename; output;
```

Move to the top of the bar and write the value of MEAN.

```
function="move"; y=y+(mean)-3; output;
function="label"; x=x-1; text=left(put(mean,3.));
   position="5"; style="Arial"; size=3; output;
```

After all the observations are processed, add an axis line, title, footnote, and frame. The MOVE and DRAW functions draw the axis line. The LABEL function writes the title and the footnote. The FRAME function draws a border around the output.

```
if end then do;
   function="move"; x=10; y=20; output;
   function="draw"; x=90; y=20; line=1;
      size=.5; color="blue"; output;
   function="label"; x=50; y=95; text="Projected Sales";
      xsys="3"; ysys="3"; position="5"; style="Arial/bold";
      size=5; color=" "; output;
      x=92; y=5; size=3; style="Arial"; text="GANSCALE"; output;
   function="frame"; color="blue"; when="b";
      style="empty"; output;
end;
```

```
run;
```

Display the annotate graphics. The ANNOTATE= option identifies the data set that contains the graphics commands. DATASYS tells the procedure to use the maximum and minimum data values to construct the output environment. In addition, the values of X and Y are scaled to fit the environment and all of the bars are displayed on the graph.

```
proc ganno annotate=wrldanno
   datasys;
run;
quit;
```

Example 2: Storing Annotate Graphics

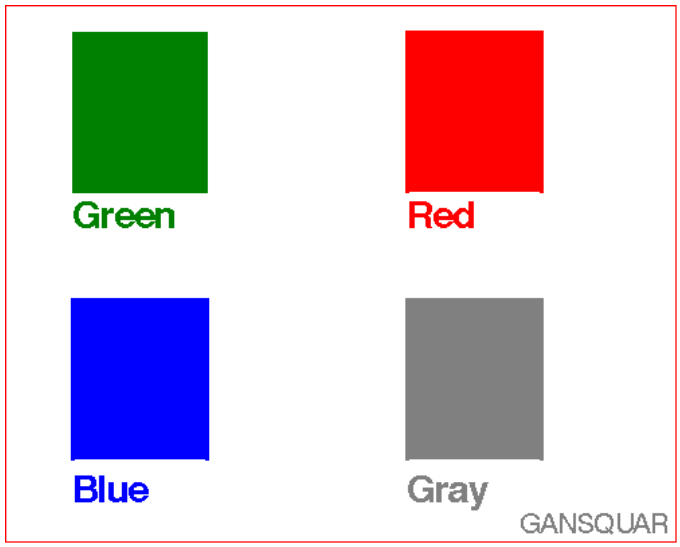
Features: PROC GANNO statement options DESCRIPTION=, GOUT=, and NAME=

Sample library member: GANSQUAR

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example creates an Annotate data set that draws four colored squares, displays the data set as a single graphics output, and stores the output as a catalog entry in a permanent catalog. In this example, the NAME= option specifies a text string that identifies the name that is stored with the graphics output in the catalog.

Output 33.2 *Four Squares*



Program

```
goptions reset=all border;

data squares;
   length function style color $ 8 text $ 15;
   xsys="3"; ysys="3";

   color="green";
   function="move"; x=10; y=65; output;
   function="bar";  x=30; y=95; style="solid"; output;

   function="label"; x=10; y=63; position="6";
      style="swissb"; size=2; text="Green"; output;

   color="red";
   function="move"; x=60; y=65; output;
   function="bar";  x=80; y=95; output;
   function="label"; x=60; y=63; position="6";
      style="swissb"; size=2; text="Red"; output;

   color="blue";
   function="move"; x=10; y=15; output;
   function="bar";  x=30; y=45; output;
   function="label"; x=10; y=12; position="6";
      style="swissb"; size=2; text="Blue"; output;

   color="gray";
   function="move"; x=60; y=15; output;
   function="bar";  x=80; y=45; output;
   function="label"; x=60; y=12; position="6";
      style="swissb"; size=2; text="Gray"; output;

   x=88; y=5; position="5"; size=1.5; style="swiss";
      text="GANSQUAR"; output;

   function="frame"; color="red"; when="b";
      style="empty"; output;
run;

proc ganno annotate=squares
   gout=excat
   name="GANSQUAR"
   description="Four squares";
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the Annotate data set, SQUARES. XSYS and YSYS specify coordinate system 3 (percent of the graphics output area) for X and Y.

```
data squares;
   length function style color $ 8 text $ 15;
   xsys="3"; ysys="3";
```

Draw the first square. The COLOR variable assigns the color for the square. The FUNCTION variable selects the operation to be performed by the Annotate facility. The X and Y variables contain coordinate values. The BAR function draws the square. When the STYLE variable is used with the BAR function, it selects the fill pattern for the bar.

```
   color="green";
   function="move"; x=10; y=65; output;
   function="bar";  x=30; y=95; style="solid"; output;
```

Label the first square. The LABEL function creates the label. The POSITION value of 6 left-justifies the text with respect to X and Y. The TEXT variable specifies the text string to be written.

```
   function="label"; x=10; y=63; position="6";
      style="swissb"; size=2; text="Green"; output;
```

Draw and label the second square.

```
   color="red";
   function="move"; x=60; y=65; output;
   function="bar";  x=80; y=95; output;
   function="label"; x=60; y=63; position="6";
      style="swissb"; size=2; text="Red"; output;
```

Draw and label the third square.

```
   color="blue";
   function="move"; x=10; y=15; output;
   function="bar";  x=30; y=45; output;
   function="label"; x=10; y=12; position="6";
      style="swissb"; size=2; text="Blue"; output;
```

Draw and label the fourth square.

```
   color="gray";
   function="move"; x=60; y=15; output;
   function="bar";  x=80; y=45; output;
   function="label"; x=60; y=12; position="6";
      style="swissb"; size=2; text="Gray"; output;
```

Add a footnote.

```
   x=88; y=5; position="5"; size=1.5; style="swiss";
      text="GANSQUAR"; output;
```

Draw a red frame.

```
   function="frame"; color="red"; when="b";
      style="empty"; output;
run;
```

Display the annotate graphics. GOUT= assigns the catalog in which the graphics output is stored. NAME= assigns a name to the entry stored in the WORK.EXCAT catalog. DESCRIPTION= assigns a description to the catalog entry.

```
proc ganno annotate=squares
   gout=excat
   name="GANSQUAR"
   description="Four squares";
run;
quit;
```

Example 3: Using the NAME= Option to Produce Multiple Graphs

Features:	PROC GANNO statement option NAME=
Sample library member:	GANMULTI
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

In this example, the GANNO procedure uses the NAME= option to generate multiple graphs from one Annotate data set. Since NAME= is assigned the variable COLOR, the GANNO procedure generates separate graphics output for each value of the COLOR, as shown in Output 33.17 on page 899 , Output 33.18 on page 900 , Output 33.19 on page 900 and Output 33.20 on page 901 .

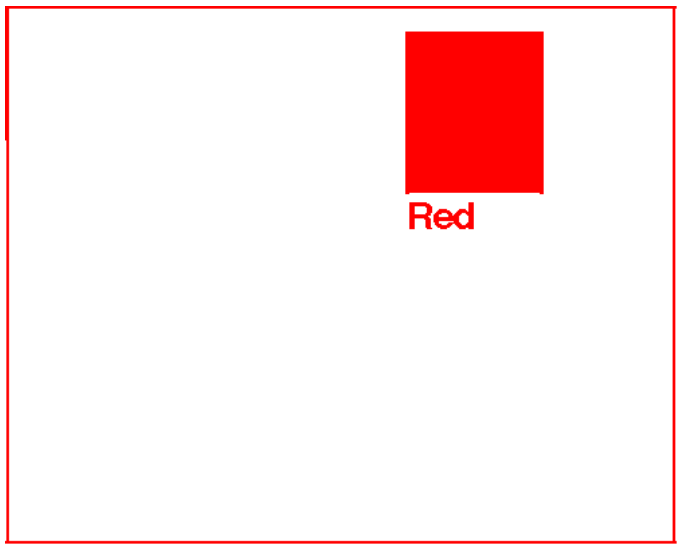
Each output is stored as a separate entry in the temporary output catalog WORK.EXCAT. The entries are named according to the values of COLOR: `BLUE`, `GRAY`, `GREEN`, and `RED`. Note that the output for `GRAY` includes the footnote shown in “Example 2: Storing Annotate Graphics” on page 896 . The output for `RED` shows the frame that is generated by the Annotate data set. The black borders in the other outputs are not generated by the code.

Output 33.3 *Output for COLOR Value BLUE (WORK.EXCAT.BLUE.GRSEG)*

Output 33.4 *Output for COLOR Value GRAY (WORK.EXCAT.GRAY.GRSEG)*

Output 33.5 *Output for COLOR Value GREEN (WORK.EXCAT.GREEN.GRSEG)*

Output 33.6 *Output for COLOR Value RED (WORK.EXCAT.RED.GRSEG)*



Program

```
goptions reset=all border;

data squares;
   length function style color $ 8 text $ 15;
   xsys="3"; ysys="3";

   color="green";
   function="move"; x=10; y=65; output;
   function="bar";  x=30; y=95; style="solid"; output;

   function="label"; x=10; y=63; position="6";
      style="swissb"; size=2; text="Green"; output;

   color="red";
   function="move"; x=60; y=65; output;
   function="bar";  x=80; y=95; output;
   function="label"; x=60; y=63; position="6";
      style="swissb"; size=2; text="Red"; output;

   color="blue";
   function="move"; x=10; y=15; output;
   function="bar";  x=30; y=45; output;
   function="label"; x=10; y=12; position="6";
      style="swissb"; size=2; text="Blue"; output;

   color="gray";
   function="move"; x=60; y=15; output;
   function="bar";  x=80; y=45; output;
   function="label"; x=60; y=12; position="6";
      style="swissb"; size=2; text="Gray"; output;

   x=88; y=5; position="5"; size=1.5; style="swiss";
      text="GANSQUAR"; output;

   function="frame"; color="red"; when="b";
      style="empty"; output;
```

```
run;

proc ganno annotate=squares
           name=color
           gout=excat
           description="Individual squares";
run;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the Annotate data set, SQUARES. XSYS and YSYS specify coordinate system 3 (percent of the graphics output area) for X and Y.

```
data squares;
   length function style color $ 8 text $ 15;
   xsys="3"; ysys="3";
```

Draw the first square. The COLOR variable assigns the color for the square. The FUNCTION variable selects the operation to be performed by the Annotate facility. The X and Y variables contain coordinate values. The BAR function draws the square. When the STYLE variable is used with the BAR function, it selects the fill pattern for the bar.

```
   color="green";
   function="move"; x=10; y=65; output;
   function="bar";  x=30; y=95; style="solid"; output;
```

Label the first square. The LABEL function creates the label. The POSITION value of 6 left-justifies the text with respect to X and Y. The TEXT variable specifies the text string to be written.

```
   function="label"; x=10; y=63; position="6";
      style="swissb"; size=2; text="Green"; output;
```

Draw and label the second square.

```
   color="red";
   function="move"; x=60; y=65; output;
   function="bar";  x=80; y=95; output;
   function="label"; x=60; y=63; position="6";
      style="swissb"; size=2; text="Red"; output;
```

Draw and label the third square.

```
   color="blue";
   function="move"; x=10; y=15; output;
   function="bar";  x=30; y=45; output;
   function="label"; x=10; y=12; position="6";
      style="swissb"; size=2; text="Blue"; output;
```

Draw and label the fourth square.

```
   color="gray";
   function="move"; x=60; y=15; output;
   function="bar";  x=80; y=45; output;
   function="label"; x=60; y=12; position="6";
      style="swissb"; size=2; text="Gray"; output;
```

Add a footnote.

```
   x=88; y=5; position="5"; size=1.5; style="swiss";
      text="GANSQUAR"; output;
```

Draw a red frame.

```
   function="frame"; color="red"; when="b";
      style="empty"; output;
run;
```

Generate the annotate graphics, separating graphs by color. NAME= identifies the variable whose values PROC GANNO uses to generate the output. GANNO produces separate output for each value of COLOR. The COLOR value is the name of the catalog entry.

```
proc ganno annotate=squares
           name=color
           gout=excat
           description="Individual squares";
run;
```

Chapter 34

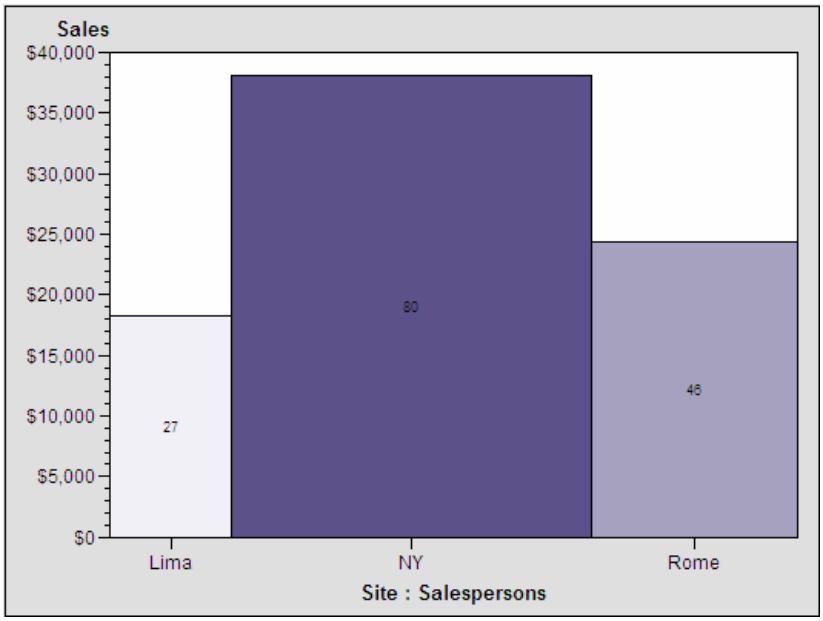
GAREABAR Procedure

Overview: GAREABAR Procedure

What Does the GAREABAR Procedure Do?

In SAS 9.4M7 and earlier releases, the GAREABAR procedure produces an area bar that displays two statistics for each category of data. For example, in the following chart, for each bar, the width, and the height of each bar represent different values, proportionally. The chart creates one bar for each unique value of the SITE variable. The height of each bar represents the SUM of the sales for that SITE. The width of each bar represents the number of salespersons generating revenue for that site.

Figure 34.1 Sample Area Bar Chart



Concepts: GAREABAR Procedure

Concepts: GAREABAR Procedure

The GAREABAR procedure produces a chart based on the values of a *chart variable*, a *width variable*, and a sum calculation variable specified by the SUMVAR= option. The chart variable can be either character or numeric. All values of the chart variable are treated as discrete. The chart values are displayed in data order. PROC GAREABAR does not calculate a midpoint.

For the VBAR statement, the width variable defines the width of the bar along the horizontal axis. The SUMVAR= variable determines the height of the bar on the vertical axis.

For the HBAR statement, the width variable defines the width of each bar on the vertical axis. The SUMVAR= variable determines the length of the bar on the horizontal axis.

The width variable, the SUMVAR= option variable, and the SUBGROUP= option variable can be calculated and displayed as a percentage of the total or as a sum. The default is sum.

An example using the SUBGROUP option is shown in “Example 4: Area Bar Chart with Subgroups; Using the RSTAT= option and the WSTAT= option to Calculate Statistics as Percentages” on page 921.

Syntax: GAREABAR Procedure

Restriction: Starting with SAS 9.4M8, because the ACTIVEX and ACTXIMG graphics devices are obsolete, the GAREABAR procedure is also obsolete. If you attempt to run the GAREABAR procedure, an error message is written to the SAS log.

Requirements: In SAS 9.4M7 and earlier releases, you must specify either the ACTIVEX or ACTXIMG device in the GOPTION statement. These devices are recommended for use with the HTML and RTF destinations only.

At least one HBAR, HBAR3D, VBAR, or VBAR3D statement is required.

Supported Statements: FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

Supports: Run-group processing

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 153.

The procedure can include the SAS/GRAPH statements BY on page 424 and NOTE on page 507, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See “SAS/GRAPH Statements” on page 397 and *SAS DATA Step Statements: Reference* for more information.

Tips: In SAS 9.4M8 and later releases, you can use the ODS Graphics SGPLOT procedure POLYGON statement to create basic area bar charts. See the post Area Bar Charts using SGPLOT on the Graphically Speaking blog.

When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, `ODS PDF; procedure-code; ODS PDF CLOSE;`). See Chapter 7, “Using Run-Group Processing,” on page 75 for more information.

When the graphics output is sent to a graphics output file, this procedure uses ODS Graphics Indexing when necessary to avoid overwriting existing output files. See “About Filename Indexing” on page 137. To reset the current index value and overwrite output files that already exist, use the following statement:

```
ods graphics / reset=index;
```

PROC GAREABAR <DATA=*input-data-set*;>

HBAR | HBAR3D | VBAR | VBAR3D *chart-variable*width-variable* / SUMVAR=*numeric-variable*<*options*>

PROC GAREABAR Statement

Identifies the data set containing the chart variables.

Requirement: An input data set is required.

Syntax

PROC GAREABAR <DATA=*input-data-set*>;

Optional Argument

PROC GAREABAR statement options affect all graphs produced by the procedure.

DATA=*input-data-set*;
specifies the SAS data set that contains the variable(s) to chart. By default the procedure uses the most recently created SAS data set.

HBAR, HBAR3D, VBAR, and VBAR3D Statement

Create horizontal or vertical bar charts in which the length or height of the bar represents the value of a chart statistic for each category of data. A second statistic is represented by the width of each bar.

Requirement: You must specify at least one chart variable, one width variable, and the SUMVAR= option.

Supported Statements: FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

Syntax

HBAR | HBAR3D | VBAR | VBAR3D *chart-variable*width-variable* /
SUMVAR=*numeric-variable* <*options*>;

Summary of Optional Arguments

Appearance options

CFRAME=*background-color*
specifies a background color for the graph.

COUTLINE=*bar-outline-color* | SAME
outlines all of the bars with the color specified.

CTEXT=*text-color*
specifies a color for all text on the chart.

FRAME

NOFRAME
specifies whether the two-dimensional axis area frame or the three-dimensional backplane is drawn.

LEGEND=LEGEND<1...99>
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.

NOLEGEND

suppresses the legend that is automatically generated by the SUBGROUP= option.

Catalog entry description options

DESCRIPTION="*description*"
: specifies the description of the plot.

NAME="*name*"
: specifies the name of the graphics output file created.

Midpoint options

CONTINUOUS
: specifies that the graph data be treated as continuous.

DISCRETE
: treats the chart variable axis data as discrete data.

SUBGROUP=*subgroup-variable*
: divides the bars into segments according to the values of the *subgroup-variable* column.

Statistic options

WIDTHSTAT= FIRST | LAST | SUM | PCT | PERCENT
: specifies the statistic to be applied to the width-variable.

Required Arguments

chart-variable
: specifies the variable that defines the categories of data to chart. This variable can be either character or numeric. Each unique value of the chart variable results in a separate bar.

width-variable
: specifies the variable that defines the width of each bar. The *width-variable* is always numeric. The width of each bar represents the sum of the *width-variable* values for that category. The default statistic is sum.

SUMVAR=*numeric-variable*
: specifies the numeric variable used for statistical calculations. The resulting statistics are represented by the height of each vertical bar or length of each horizontal bar. The default statistic is sum.

Optional Arguments

The options in an HBAR, HBAR3D, VBAR, and VBAR3D statement affect all bars that are produced by the statement. You can specify as many options as you want and list them in any order.

CFRAME=*background-color*
: specifies a background color for the graph. The specified color must be a valid SAS/GRAPH color name.

Alias	CFR=
Style reference	Color attribute of the GraphBackground element
Restriction	Not supported by Java

CONTINUOUS
specifies that the graph data be treated as continuous. Continuous data can take any of an infinite number of values between whole numbers, so it might not be measured accurately. The default is discrete.

Restriction Not supported by Java

COUTLINE=*bar-outline-color* | SAME
outlines all of the bars with the color specified. SAME specifies that the outline color of a bar is the same as the interior pattern color. The default color is the outline color of the current style.

Alias CO=

Style reference Color attribute of the GraphOutlines element

Restriction Not supported by Java

CTEXT=*text-color*
specifies a color for all text on the chart. The GAREABAR procedure looks for the text color in the following order:

1. colors specified for labels and values on assigned LEGEND statements, which override the CTEXT= option specified in the GAREABAR statement
2. the color specified by the CTEXT= option in the GAREABAR statement
3. the color specified by the CTEXT= option in a GOPTIONS statement
4. the color specified in the current style

Alias CT=

Style reference Color attribute of the GraphLabelText and GraphValueText elements

Restriction Not supported by Java

DESCRIPTION="*description*"
specifies the description of the plot. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is displayed as follows:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" on page 251 for more information
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open
- the description and the properties for the output in the Results window
- the description and properties for the catalog entry in the SAS Explorer
- the **Description** field of the PROC GREPLAY window

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on statements such as TITLE, FOOTNOTE, and NOTE. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default GAREABAR of *chart-variable*

Restriction Not supported by Java

DISCRETE

treats the chart variable axis data as discrete data. Discrete data is characterized as data in which the variable can take only one of a finite set of values. The GAREABAR procedure creates a separate bar for each unique value of the chart variable. If the chart variable has a format associated with it, each formatted value is treated as a unique value. The default is discrete.

Restriction Not supported by Java

FRAME | NOFRAME

specifies whether the two-dimensional axis area frame or the three-dimensional backplane is drawn. The default is FRAME, which draws a frame around the axis areas (in two-dimensional bar charts) or generates a colored three-dimensional backplane (in three-dimensional bar charts). For three-dimensional charts, NOFRAME removes the backplane color, and leaves the backplane grid, the vertical axis and plane, and the horizontal axis and plane.

The NOFRAME option overrides the CFRAME= option.

Alias FR | NOFR

Restriction Not supported by Java

LEGEND=LEGEND<1...99>

assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option. The LEGEND= option itself does not generate a legend.

LEGEND= is ignored if any of the following are true:

- The SUBGROUP= option is not used.
- The specified LEGEND definition is not in effect.
- The NOLEGEND option is used.

Restrictions The LEGEND statement options are partially supported by ActiveX.

Not supported by Java

See “LEGEND Statement” on page 432

NAME="*name*"

specifies the name of the graphics output file created.

The following applies to *name*:

- **For the graphics output file name:**
  - All characters are represented in lowercase.
  - The ACTXIMG device must be used in order to generate a graphics output file.
  - The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

- The name can include special characters.
- Each time the graph is generated in a SAS session, SAS/GRAPH adds a number to the name, or increments the last number used to create a unique file name for the output. See "About Filename Indexing" on page 137.

Default graph.png

Restriction Not supported by Java

NOLEGEND
suppresses the legend that is automatically generated by the SUBGROUP= option. The NOLEGEND option is ignored if the SUBGROUP= option is not used.

Restriction Not supported by Java

SUBGROUP=*subgroup-variable*
divides the bars into segments according to the values of the *subgroup-variable* column. The *subgroup-variable* column can be either character or numeric, and is always treated as a discrete variable. The SUBGROUP= option creates a separate segment within each bar for each unique value of the subgroup variable.

Restriction Not supported by Java

WIDTHSTAT= FIRST | LAST | SUM | PCT | PERCENT
specifies the statistic to be applied to the width-variable. Values for WIDTHSTAT= are as follows:

FIRST	specifies that the width of each bar is determined solely by the value of the first observation of the *width-variable* in the data for that bar.
LAST	specifies that the width of each bar is determined solely by the value of the last observation of the *width-variable* in the data for that bar.
SUM	specifies that the width of each bar is the sum of the values of the *width-variable* for that bar.
PCT | PERCENT	specifies that the width of each bar is the sum of the values of the *width-variable* for that bar. The width is expressed as a percentage of the sum of the values of the *width-variable* for all of the bars.

Alias WSTAT=

Default SUM

Restriction Not supported by Java

Details

The HBAR, HBAR3D, VBAR, and VBAR3D statements specify the variables that define the categories, and width of each bar. The SUMVAR= option variable calculates the length or height of each bar. These statements do the following;

- calculate the chart statistic for each bar (the default is SUM)
- scale the response axes and the bars according to the statistic value
- calculate the width of each bar, based on the value of the width variable
- draw a frame around the axis area using a color determined by the current style

You can use statement options to change the type of chart, to display specific statistics, and to modify the appearance of the chart. You can also specify an additional variable to subgroup your data, which divides the bars into segments and displays a legend to identify the segments.

In addition, you can make the following changes with global statements:

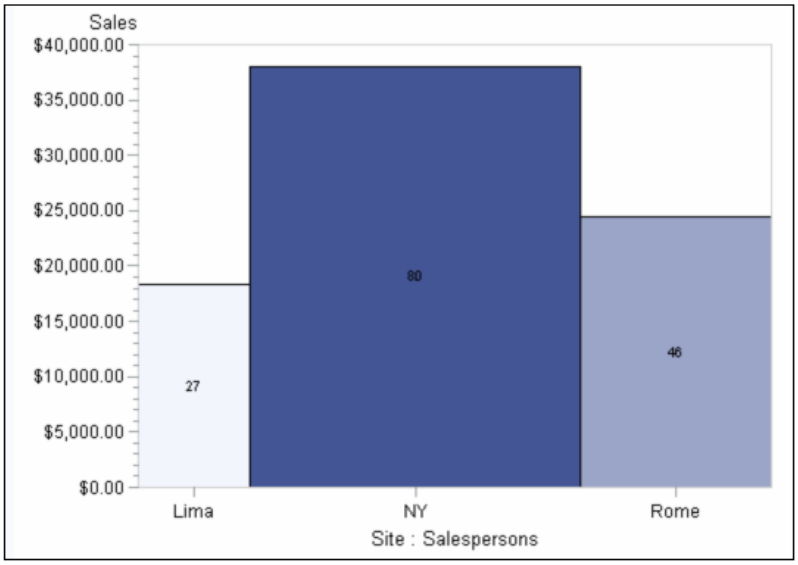
- use the LEGEND statement to modify the legend
- use the TITLE and FOOTNOTE statements to add titles and footnotes to the chart
- use the PATTERN statement to create PATTERN definitions that define the color and type of area fill for patterns used in graphs.

Examples: GAREABAR Procedure

Example 1: Generating an Area Bar Chart

Features:	VBAR Statement
Restriction:	This example is valid only in SAS 9.4M7 and earlier releases. It does not run in SAS 9.4M8 and later releases.
Sample library member:	GABSUMVR
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This area bar chart reveals three geographic sites (Lima, NY, Rome) along the horizontal axis. The width of each bar represents the sum of the salespersons assigned to each site. The height of each bar represents the sum of the sales for each site. The chart shows that NY had the greatest sales, as well as the greatest number of salespersons.



Program

```
filename odsout ".";

data totals;
 input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima   1 4043.97   4
NY      1 8225.26  12
Rome 1 3543.97    6
Lima  2 3723.44    5
NY     2 8595.07   18
Rome  2 5558.29   10
Lima  3 4437.96    8
NY     3 9847.91   24
Rome 3 6789.85   14
Lima 4 6065.57   10
NY     4 11388.51  26
Rome 4 8509.08   16
;

ods _all_ close;

goptions reset=all dev=actximg;

ods html path=odsout file="areabarchartsum.html";

proc gareabar data=totals;
   vbar site*salespersons /
      sumvar=sales;
run;
quit;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Create a file reference for the ODS output The current working directory is specified in this example.

```
filename odsout ".";
```

Create the WORK.TOTALS data set.

```
data totals;
 input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima   1 4043.97   4
NY      1 8225.26  12
Rome 1 3543.97    6
Lima  2 3723.44    5
NY     2 8595.07   18
Rome  2 5558.29   10
Lima  3 4437.96    8
NY     3 9847.91   24
Rome 3 6789.85   14
Lima 4 6065.57   10
NY     4 11388.51  26
Rome 4 8509.08   16
;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Set the graphics environment. Specify the ACTXIMG device.

```
goptions reset=all dev=actximg;
```

Open the ODS HTML destination. The PATH= option specifies the file reference for the ODS output that was created previously. The FILE= option specifies a name for the HTML output file.

```
ods html path=odsout file="areabarchartsum.html";
```

Run PROC GAREABAR with VBAR statement. The VBAR statement creates a vertical bar for each value of SITE. The SALESPERSONS variable sets the width of the bars. The SUMVAR=SALES option controls the height of each of the bars.

```
proc gareabar data=totals;
   vbar site*salespersons /
      sumvar=sales;
run;
quit;
```

Close ODS HTML.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs (not required in SAS Studio).

```
ods html; /* Not required in SAS Studio */
```

Example 2: Generating an Area Bar Chart with a Numeric Chart Variable

Features: VBAR statement options
SUMVAR=
WSTAT=

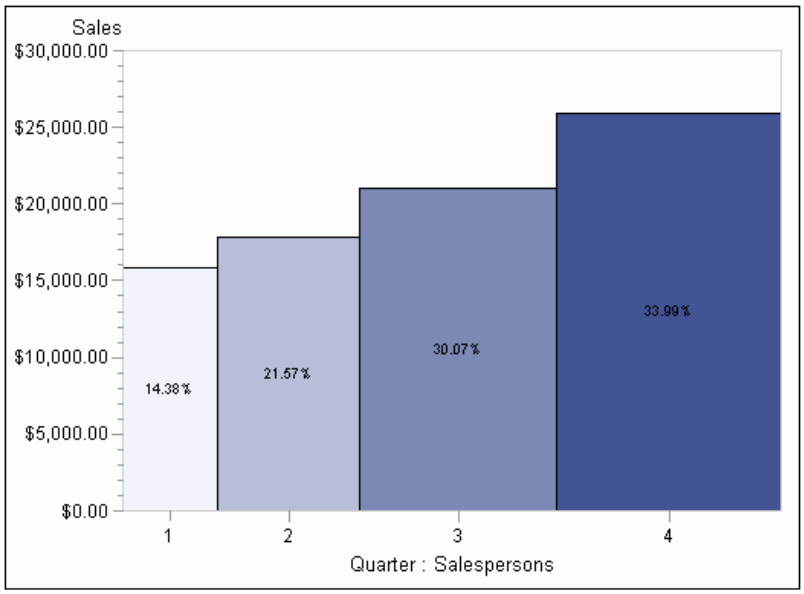
Restriction: This example is valid only in SAS 9.4M7 and earlier releases. It does not run in SAS 9.4M8 and later releases.

Sample library member: GABNUMVR

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This chart displays a numeric chart variable, QUARTER, representing the four quarters of an unspecified year. The GAREABAR procedure treats all values of a numeric chart variables as discrete, unless the CONTINUOUS option is used. GAREABAR does not calculate midpoints.

The total sales for each quarter of the year are represented by the height of each bar along the vertical axis. The width of each bar along the horizontal axis indicates the percentage of salespersons during each quarter. The chart shows the correlation between the number of salespersons, and the total sales.



Program

```
filename odsout ".";

data totals;
 input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima  1 4043.97    4
NY     1  8225.26   12
Rome  1 3543.97    6
Lima  2  3723.44    5
NY      2  8595.07   18
Rome  2 5558.29   10
Lima  3 4437.96    8
NY     3  9847.91   24
Rome  3 6789.85   14
Lima  4  6065.57   10
NY     4  11388.51  26
Rome  4  8509.08   16
;

ods _all_ close;

goptions reset=all dev=actximg;

ods html path=odsout file="areabarchartnum.html";

proc gareabar data=totals;
   vbar quarter*salespersons/
      sumvar=sales
      wstat=pct;
run;
quit;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Create a file reference for the ODS output The current working directory is specified in this example.

```
filename odsout ".";
```

Create the WORK.TOTALS data set.

```
data totals;
 input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima  1 4043.97    4
NY     1  8225.26   12
Rome  1 3543.97    6
Lima  2  3723.44    5
NY      2  8595.07   18
Rome  2 5558.29   10
```

```
Lima  3 4437.96    8
NY     3  9847.91   24
Rome  3 6789.85   14
Lima  4  6065.57   10
NY     4  11388.51  26
Rome  4  8509.08   16
;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Set the graphics environment. Specify the ACTXIMG device.

```
goptions reset=all dev=actximg;
```

Open the ODS HTML destination. The PATH= option specifies the file reference for the ODS output that was created previously. The FILE= option specifies a name for the HTML output file.

```
ods html path=odsout file="areabarchartnum.html";
```

Run PROC GAREABAR with VBAR statement. The VBAR statement creates a vertical bar for each value of QUARTER. The SALESPERSONS variable sets the width of the bars. The SUMVAR=SALES option controls the height of each of the bars. The WSTAT=PCT option displays the width statistic as a percentage (the number of salespersons as a percentage of the whole).

```
proc gareabar data=totals;
   vbar quarter*salespersons/
      sumvar=sales
      wstat=pct;
run;
quit;
```

Close ODS HTML.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs (not required in SAS Studio).

```
ods html; /* Not required in SAS Studio */
```

Example 3: Generating an Area Bar Chart with Subgroups

Features: HBAR statement options
- SUBGROUP=
- SUMVAR=
- RSTAT=
- WSTAT=

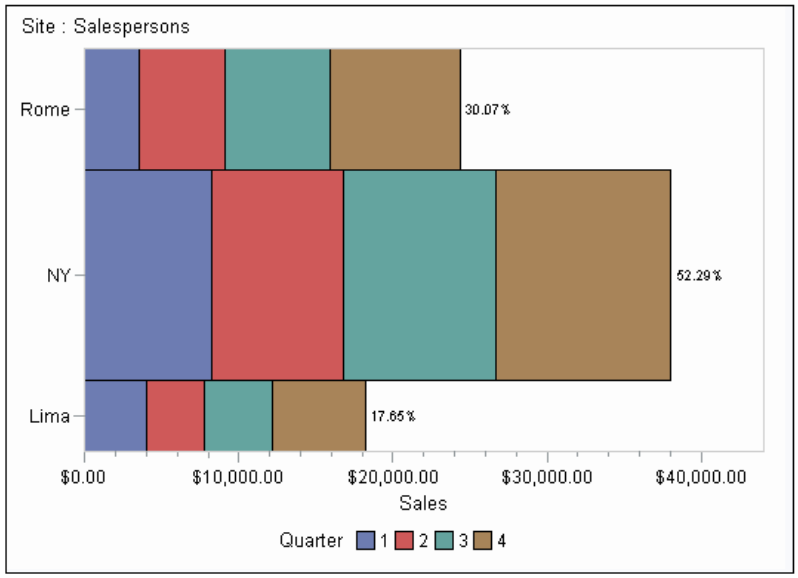
Restriction: This example is valid only in SAS 9.4M7 and earlier releases. It does not run in SAS 9.4M8 and later releases.

Sample library member: GABSUBGR

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses the SUBGROUP= option to display the same statistics as displayed by Examples 1 and 2. Similar to Example 1, this example shows the total sales for each of the three geographic sites. The relative thickness of each bar represents the number of salespersons at each site.

The addition of subgroups to this chart shows the relative percentage of sales for each quarter. This chart demonstrates that all of the sites had most of their sales posted in the fourth quarter.



Program

```
filename odsout ".";

data totals;
 input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima 1 4043.97  4
NY    1 8225.26 12
Rome  1 3543.97  6
Lima  2 3723.44  5
NY     2 8595.07 18
Rome  2 5558.29 10
Lima  3 4437.96  8
NY      3 9847.91 24
Rome  3 6789.85 14
Lima  4 6065.57 10
NY     4 11388.51 26
Rome 4  8509.08 16
```

```
;

ods _all_ close;

goptions reset=all dev=actximg;

ods html path=odsout file="areabarchartsub.html";

proc gareabar data=totals;
   hbar site*salespersons /
      sumvar=sales
      subgroup=quarter
      wstat=PCT;
run;
quit;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Create a file reference for the ODS output The current working directory is specified in this example.

```
filename odsout ".";
```

Create the WORK.TOTALS data set.

```
data totals;
 input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima 1 4043.97  4
NY    1 8225.26 12
Rome  1 3543.97  6
Lima  2 3723.44  5
NY     2 8595.07 18
Rome  2 5558.29 10
Lima  3 4437.96  8
NY      3 9847.91 24
Rome  3 6789.85 14
Lima  4 6065.57 10
NY     4 11388.51 26
Rome 4  8509.08 16
;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Set the graphics environment. Specify the ACTXIMG device.

```
goptions reset=all dev=actximg;
```

Open the ODS HTML destination. The PATH= option specifies the file reference for the ODS output that was created previously. The FILE= option specifies a name for the HTML output file.

```
ods html path=odsout file="areabarchartsub.html";
```

Run PROC GAREABAR with an HBAR statement. The HBAR statement creates a horizontal bar for each value of SITE. The SALESPERSONS variable sets the width of each bar. The SUMVAR=SALES option sets the length of each bar. The WSTAT=PERCENT option displays the width statistic as a percentage, which enables you to compare the distribution of salespersons for each quarter. The SUBGROUP=QUARTER option and the RSTAT=SUM option are reflected in the statistics that are displayed as absolute numbers along the horizontal bar.

```
proc gareabar data=totals;
   hbar site*salespersons /
      sumvar=sales
      subgroup=quarter
      wstat=PCT;
run;
quit;
```

Close ODS HTML.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs (not required in SAS Studio).

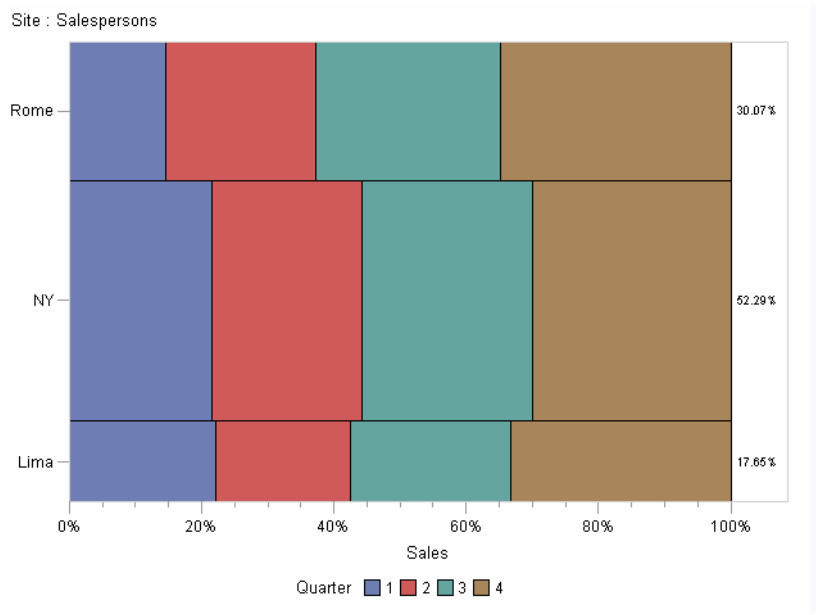
```
ods html; /* Not required in SAS Studio */
```

Example 4: Area Bar Chart with Subgroups; Using the RSTAT= option and the WSTAT= option to Calculate Statistics as Percentages

Features:	HBAR statement options SUBGROUP= RSTAT= WSTAT=
Restriction:	This example is valid only in SAS 9.4M7 and earlier releases. It does not run in SAS 9.4M8 and later releases.
Sample library member:	GABWSTAT
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses the RSTAT= option and the WSTAT= option to calculate percentages for the length variable (sumvar) and the width variable (chart variable). The SUBGROUP= option subgroups each bar by quarter.

When the SUBGROUP= option is specified, you can use the RSTAT= option to specify whether the SUMVAR= option variable is to be calculated as a percentage or as a sum.



Program

```
filename odsout ".";

data totals;
input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima    1   4043.97    4
NY       1   8225.26   12
Rome   1   3543.97    6
Lima    2   3723.44    5
NY     2    8595.07   18
Rome  2   5558.29   10
Lima    3   4437.96    8
NY       3   9847.91   24
Rome   3   6789.85   14
Lima    4   6065.57   10
NY       4 11388.51   26
Rome   4   8509.08   16
;

ods _all_ close;

goptions reset=all dev=actximg;

ods html path=odsout file="areabarchartwstat.html";

proc gareabar data=totals;
   hbar site*salespersons /
      sumvar=sales
      subgroup=quarter
      rstat=PCT
      wstat=PCT;
run;
quit;
```

```
ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Create a file reference for the ODS output The current working directory is specified in this example.

```
filename odsout ".";
```

Create the WORK.TOTALS data set.

```
data totals;
input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima    1   4043.97    4
NY       1   8225.26   12
Rome   1   3543.97    6
Lima    2   3723.44    5
NY     2    8595.07   18
Rome  2   5558.29   10
Lima    3   4437.96    8
NY       3   9847.91   24
Rome   3   6789.85   14
Lima    4   6065.57   10
NY       4 11388.51   26
Rome   4   8509.08   16
;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Set the graphics environment. Specify the ACTXIMG device.

```
goptions reset=all dev=actximg;
```

Open the ODS HTML destination. The PATH= option specifies the file reference for the ODS output that was created previously. The FILE= option specifies a name for the HTML output file.

```
ods html path=odsout file="areabarchartwstat.html";
```

Run PROC GAREABAR with an HBAR statement. The HBAR statement creates a horizontal bar for each value of SITE. The SALESPERSONS variable sets the width of each bar. The SUMVAR=SALES option sets the length of each bar. The WSTAT=PERCENT option displays the width statistic as a percentage. (The percentage of salespersons is shown by the relative thickness of each bar along the vertical axis.) The SUBGROUP=QUARTER option and the RSTAT=PCT option, request that the sales for each quarter are displayed as a percentage on the horizontal axis.

```
proc gareabar data=totals;
   hbar site*salespersons /
      sumvar=sales
      subgroup=quarter
      rstat=PCT
      wstat=PCT;
```

```
run;
quit;
```

Close ODS HTML.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs (not required in SAS Studio).

```
ods html; /* Not required in SAS Studio */
```

Chapter 35

GBARLINE Procedure

Overview: GBARLINE Procedure

About the GBARLINE Procedure

The GBARLINE procedure produces bar-line charts. Bar-line charts are vertical bar charts with one or more plot overlays. These charts graphically represent the value

of a statistic calculated for one or more variables in an input SAS data set. The charted variables can be either numeric or character.

The procedure calculates these statistics:

- sum
- mean
- frequency or cumulative frequency
- percentage or cumulative percentage

Use the GBARLINE procedure to do the following tasks:

- display and compare exact and relative magnitudes
- examine the contribution of parts to the whole
- analyze where data are out of balance
- display a long series of data, and show trends and patterns

In conjunction with the SYMBOL statement, the GBARLINE procedure can produce needle plot overlays, and overlay plots with stepped interpolation.

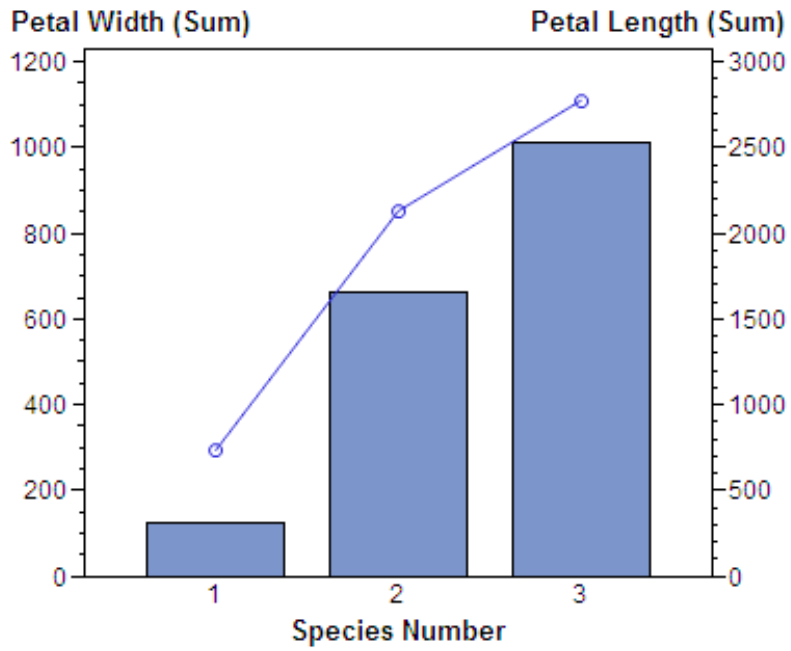
Note: PROC GBARLINE is not supported by Java.

Chart Terminology

Bar-line charts display the magnitude of data with bars, each of which represents a category of data (midpoint). The height of the bars represents the value of the bar statistic for the corresponding midpoint.

Figure 35.105 on page 927 shows the relationship between petal width and petal length for three species of flowers. The horizontal axis is the midpoint axis and the vertical axes are response axes. The right response axis is the PLOT statement axis and the left vertical axis is the BAR statement axis. Each axis is labeled with the variable name or label. Each species is a midpoint, so each bar is labeled with the species identifier.

Figure 35.1 Bar-Line Chart



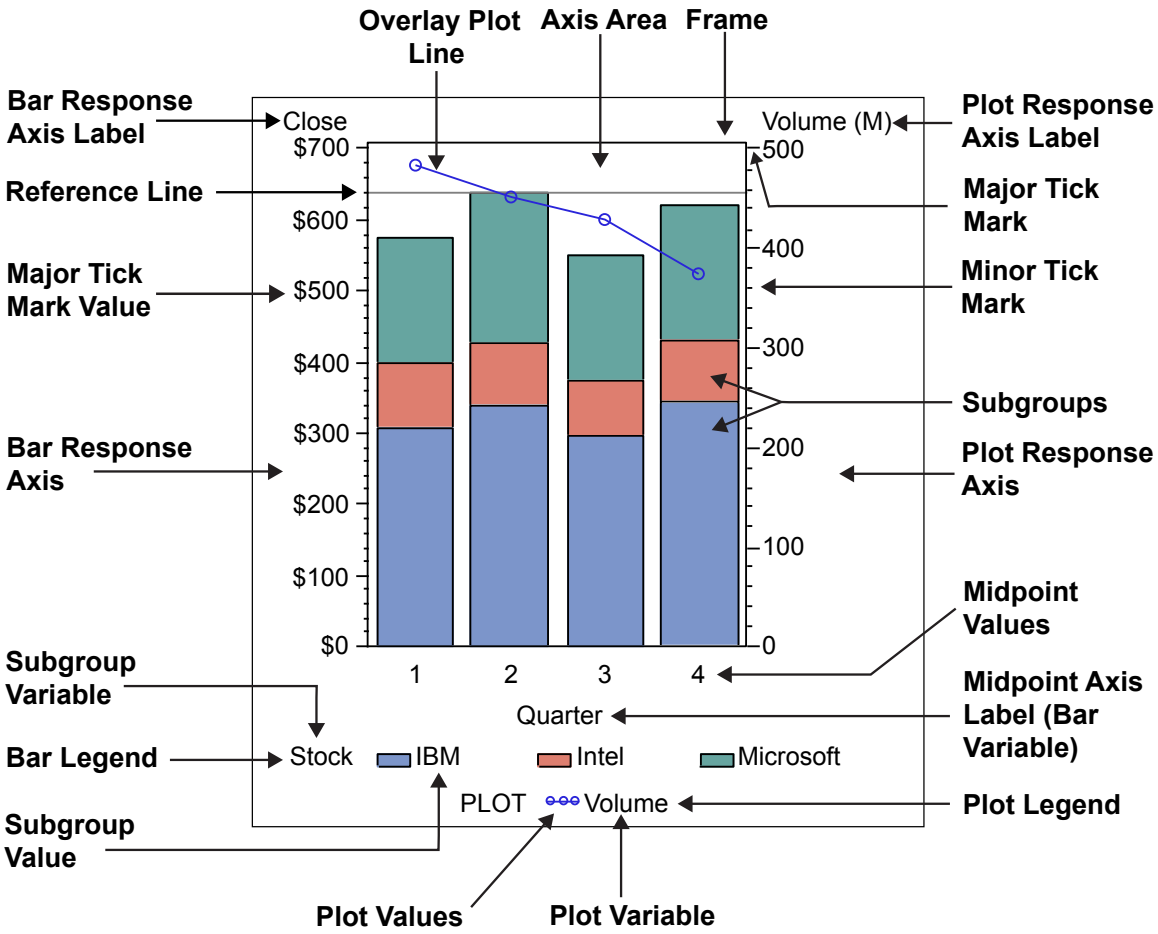
Concepts: GBARLINE Procedure

About Bar-Line Charts

The GBARLINE procedure produces a bar chart based on the values of a chart variable and an optional response variable (SUMVAR= option). The computed statistic can be set with the TYPE= option. Each line plot uses the same chart variable and has an optional response variable (SUMVAR= option). A computed statistic can be set with the TYPE= option.

Figure 35.106 on page 928 illustrates the parts of a bar-line chart.

Figure 35.2 Parts of a Bar-Line Chart



Bar-line charts have three axes:

- a midpoint axis that shows the categories of data, based on the chart variable
- a left response axis that displays the scale of values for the bar statistic (based on the response variable, if specified)
- a right response axis that displays the scale of values for the line statistic (based on the response variable, if specified)

The response axes are divided into evenly spaced intervals identified with major tick marks that are labeled with the corresponding statistic value. Minor tick marks are evenly distributed between the major tick marks. Each axis is labeled with the variable name or label. The right response axis is scaled to accommodate all the line variable response values when multiple PLOT statements are present.

About the Chart Variable

The *chart variable* is the variable in the input data set whose value determines the categories of data represented by the bar and lines. The chart variable generates the midpoints to which each observation in the data set contributes.

A character chart variable is always discrete.

About Midpoints

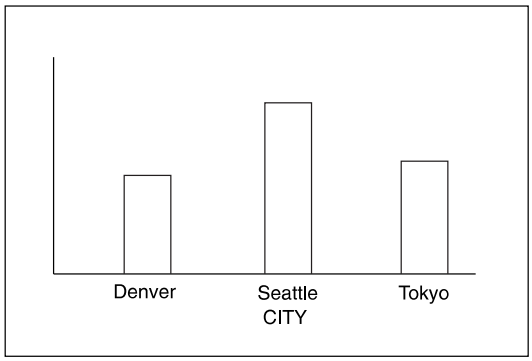
Definition: Midpoints

Midpoints are the values of the chart variable that identify categories of data. By default, midpoints are selected or calculated by the procedure. The way the procedure handles the midpoints depends on whether the values of the chart variable are character, discrete numeric, or continuous numeric.

Character Values

A character chart variable generates a midpoint for each unique value of the variable. In the following example, the chart variable CITY contains the names of three different cities, and each city is a midpoint, resulting in three midpoints for the chart:

Figure 35.3 *Character Midpoints*



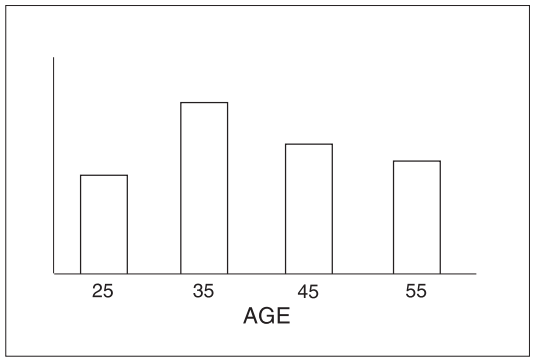
By default, character midpoints are arranged in alphabetic order. If a character variable has an associated format, then the values are arranged in order of the formatted values.

Continuous Numeric Values

A continuous numeric variable generates midpoints that represent ranges of values. By default, the GBARLINE procedure determines the number of uniform ranges (LEVELS), calculates the number of observations in each range, and then computes the TYPE= statistic based on this frequency. A value that falls exactly on a range boundary is placed in the higher range.

In the following example, the numeric variable AGE has been divided into five equal levels that span the data range. The horizontal axis tick values are at the midpoint of each level.

Figure 35.4 *Continuous Numeric Midpoints*



By default, midpoints of ranges are arranged in ascending order.

Selecting and Ordering Midpoints

For character or discrete numeric values, you can use the MIDPOINTS= option to rearrange the midpoints or to exclude midpoints from the chart. For example, to change the default alphabetic order of the midpoints in Figure 35.107 on page 929, specify the following midpoints:

```
midpoints="Tokyo" "Denver" "Seattle"
```

To exclude the midpoint for Denver, specify the following midpoints:

```
midpoints="Tokyo" "Seattle"
```

In this case, values excluded by the option are not included in the calculation of the chart statistic.

You can order or select discrete numeric midpoint values just as you do character values, but you omit the quotation marks when specifying numeric values.

For continuous numeric variables, use the LEVELS= or MIDPOINTS= option to change the number of midpoints. These options can also control the range of values each midpoint represents or change the order of the midpoints. To control the range of values each midpoint represents, use the MIDPOINTS= option to specify the midpoint value of each range. For example, to select the ranges 20–29, 30–39, and 40–49, specify the following values:

```
midpoints=25 35 45;
```

Alternatively, to select the number of midpoints that you want and let the procedure calculate the ranges and midpoints, use the LEVELS= option.

You can also use formats to control the ranges of continuous numeric variables, but in that case the values are no longer continuous but become discrete.

Note: You cannot use the MIDPOINTS= option to exclude continuous numeric values from the chart. This is because values below or above the ranges specified by the option are automatically included in the first and last midpoints. To exclude

continuous numeric values from a chart, use a WHERE statement in a DATA step or the WHERE= data set option.

See also the descriptions of the options "LEVELS=*number-of-midpoints* | ALL" on page 951 and "MIDPOINTS=*value-list*" on page 952.

About Response Variables

Response variables can be specified for either the bar chart or any line plot with the SUMVAR= option.

For example:

```
BAR age  / DISCRETE SUMVAR=weight;  PLOT / SUMVAR=height;
```

When you specify a response variable, the only statistics available are SUM or MEAN. SUM is the default. To change the statistic, you specify the TYPE= option (for example, TYPE=MEAN).

If you do not specify a response variable, a summary statistic for the chart variable is computed. By default it is FREQ (frequency). You can use the TYPE= option to indicate another statistic: PERCENT, CFREQ (cumulative frequency) or CPERCENT (cumulative percent).

For more information about these statistics, see "About Chart Statistics" on page 931. See also the PLOT statement option descriptions of "SUMVAR=*plot-variable*" on page 971 and "TYPE=*statistic*" on page 971.

About Chart Statistics

Definition: Chart Statistics

The *chart statistics* are the statistical values calculated for the chart variable or the response variable. When there is no response variable, the GBARLINE procedure calculates one of four possible statistics with the default being FREQ. When there is a response variable, one of two possible statistics is computed with the default being SUM. You can specify the chart statistic with the TYPE= option for both the bar chart and any line plot. For the bar chart, the default statistic is frequency. For the plot variable, the default statistic is sum.

The examples given in the descriptions of these statistics in the following topics assume a data set with two variables, CITY and SALES. The values of CITY are **Denver**, **Seattle**, and **Tokyo**. There are 21 observations: seven for Denver, nine for Seattle, and five for Tokyo.

Frequency

The frequency statistic is the total number of observations in the data set for each midpoint. For example, seven observations of the bar variable, CITY, contain the value **Denver**, so the frequency for the **Denver** midpoint is 7.

Cumulative Frequency

The cumulative frequency statistic adds the frequency for the current midpoint to the frequency of all of the preceding midpoints. For example, the frequency for the **Denver** midpoint is 7, and the frequency for the next midpoint, **Seattle**, is 9. Therefore, the cumulative frequency for **Seattle** is 16 and the cumulative frequency for Tokyo is 21.

Percentage

The percentage statistic is calculated by dividing the frequency for each midpoint by the total frequency count for all midpoints in the chart or group and multiplying it by 100. For example, the frequency count for the **Denver** midpoint is 7 and the total frequency count for the chart is 21, so the percentage statistic for **Denver** is 33.3%.

Cumulative Percentage

The cumulative percentage statistic adds the percentage for the current midpoint to the percentage for all of the preceding midpoints in the chart or group. For example, the percentage for the **Denver** midpoint is 33.3, and the percentage for the next midpoint, **Seattle**, is 42.9, so the cumulative percentage for **Seattle** is 76.2.

Sum

The sum statistic is the total of the values, for each midpoint, for the variable specified by the SUMVAR= option. For example, if you specify SUMVAR=SALES and the values of the SALES variable for the seven **Denver** observations are **8734**, **982**, **1504**, **3207**, **4502**, **624**, and **918**, the sum statistic for the **Denver** midpoint is 20,471.

You must use the SUMVAR= option to specify the variable for which you want the sum statistic.

Mean

The mean statistic is the average of the values, for each midpoint, for the variable specified by the SUMVAR= option. For example, if TYPE=MEAN and SUMVAR=SALES, the mean statistic for the `Denver` midpoint is 2924.42.

You must use the SUMVAR= option to specify the variable for which you want the mean statistic.

Calculating Weighted Statistics

By default, each observation is counted only once in the calculation of a chart statistic. To calculate weighted statistics in which an observation can be counted more than once, use the FREQ= option. This option identifies a variable whose values are used as a multiplier for the observation in the calculation of the statistic. If the value of the FREQ= variable is missing, zero, or negative, then the observation is excluded from the calculation.

If you use the SUMVAR= option, then the SUMVAR= variable value for an observation is multiplied by the FREQ= variable value for the observation. The product of this calculation determines the chart statistic.

For example, to use a variable called COUNT to produce weighted statistics, assign FREQ=COUNT. If you also assign the variable HEIGHT to the SUMVAR= option, then the following table shows how the values of COUNT and HEIGHT would affect the statistic calculation:

Table 35.1 *Example of Weighted Statistics*

Value of COUNT	Value of HEIGHT	Number of times the observation is used	Value used for HEIGHT
1	55	1	55
5	65	5	325
.	63	0	-
-3	60	0	-

By default, the percentage and cumulative percentage statistics are calculated based on the frequency. Suppose you want to graph a percentage or cumulative percentage based on a sum. You can use the FREQ= option to specify a variable to use for the sum calculation and then specify PCT as the statistic, as shown in this example:

```
freq=count type=pct;
```

Because the variable that is specified by the FREQ= option determines the number of times an observation is counted, the value of COUNT is the equivalent of the sum statistic.

See also the descriptions of the options "TYPE=*statistic*" on page 958, "SUMVAR=*numeric-variable*" on page 958, and "FREQ=*numeric-variable*" on page 949.

Note: The FREQ= option is not supported by ActiveX or Java.

Missing Values

By default, the GBARLINE procedure ignores missing midpoint values for the chart variable. If you specify the MISSING option, then missing values are treated as a valid midpoint and are included on the axis. Missing values for the subgroup variables are always treated as valid subgroups.

When the value of the variable that is specified in the FREQ= option is missing, zero, or negative, the observation is excluded from the calculation of the chart statistic.

When the value of the variable specified in the SUMVAR= option is missing, the observation is excluded from the calculation of the chart statistic.

If all of the values for a response variable are missing for the bar chart, a midpoint is drawn, but no bar appears above it. For a line plot, no marker is drawn and the line connects the adjacent markers.

Plot Variable Values Out of Range

Exclude data values from a plot overlay by restricting the range of axis values with the RAXIS= options or with the ORDER= option in an AXIS statement. When an observation contains a value outside of the specified axis range, the GBARLINE procedure excludes the observation from the plot and issues a message to the log.

If you specify interpolation with a SYMBOL definition, then the values outside the axis range are excluded from interpolation calculations by default. As a result, this can change interpolated values for the plot overlay.

To specify that values outside of the axis range are included in the interpolation calculations, use the MODE= option in a SYMBOL statement. When MODE=INCLUDE, values that fall outside of the axis range are included in interpolation calculations but are excluded from the plot. The default (MODE=EXCLUDE) omits observations that are outside of the axis range from interpolation calculations. See the "SYMBOL Statement" on page 470 for details.

Controlling Patterns, Outlines, Colors, and Images

How to Control Patterns, Outlines, Colors, and Images

Default patterns, colors, outlines, and, in some cases, images, are defined by the current style, whether that style is the default GSTYLE or one that you specify with the ODS statement. You can turn off styles by specifying the NOGSTYLE system option. Alternatively, you can override individual aspects of a graph's appearance by specifying PATTERN statements, SYMBOL statements, graphics options, and procedure options.

The following sections summarize pattern behavior for the GBARLINE procedure. For more information, see the "PATTERN Statement" on page 455 and the "SYMBOL Statement" on page 470.

Default Patterns, Symbols, Lines, Colors, and Outlines

The default pattern that the GBARLINE procedure uses is a solid fill. The default colors are determined by the current style and the device.

By default, the system option GSTYLE is in effect. As a result, the procedure generates output using the style's default bar fill colors, plot line colors, widths, symbols, patterns, and outline colors. Specifically, the GBARLINE procedure uses the default values when you do not specify any of the following:

- *any* PATTERN statements
- the CPATTERNS= graphics options
- the COLORS= graphics options
- the COUTLINE= option in the BAR statement
- any SYMBOLS statements

If you do not specify any of these statements or options, then the GBARLINE procedure performs the following operations:

- selects the first default fill pattern, which is always solid, and rotates it through the list of colors available in the current style, generating one solid pattern for each color. When the solid patterns are exhausted, the procedure selects the next default subgroup bar pattern (empty) and rotates it through the appropriate set of colors. It continues in this fashion until all of the required patterns have been assigned.

  If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color

list with the COLORS= graphics option, then the procedure uses all the colors in the list to generate the patterns.

- uses the style's outline color to outline every patterned area.
- uses the style's default symbol for the initial PLOT statement points. The procedure uses the second default symbol for the next PLOT statement, and the third default symbol for the next PLOT statement. The procedure continues orderly through the set of symbols belonging to that style until all the PLOT statements have been satisfied.
- connects all the plot symbols with a solid line.

If you specify the NOGSTYLE system option, the fill pattern is solid and the color comes from the device's color list. The GBARLINE procedure uses a solid fill for the bars that it rotates once through the device's default color list, skipping the foreground color. (Typically, the foreground color is the first color in the device's color list.) Suppose no SYMBOL or PATTERN statements are in effect and the COLORS= option is not used in the GOPTIONS statement. In this case the plot line colors begin with the next color from the same color list used to color the bars. By doing this, the procedure prevents the plot line from being the same color as a bar fill. Specifically, GBARLINE performs the following operations:

- selects the first default fill, which is always solid, and rotates it through the color list, generating one solid pattern for each color. If the first color in the device's color list is black (or white), the procedure skips that color and begins generating patterns with the next color.
- uses the foreground color to outline every patterned area.
- selects the next default pattern fill (if it needs additional patterns), and rotates that pattern through the color list, skipping the foreground color as before. The procedure continues in this fashion until it has generated enough patterns for the chart.
- uses the device's default color to outline every patterned area.
- selects the next color in the list after the last bar color and uses it to draw the first PLOT statement symbol and connecting line.
- rotates through the color list for any subsequent PLOT statements.

If the procedure needs additional patterns, PROC GBARLINE selects the next default pattern fill (empty) and rotates it through the color list, skipping the foreground color as before. The procedure continues in this fashion until it has generated enough patterns for the chart.

You might change or override the default behavior for patterns if you specify a color list with the COLORS= option in a GOPTIONS statement. For example, if the COLORS= list contains more than one color, then the procedure rotates the default solid pattern through that list. It uses every color, even if the foreground color is black (or white). The default outline color remains the foreground color or the color specified by the current style.

For a description of the COLORS= or other graphics options, see “Graphics Options and Device Parameters Dictionary” on page 577.

User-Defined Patterns, Colors, Lines, Symbols, and Outlines

To override the default patterns and select fills and colors for the bars, use the PATTERN statement. Only solid and empty bar patterns are valid; all other pattern fills are ignored. For a complete description of all bar patterns, see " VALUE=*bar/block-pattern*" on page 457.

When you use PATTERN statements, the procedure uses the specified patterns until all of the PATTERN definitions that they generate have been used. Then, if more patterns are required, the procedure returns to the default pattern rotation. To change the outline color of any pattern, whether the pattern is default or user-defined, use the COUTLINE= option in the BAR statement that generates the chart. (See "COUTLINE=*bar-outline-color* | SAME" on page 945.) To override the default plot colors, symbols, and line widths, use the SYMBOL statement. For a complete description of its parameters, see the "SYMBOL Statement" on page 470. The SYMBOL statements are used in order for each PLOT statement. If there are fewer SYMBOL statements than PLOT statements, default SYMBOL values are used for subsequent plots.

Adding Images to Bar-Line Charts

You can apply images to the bars and to the background of bar-line charts developed with the BAR statement.

You can use PATTERN statements to specify images to fill the bars. For details, see "Displaying Images on Data Elements" on page 389.

You can use the IBACK= graphics option to specify image files that fill the background area. For additional information, including a list of recognized image file types, see "Image File Types Supported by SAS/GRAPH" on page 384 and "Displaying an Image in a Graph Background" on page 385.

Controlling When Bar Patterns Change

The PATTERNID= option controls when the pattern changes. By default, all of the bars are the same pattern. If you specify PATTERNID=MIDPOINT, then the pattern changes every time the midpoint value changes.

Instead of changing the pattern for each midpoint, you can change the pattern for each BY group by changing the value of the PATTERNID= option. See "PATTERNID=BY | MIDPOINT | SUBGROUP" on page 955 for details.

Controlling Axis Color

By default, axis elements use the first color in the color list or the colors that are specified by AXIS statement color options. However, BAR statement options can also control the color of the axis lines, text, and frame.

Table 35.2 *Controlling Axis Color*

Axis Element.	Specify Color with this Option
the axis text	CTEXT=
the axis lines	CAXIS=
the area within the frame	CFRAME=

Syntax: GBARLINE Procedure

Restriction: Not supported by Java

Requirement: One BAR statement

Supported Statements: AXIS , FOOTNOTE, GOPTIONS, LEGEND, PATTERN, SYMBOL, TITLE

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 153.

The procedure can include the SAS/GRAPH statement BY on page 424, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See “SAS/GRAPH Statements” on page 397 and *SAS DATA Step Statements: Reference* for more information.

PROC GBARLINE DATA=*input-data-set*
< ANNOTATE=*Annotate-data-set* >
< IMAGEMAP=*output-data-set* >
< UNIFORMAXES >;

BAR *bar-variable* < / *options* >;

<PLOT –1< / *options* >> <;PLOT–2< / *options* > …>;

PROC GBARLINE Statement

Identifies the data set containing the chart and response variables. Can specify an annotate data set.

Restriction: Not supported by Java

Requirement: An input data set is required.

Syntax

PROC GBARLINE DATA=*input-data-set*
< ANNOTATE=*Annotate-data-set* >
< IMAGEMAP=*output-data-set* >
< UNIFORMAXES >;

Required Argument

DATA=*input-data-set*
specifies the SAS data set that contains the variable or variables to chart. If not specified, an input data set must have been created before issuing the PROC GBARLINE statement.

Default The GBARLINE procedure uses the most recently created SAS data set.

See "The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

"About the Chart Variable" on page 928

Optional Arguments

PROC GBARLINE statement options affect all graphs produced by the procedure.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GBARLINE procedure. To annotate individual graphs, use the ANNOTATE= option in the BAR statement.

Alias ANNO=

See Chapter 28, "Using Annotate Data Sets," on page 721

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255 and "Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" on page 262

UNIFORMAXES

generates identical scales automatically on both the bar and plot axes. Use this option to perform axes scaling automatically, instead of generating the data values on the axes with the ORDER= option in the AXIS statement.

Interaction The UNIFORMAXES option overrides an ORDER= option when both are specified.

Notes The UNIFORMAXES option might automatically scale the bar axis beyond the range specified by the FORMAT statement.

When one axis range is much larger than the other, the resulting bars or plots in the graph appear flattened.

Invoking different statistical functions between the bar and plot statements when using this option might yield unexpected results. For example, specifying a cumulative percentage in a BAR statement and a cumulative frequency in a PLOT statement, in conjunction with UNIFORMAXES, might generate an abnormal graph.

BAR Statement

Creates vertical bar charts in which the height of the bars represents the value of the bar statistic for each category of data.

Restriction:	Not supported by Java
Requirement:	One bar variable is required.
Supported Statements:	AXIS , FOOTNOTE, LEGEND, PATTERN, TITLE
Supports:	Drill-down functionality

Syntax

BAR *chart-variable* < / *options* >;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate all graphs that are produced by the GBARLINE procedure.

CAUTOREF=*reference-line-color*
: specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option.

CAXIS=*axis-color*

specifies a color for the response and midpoint axis lines and for the default axis area frame.

CERROR=*error-bar-color*
specifies the color of error bars.

CFRAME=*background-color*
specifies the color with which to fill the axis area.

COUTLINE=*bar-outline-color* | SAME
outlines all bars or bar segments and legend values in the subgroup legend (if it appears) using the specified color.

CREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies colors for reference lines.

CTEXT=*text-color*
specifies a color for all text on the axes and legend.

FRAME

NOFRAME
specifies whether the axis area frame is drawn.

LAUTOREF=*reference-line-type*
specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option.

LEGEND=LEGEND<1 …99>
assigns the specified LEGEND definition to the plot part of the graph.

LREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines.

NOLEGEND
suppresses the legend generated by the LEGEND= option.

PATTERNID=BY | MIDPOINT | SUBGROUP
specifies how fill patterns are assigned.

SPACE=*bar-spacing*
specifies the amount of space between individual bars along the midpoint axis.

WAUTOREF=*reference-line-width*
specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option.

WIDTH=*bar-width*
specifies the width of the bars.

WOUTLINE=*bar-outline-width*
specifies the width of the bar outline in pixels.

WREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines.

Axes options

ASCENDING
arranges the bars in ascending order of the value of the bar statistic.

AUTOREF
draws a reference line at each major tick mark on the bar (left) response axis.

AXIS=AXIS<1 …99>
specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

CLIPREF
clips the reference lines at the bars.

DESCENDING
arranges the bars in descending order of the value of the chart statistic.

MAXIS=AXIS<1 …99>
assigns the specified AXIS definition to the midpoint axis.

MINOR=*number-of-minor-ticks*
specifies the number of minor tick marks between each major tick mark on the bar response axis.

NOAXIS
suppresses the left BAR response axis and displays the midpoint and right PLOT axes.

NOBASEREF
suppresses the zero reference line when the SUM or MEAN bar statistic has negative values.

NOZERO
suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar.

RANGE
displays on the axis of the chart the range of numeric values represented by each bar.

RAXIS=*value-list* | AXIS<1 – 99>

AXIS=*value-list* | AXIS<1 – 99>
specifies values for the major tick mark divisions on the response axis or assigns the specified AXIS definition to the axis.

REF=*value* | (*value*) | (*value-list*)
draws reference lines at the specified points on the chart response axis.

Catalog entry description options

DESCRIPTION="*description*"
specifies a description of the output.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Midpoint options

DISCRETE
treats a numeric chart variable as a discrete variable rather than as a continuous variable.

LEVELS=*number-of-midpoints* | ALL
specifies the number of midpoints to be graphed for the chart variable.

MIDPOINTS=OLD
generates default midpoints

MIDPOINTS=*value-list*
specifies the midpoint values for the bars

MISSING
accepts a missing value as a valid midpoint for the chart variable.

SUBGROUP=*subgroup-variable*
divides the bars into segments according to the values of *subgroup-variable*.

ODS options

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Statistic options

CFREQ
displays the cumulative frequency statistic above the bars.

CLM=*confidence-level*
specifies the confidence intervals to use when drawing error bars.

CPERCENT
displays the cumulative percentage statistic above the bars.

ERRORBAR=BARS | BOTH | TOP
draws confidence intervals dependent on the statistic type.

FREQ
displays the frequency statistic above the bars.

FREQ=*numeric-variable*
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

INSIDE=*statistic*
displays the values of the specified statistic inside the bars.

MEAN
displays the mean statistic above the bars.

OUTSIDE=*statistic*
displays the values of the specified statistic above the bars.

PERCENT
displays the percentages of observations having a given value for the bar variable above the bars.

STATFMT=*format-specification*
overrides the GBARLINE default format of the displayed statistical value.

SUM
displays the sum statistic above the bars.

SUMVAR=*numeric-variable*
specifies a numeric variable for sum or mean calculations.

TYPE=*statistic*
specifies the chart statistic.

Required Argument

chart-variable
specifies the variable that defines the categories of data to chart. The variable must be in the input data set.

See "About the Chart Variable" on page 928

Optional Arguments

Options in the BAR statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order. For details about specifying colors, see Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363. For details about specifying images, see Chapter 24, "Adding Images to SAS/GRAPH Output," on page 383. For a complete description of the graphics options, see "Graphics Options and Device Parameters Dictionary" on page 577.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GBARLINE procedure. To annotate individual graphs, use the ANNOTATE= option in the BAR statement.

Alias ANNO=

See Chapter 28, "Using Annotate Data Sets," on page 721

ASCENDING
arranges the bars in ascending order of the value of the bar statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. ASCENDING reorders the bars from shortest to longest. The ordering is left to right.

ASCENDING overrides any midpoint order specified in the MIDPOINTS= option or specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

AUTOREF
draws a reference line at each major tick mark on the bar (left) response axis. To draw reference lines at specific points on the response axis, use the REF= option.

By default, reference lines are drawn in front of the bars. To draw reference lines behind the bars, use the CLIPREF option.

AXIS=AXIS<1 …99>
specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

See "AXIS Statement" on page 399

CAUTOREF=*reference-line-color*
specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option. If you do not specify the CAUTOREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE system option is specified. To specify a line type for these reference lines, use the LAUTOREF= option.

Style reference Color attribute of the GraphGridLines element

CAXIS=*axis-color*
specifies a color for the response and midpoint axis lines and for the default axis area frame. If you omit the CAXIS option, the default color is defined by the current style or is the first color in the color list if the NOGSTYLE option is specified.

Style reference Color attribute of the GraphAxisLines element

CERROR=*error-bar-color*
specifies the color of error bars. The default color is the color of the response axis, which is controlled by the CAXIS= option.

Style reference Color attribute of the GraphError element

CFRAME=*background-color*
specifies the color with which to fill the axis area. The axis area color does not affect the frame color, which is always the same as the midpoint axis line color and controlled by the CAXIS= option. By default, the axis area is not filled.

The CFRAME= option is overridden by the NOFRAME option.

Alias CFR=

Style reference Color attribute of the GraphWalls element

Notes If the background color, the bar color, and the outline color are the same, then you cannot distinguish the bars.

If the specified style contains an embedded image, the image is drawn instead of the specified CFRAME color.

CFREQ
displays the cumulative frequency statistic above the bars. A maximum of two statistics can be printed if the INSIDE= option is also used.

Restriction This option is ignored if the bars are too narrow to avoid overlapping values.

Interaction The CFREQ option is ignored if the FREQ option is specified.

See "About Chart Statistics" on page 931

"Displaying Statistics in Bar-Line Charts" on page 961

CLIPREF
clips the reference lines at the bars. Using this option makes the reference lines appear to be behind the bars.

CLM=*confidence-level*
specifies the confidence intervals to use when drawing error bars. Values for *confidence-level* must be greater than or equal to 50 and strictly less than 100. The default is 95.

See "ERRORBAR=BARS | BOTH | TOP" on page 948 for details about how error bars are computed and drawn.

COUTLINE=*bar-outline-color* | SAME
outlines all bars or bar segments and legend values in the subgroup legend (if it appears) using the specified color. SAME specifies that the outline color of a bar or a bar segment or a legend value is the same as the interior pattern color.

The default outline color depends in the PATTERN statement:

- If you do not specify a PATTERN statement, the default outline color is the color of the current style.
- If you specify the NOGSTYLE system option and no PATTERN statement, the default outline color is black for the ActiveX device. Otherwise, the default

outline color is the foreground color. If you specify an empty PATTERN statement, then the default outline color is the same as the fill color.

Alias CO=

Style reference Color attribute of the GraphOutLines element

See "Controlling Patterns, Outlines, Colors, and Images" on page 935

CPERCENT

displays the cumulative percentage statistic above the bars. A maximum of two statistics can be printed using the INSIDE= option for the second statistic. This option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ, CFREQ, or PERCENT option is specified.

Alias CPCT

See "About Chart Statistics" on page 931

"Displaying Statistics in Bar-Line Charts" on page 961

CREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)

specifies colors for reference lines. Specifying a single color without parentheses applies that color to all reference lines, including lines drawn with the AUTOREF and REF= options. The CAUTOREF= option overrides the CREF= reference line color for reference lines drawn with the AUTOREF option. Specifying a single color in parentheses applies that color only to the first reference line drawn with the REF= option. Specifying a reference color list applies colors in sequence to successive lines drawn with the REF= option. The syntax of the color list requires parentheses and color names separated by spaces (*color1 color2 ...colorN*) or separated by commas (*color1*, *color2 ...*, *colorN*). If you do not specify the CREF= option, the GBARLINE procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE system option is specified. To specify line types for these reference lines, use the LREF= option.

Alias CR=

Style reference LineStyle attribute of the GraphGridLines element

CTEXT=*text-color*

specifies a color for all text on the axes and legend. This includes axis labels, tick mark values, legend labels, and legend value descriptions. The GBARLINE procedure looks for the text color in the following order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the BAR statement.
2. the color specified by the CTEXT= option in the BAR statement.
3. the color specified by the CTEXT= option in a GOPTIONS statement.
4. the color specified in the current style. However, if the NOGSTYLE system option is specified, then black for the ActiveX device and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, then the CTEXT= color overrides the AXIS statement's COLOR= specification. This means that the CTEXT= color is used for axis labels and values. The AXIS statement's COLOR= color is still used for all other axis elements, such as tick marks.

Alias CT=

Note If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

DESCENDING

arranges the bars in descending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. DESCENDING reorders the bars from longest to shortest. The ordering is left to right.

Interaction DESCENDING overrides any midpoint order that is specified with the MIDPOINTS= option or that is specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the SAS Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default BAR AND LINE PLOT OF *chart-variable*

DISCRETE

treats a numeric chart variable as a discrete variable rather than as a continuous variable. The GBARLINE procedure creates a separate midpoint and, hence, a separate bar for each unique value of the chart variable. If the chart variable has a format associated with it, then each formatted value is treated as a midpoint.

Interactions The LEVELS=*number-of-midpoints* option is ignored when you use the DISCRETE option.

The MIDPOINTS= option overrides the DISCRETE option.

The ORDER= option in an AXIS statement that is assigned to the midpoint axis can rearrange or exclude discrete midpoint values.

ERRORBAR=BARS | BOTH | TOP

draws confidence intervals for either of the following:

- the mean of the SUMVAR= variable for each midpoint if you specify TYPE=MEAN
- the percentage of observations assigned to each midpoint if you specify TYPE=PCT with no SUMVAR= option.

The ERRORBAR= option cannot be used with values of the TYPE= option other than MEAN or PCT. Valid values for ERRORBAR= are as follows:

BARS
draws error bars as bars half the width of the main bars.

BOTH
draws error bars as two ticks joined by a line (default).

TOP
draws the error bar as a tick for the upper confidence limit that is joined to the top of the bar by a line.

By default, ERRORBAR= uses a confidence level of 95%. You can specify different confidence levels with the CLM= option.

When you use ERRORBAR= with TYPE=PCT, the confidence interval is based on a normal approximation. Let TOTAL be the total number of observations, and PCT be the percentage assigned to a given midpoint. The standard error of the percentage is approximated as follows:

```
APSTDERR=100 * SQRT((PCT/100) * (1-(PCT/100)) / TOTAL);
```

Let LEVEL be the confidence level specified using the CLM= option, with a default value of 95. The upper confidence limit for the percentage is computed as follows:

```
UCLP = PCT + APSTDERR * PROBIT( 1-(1-LEVEL/100)/2 );
```

The lower confidence limit for the percentage is computed as follows:

```
LCLP = PCT - APSTDERR * PROBIT( 1-(1-LEVEL/100)/2 );
```

When you use ERRORBAR= with TYPE=MEAN, the sum variable must have at least two nonmissing values for each midpoint. Let N be the number of observations assigned to a midpoint. Let, MEAN be the mean of those observations and STD be the standard deviation of the observations. The standard error of the mean is computed as follows:

```
STDERR = STD / SQRT(N);
```

Let LEVEL be the confidence level specified using the CLM= option, with a default value of 95. The upper confidence limit for the mean is computed as follows:

```
UCLM = MEAN + STDERR * TINV( 1-(1-LEVEL/100)/2, N-1);
```

The lower confidence limit for the mean is computed as follows:

```
LCLM = MEAN - STDERR * TINV( 1-(1-LEVEL/100)/2, N-1);
```

Suppose you want the error bars to represent a given number, C, of standard errors instead of a confidence interval, and the number of observations assigned to each midpoint is the same. You can then find the appropriate value for the CLM= option by running a DATA step. For example, suppose you want error bars that represent one standard error (C=1) with a sample size of N. You can run the following DATA step to compute the appropriate value for the CLM= option and assign that value to a macro variable &LEVEL:

```
data null; c = 1; n = 10;
level = 100 * (1 - 2 * (1 - probt( c, n-1)));
put all;
call symput("level",put(level,best12.));
run;
```

Then, when you run the GBARLINE procedure, you can specify CLM=&LEVEL.

Note that this method does not work precisely if different midpoints have different numbers of observations. However, choosing an average value for N can yield sufficiently accurate results for graphical purposes if the sample sizes are large or do not vary much.

FRAME | NOFRAME

specifies whether the axis area frame is drawn. The default is FRAME, which draws a frame around the axis area. Specifying NOFRAME removes the axis area frame, including any background color or image. To remove one or more axis elements, use either the AXIS statement or the NOAXIS option.

The color of the frame or backplane outline is the color of the midpoint axis, which is determined by the CAXIS= option.

Alias FR | NOFR

Interaction The NOFRAME option overrides the CFRAME= option and the IBACK= graphics option.

FREQ

displays the frequency statistic above the bars. Non-integer values are rounded down to the nearest integer. A maximum of two statistics can be printed using the INSIDE= option for the second. This option is ignored if the bars are too narrow to avoid overlapping values.

Interaction This option overrides the CFREQ, PERCENT, CPERCENT, SUM, and MEAN options.

See “About Chart Statistics” on page 931

“Displaying Statistics in Bar-Line Charts” on page 961

FREQ=*numeric-variable*

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times that is specified by the value of *numeric-variable* for that observation. If the value of *numeric-variable* is missing, zero, or negative, then the observation is not used in the statistic calculation. Non-integer values of *numeric-variable* are truncated to integers. The FREQ= option is valid with all chart statistics.

You cannot use TYPE=PERCENT, TYPE=CPERCENT, TYPE=FREQ, or TYPE=CFREQ with the SUMVAR= option. You must use the FREQ= option to

calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum.

The statistics are affected by applying a format to *numeric-variable*.

Restriction Not supported by Java and ActiveX

See "Calculating Weighted Statistics" on page 933

HTML=*variable*

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction Not supported by Java

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

Example "Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs" on page 978

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

INSIDE=*statistic*

displays the values of the specified statistic inside the bars. *Statistic* can be one of the following:

- FREQ
- CFREQ
- PERCENT | PCT
- CPERCENT | CPCT
- SUM
- MEAN

Interaction To display statistics with INSIDE=SUM or INSIDE=MEAN, you must also specify the SUMVAR= option.

See "About Chart Statistics" on page 931

"Displaying Statistics in Bar-Line Charts" on page 961

LAUTOREF=*reference-line-type*

specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style. If the NOGSTYLE system option is specified, the default value is 1, which draws a solid line. To specify a color for these reference lines, use the CAUTOREF= option.

Style reference LineStyle attribute of the GraphGridLines element

LEGEND=LEGEND<1 …99>

assigns the specified LEGEND definition to the plot part of the graph. The LEGEND= option is ignored if the specified LEGEND definition is not in effect. When you specify the LEGEND option, the BAR statement generates a legend even if the SUBGROUP= option is not specified. This output differs from the output generated by the GCHART procedure where the SUBGROUP option creates a legend by default. In this case, only one bar is represented in the legend.

To create a legend based on the chart midpoints instead of the subgroups, use the chart variable as the subgroup variable: `bar city / subgroup=city;`

You can specify both a BAR and PLOT legend on the same graph. If you specify the same location with the LEGEND= option in the BAR statement and the POSITION= option in the LEGEND statement, a single combined legend is drawn. However, ActiveX output displays separate but adjacent legends.

The ActiveX device does not support all LEGEND statement options. See the "LEGEND Statement" on page 432 for more information.

Restriction Partially supported by ActiveX

See "SUBGROUP=*subgroup-variable*" on page 958

"LEGEND Statement" on page 432

Example "Example 2: Grouping and Subgrouping a Block Chart" on page 1095

LEVELS=*number-of-midpoints* | ALL

specifies the number of midpoints to be graphed for the chart variable. After you specify the number of midpoints that you want, the range for each numeric midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985). If you specify LEVELS=ALL, then all unique numeric or character midpoint values are graphed. If your data contains a large number of unique midpoint values (more than 200), then you can use the XPIXELS and YPIXELS GOPTIONS. This enables the device driver to render a larger (and more readable) graph. The LEVELS=*number-of-midpoints* option is ignored if any of the following is true:

- The chart variable is character type.
- The DISCRETE option is used.
- The MIDPOINTS= option is used.

LREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. Specifying a line type without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the LAUTOREF= option overrides LREF=*reference-line-type* for reference lines drawn with the AUTOREF option. Specifying a single line type in parentheses applies that line type to the first reference line drawn with the REF= option. Specifying a line type list applies line types in sequence to successive reference lines drawn with the REF= option. The syntax of the line-type list requires parentheses and is delimited by spaces (*type1 type2 ...typeN*). If you do not specify the LREF= option, the GBARLINE procedure uses the type specified by the AXIS statement's STYLE= option. If neither option is specified, the default line type is retrieved from the current style. If the NOGSTYLE system option is specified, the default value is 1, which draws a solid line. To specify colors for these reference lines, use the CREF= option.

Style reference GraphReference

MAXIS=AXIS<1 …99>
assigns the specified AXIS definition to the midpoint axis. The MAXIS= option is ignored if the specified AXIS definition does not exist.

See "AXIS Statement" on page 399

"About Midpoints" on page 929

MEAN
displays the mean statistic above the bars. A maximum of two statistics can be printed using the INSIDE= option for the second.

Restriction This option is ignored if the bars are too narrow to avoid overlapping values.

Interactions This option is ignored if the FREQ, CFREQ, PERCENT, CPERCENT, or SUM option is specified.

The MEAN option is ignored unless you also use the SUMVAR= option.

See "About Chart Statistics" on page 931

"Displaying Statistics in Bar-Line Charts" on page 961

MIDPOINTS=*value-list*
specifies the midpoint values for the bars The way you specify *value-list* depends on the type of the bar variable.

- For numeric chart variables, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - *n* < *...n*>
  - *n* TO *n* <BY *increment*>
  - *n*< *...n*> TO *n* <BY *increment*> <*n* < *...n*>>

  If a numeric chart variable has an associated format, the specified values must be the *unformatted* values.

If you omit the DISCRETE option, then by default these statements are true:

- Numeric variable values are treated as continuous.
- The lowest midpoint consolidates all data points from negative infinity to the average of the first two midpoints.
- The highest midpoint consolidates all data points from the average of the last two midpoints up to infinity.
- All other values in *value-list* specify the median of a range of values, and the GBARLINE procedure calculates the midpoint values.

  If you include the DISCRETE option, then each value in *value-list* specifies a unique numeric value.

- For character bar variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:
- *'value-1' < ...'value-n'>*

  If a character variable has an associated format, the specified values must be the *formatted* values.

  You are able to specify up to 256 characters for each variable value.

For a complete description of *value-list*, see " ORDER=(*value-list*)" on page 407 in the AXIS statement.

If the *value-list* for either type of variable specifies so many midpoints that the axis values overwrite each other, then the values might be unreadable. In this case the procedure writes a warning to the SAS log. On many devices, this problem can be corrected by adjusting the size of the text with the HTEXT= graphics option. Or correct the problem by increasing the number of cells that are displayed in your graphics. Do this by using the HPOS= and VPOS= graphics options.

The ORDER= option in the AXIS statement overrides the order specified in the MIDPOINTS= option. The BAR statement options ASCENDING and DESCENDING also override both the MIDPOINTS= and ORDER= options in the AXIS statement.

Note Any character bar variable value with a long text string has the potential to cause a scaling issue and might produce a wider bar chart than desired.

See "About Midpoints" on page 929

MIDPOINTS=OLD

generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Restriction The MIDPOINTS=OLD option is ignored unless the chart variable is numeric.

MINOR=*number-of-minor-ticks*

specifies the number of minor tick marks between each major tick mark on the bar response axis.

Interaction The MINOR= option in a bar chart statement overrides the MINOR= option in an AXIS definition assigned to the response axis with the RAXIS= option.

MISSING

accepts a missing value as a valid midpoint for the chart variable. By default, observations with missing values are ignored.

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
  - If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

    Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

  - The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GBARLIN

NOAXIS

suppresses the left BAR response axis and displays the midpoint and right PLOT axes. The axis lines, axis labels, axis values, and all major and minor tick marks are suppressed on the left axis. If you specify an axis definition with the MAXIS= or RAXIS= options, then the axes are generated as defined in the AXIS

statement. However, all lines, labels, values, and tick marks are suppressed. Therefore, AXIS statement options such as ORDER=, LENGTH=, and OFFSET= are used.

To remove only selected axis elements such as lines, values, or labels, use specific AXIS statement optionsF. If NOAXIS is specified for both the BAR and PLOT statements, both response axes and the midpoint axis are suppressed.

NOAXIS does not suppress either the default frame or an axis area fill requested by the CFRAME= option. To remove the axis frame, use the NOFRAME option in the procedure.

NOBASEREF
suppresses the zero reference line when the SUM or MEAN bar statistic has negative values.

NOLEGEND
suppresses the legend generated by the LEGEND= option.

NOZERO
suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar.

Note There might be occasions when you assign bar label names to each bar with the VALUE= option in an AXIS statement and a bar is omitted from your graph. In this case the label names might be inadvertently shifted and assigned to the wrong bar.

OUTSIDE=*statistic*
displays the values of the specified statistic above the bars. *Statistic* can be one of the following:

- FREQ
- CFREQ
- PERCENT | PCT
- CPERCENT | CPCT
- SUM
- MEAN

Interaction To display statistics with OUTSIDE=SUM or OUTSIDE=MEAN, you must also specify the SUMVAR= option. A second statistic can be displayed by also using the INSIDE= option.

See “About Chart Statistics” on page 931

“Displaying Statistics in Bar-Line Charts” on page 961

PATTERNID=BY | MIDPOINT | SUBGROUP
specifies how fill patterns are assigned. By default, all of the bars are the same color. Values for PATTERNID= are as follows:

BY
changes patterns each time the value of the BY variable changes. All bars use the same pattern if the GBARLINE procedure does not include a BY statement.

MIDPOINT
changes patterns every time the midpoint value changes.

SUBGROUP
changes patterns every time the value of the subgroup variable changes. The bars must be subdivided by the SUBGROUP= option for the SUBGROUP value to have an effect. Without the SUBGROUP= option, all bars have the same pattern.

PERCENT
displays the percentages of observations having a given value for the bar variable above the bars. A maximum of two statistics can be printed using the INSIDE= option for the second.

Alias PCT

Restriction This option is ignored if the bars are too narrow to avoid overlapping values.

Interaction The PERCENT option is ignored if the FREQ or CFREQ option is specified.

See "About Chart Statistics" on page 931

"Displaying Statistics in Bar-Line Charts" on page 961

RANGE
displays on the axis of the chart the range of numeric values represented by each bar. In the graphics output:

- The starting value of each range is indicated with the less-than symbol (<).
- The ending value is indicated with the greater-than-or-equal-to symbol (>=).
- The default midpoint axis label is "*chart-variable* RANGE."

The RANGE option has no affect on axes that represent character data. By default, the values shown on the axis are determined by the value of the "MIDPOINTS=*value-list*" on page 952.

Interactions If specified, the option "DISCRETE" on page 947 overrides the RANGE option.

During the creation of vertical bar charts, when options RANGE and LEVELS= are both used, RANGE displays the endpoints of the range for each vertical bar. It does this rather than displaying just the numerical midpoint of the range as would be displayed with the LEVELS= option alone. If necessary, RANGE automatically angles or rotates the values and they might be less readable

If the RANGE, LEVELS=, and AXIS VALUE= options are all specified, the RANGE option is ignored and the LEVELS= option displays the numeric midpoint of each vertical bar.

RAXIS=*value-list* | AXIS<1 – 99>
AXIS=*value-list* | AXIS<1 – 99>
specifies values for the major tick mark divisions on the response axis or assigns the specified AXIS definition to the axis. See "MIDPOINTS=*value-list*" on page 952 for a description of *value-list*. By default, the GBARLINE procedure scales the response axis automatically and provides an appropriate number of tick marks. The left response axis applies to the BAR statement when a PLOT statement is used. Otherwise, both the left and right axes apply to the BAR statement.

You can specify negative values, but negative values are reasonable only when TYPE=SUM or TYPE=MEAN and one or more of the sums or means are less than zero. Frequency and percentage values are never less than zero.

For lists of values, a separate major tick mark is created for each individual value. A warning message is written to the SAS log if the values are not evenly spaced.

If the values represented by the bars are larger than the highest tick mark value, then the bars are truncated at the highest tick mark.

See "AXIS Statement" on page 399

REF=*value* | (*value*) | (*value-list*)
draws reference lines at the specified points on the chart response axis.

See "MIDPOINTS=*value-list*" on page 952 for a description of *value-list*.

Values can be listed in any order, but should be within the range of values represented by the chart response axis. A warning is written to the SAS log if any of the points are off of the axis, and no reference line is drawn for such points. You can use the AUTOREF option to draw reference lines automatically at all of the major tick marks.

SPACE=*bar-spacing*
specifies the amount of space between individual bars along the midpoint axis. *Bar-spacing* can be any nonnegative number, including decimal values. Units are character cells. By default, the GBARLINE procedure calculates spacing based on the size of the axis area and the width of the bars on the chart. Use SPACE=0 to leave no space between adjacent bars.

Restriction The SPACE= option is ignored if its value results in a chart that is too large to fit in the space available for the midpoint axis. As a result, a warning message is issued in the log.

STATFMT=*format-specification*
overrides the GBARLINE default format of the displayed statistical value.

Note: This option is valid starting with SAS 9.4M5.

The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use the STATFMT= option to change the default format that might contain decimal points, percentages, or commas.

If you specify options TYPE=mean, STATFMT=f6.1, INSIDE=sum, and OUTSIDE=mean, the STATFMT=f6.1 option applies only to the calculated TYPE=mean statistical value. Here is an example:

```
footnote "Total revenue is displayed inside each bar.";
proc gbarline data=sashelp.electric;
   bar year / discrete width=6 sumvar=Revenue
      type=mean statfmt=f6.1 inside=sum outside=mean;
run;
quit;
footnote;
```

In this case, the INSIDE= option values are displayed in their default format. They are unaffected by the STATFMT= option. The OUTSIDE= option values are displayed in the format that is specified in the STATFMT= option.

If you change the previous example to specify option TYPE=sum, then the STATFMT=f6.1 option applies only to the calculated TYPE=sum statistical value. In that case, the OUTSIDE= values are displayed in their default format, and the INSIDE= values are displayed in the format that is specified in the STATFMT= option.

Alias SFMT=, SFORMAT=, STATFORMAT=

Note The STATFMT= option overrides the format of the response-axis tick values.

SUBGROUP=*subgroup-variable*
divides the bars into segments according to the values of *subgroup-variable*. *Subgroup-variable* can be either character or numeric and is always treated as a discrete variable. The SUBGROUP= option creates a separate segment within each bar for every unique value of the subgroup variable for that midpoint.

When you specify the LEGEND option, the BAR statement generates a legend even if the SUBGROUP= option is not specified. This output differs from the output generated by the GCHART procedure where the SUBGROUP option automatically creates a legend by default. In this case, only one bar is represented in the legend. To assign a LEGEND definition, use the LEGEND= option.

See "LEGEND Statement" on page 432

Example "Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs" on page 978

SUM
displays the sum statistic above the bars. A maximum of two statistics can be printed using the INSIDE= option for the second.

Restriction This option is ignored if the bars are too narrow to avoid overlapping values.

Interactions This option is ignored if the FREQ, CFREQ, PERCENT, or CPERCENT option is specified.

The SUM option is ignored unless you also use the SUMVAR= option.

See "About Chart Statistics" on page 931

"Displaying Statistics in Bar-Line Charts" on page 961

SUMVAR=*numeric-variable*
specifies a numeric variable for sum or mean calculations. The GBARLINE procedure calculates the sum or, if requested, the mean of *numeric-variable* for each midpoint. The resulting statistics are represented by the length of the bars along the response axis, and they are displayed at major tick marks.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

Example "Example 1: Producing a Basic Bar-Line Chart" on page 974

TYPE=*statistic*
specifies the chart statistic.

- If the SUMVAR= option is not used, *statistic* can be one of the following:

FREQ
frequency (default)

CFREQ
cumulative frequency

PERCENT
percentage

CPERCENT
cumulative percentage

- If the SUMVAR= option is used, *statistic* can be one of the following:

SUM
sum (default)

MEAN
mean

You cannot use TYPE=PERCENT, TYPE=CPERCENT, TYPE=FREQ, or TYPE=CFREQ with the SUMVAR= option. You must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum. See also "Calculating Weighted Statistics" on page 933.

See "About Chart Statistics" on page 931 for a complete description of statistic types

URL=*character-variable*
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction This option affects graphics output that is created through the ODS HTML destination only

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

WAUTOREF=*reference-line-width*
specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CAUTOREF= option.

WIDTH=*bar-width*
specifies the width of the bars. By default, the GBARLINE procedure selects a bar width that accommodates the midpoint values displayed on the midpoint axis using a hardware font and a height of one cell. Units for *bar-width* are character cells. The value for *bar-width* must be greater than zero, but it does not have to be an integer, for example: `bar site / width=1.5;`

Restrictions Sometimes the requested bar width results in a chart that is too large to fit in the space available for the midpoint axis. In this case the procedure issues a warning in the SAS log and ignores the WIDTH= specification.

If the specified width is too narrow, the procedure might display the midpoint values vertically.

WOUTLINE=*bar-outline-width*
specifies the width of the bar outline in pixels. WOUTLINE= affects both the slice and subgroup outlines.

Style reference LineThickness attribute of the GraphOutLines element

WREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines. Line widths are specified as whole numbers. Specifying a line width without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the WAUTOREF= option overrides WREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the REF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the REF= option. The syntax of the line-width list requires parentheses and widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the CREF= option.

Style reference LineThickness attribute of the GraphReference element

Details

Description

The BAR statement specifies the variable that defines the categories of data to chart. This statement automatically performs the following operations:

- determines the midpoints
- calculates the chart statistic for each midpoint (the default is FREQ)
- scales the response axis and the bars according to the statistic value
- determines bar width and spacing
- assigns patterns to the bars (the default bar pattern is SOLID)
- draws a frame around the axis area using the color defined by the current style or the first color in the color list if the NOGSTYLE system option is specified.

You can use statement options to do the following:

- select or order the midpoints (bars)
- control the tick marks on the response axis
- change the type of chart statistic

- display specific statistics
- modify the appearance of the chart

You can also specify additional variables by which to subgroup or sum the data.

Bar charts support subgroups, which subdivide the bars into segments based on the values of a subgroup variable.

In addition, you can do the following actions:

- use global statements to add a legend, modify the axes, and change the bar patterns. See "SAS/GRAPH Statements" on page 397 for more information.
- add titles and footnotes to the chart. See "TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement" on page 507 for more information.
- use an Annotate data set to enhance the chart. See Chapter 28, "Using Annotate Data Sets," on page 721 for more information.
- display an image in the background of the chart. See "IBACK Graphics Option" on page 651 for more information.
- display images in the bars of the chart. See the option "IMAGE=*fileref* | "*external-file*"" on page 456 for the PATTERN statement.

The Chart Statistic and the Response Axis

In bar-line charts, the scale of values of the chart statistic is displayed on the left response axis. By default, the response axis is divided into evenly spaced intervals identified with major tick marks that are labeled with the corresponding statistic value. Minor tick marks are evenly distributed between the major tick marks unless a log axis has been requested. For sum and mean statistics, the major tick marks are labeled with values of the SUMVAR= variable (formatted if the variable has an associated format). The response axis is also labeled with the statistic type.

Specifying Logarithmic Axes

Logarithmic axes can be specified with the AXIS statement.

See "SAS/GRAPH Statements" on page 397 for a complete description.

Displaying Statistics in Bar-Line Charts

Statistic values on bar-line charts are not printed by default, so you must explicitly request a statistic with the FREQ, CFREQ, PERCENT, CPERCENT, SUM, MEAN, INSIDE=, or OUTSIDE= option.

For graphs generated with the ActiveX device, you can display one statistic for each bar. For graphs generated with other devices, you can display up to two statistics for each bar. Statistics can be displayed either above the bars or inside the bars.

To specify a statistic that you want to display above the bars, specify the statistic option (FREQ, CFREQ, PERCENT, CPERCENT, SUM, or MEAN) or specify OUTSIDE=*statistic*. To specify a statistic that you want to display inside the bars, specify INSIDE=*statistic*.

For graphs generated with the ActiveX device, the OUTSIDE= option overrides INSIDE=, and INSIDE= overrides the FREQ, CFREQ, PERCENT, CPERCENT,

SUM, and MEAN options. For graphs generated with other devices, the individual statistic options override the OUTSIDE= option.

If more than one statistic option is specified, only the highest priority statistic is displayed. The priority order, from highest to lowest, is as follows:

1. FREQ
2. CFREQ
3. PERCENT
4. CPERCENT
5. SUM
6. MEAN

The bars must be wide enough to accommodate the text. You can adjust the width of the bars with the WIDTH= option. To control the font and size of the text, use the HTEXT= and FTEXT= graphics options.

Ordering and Selecting Midpoints

To rearrange character or discrete numeric midpoint values or to select ranges for numeric values, use the MIDPOINTS= option. Changing the number of midpoints for numeric variables changes the range of values for individual midpoints. However, it does not change the range of values for the chart as a whole. For details, see "About Midpoints" on page 929.

Like the MIDPOINTS= option, the ORDER= option in the AXIS statement can rearrange the order of the midpoints or suppress the display of discrete numeric or character values. However, the ORDER= option cannot calculate the midpoints for a continuous numeric variable, nor can it exclude values from the calculations. For details, see the option description for " ORDER=(*value-list*) | DESCENDING" on page 438.

PLOT Statement

Creates one or more plot overlays on top of the bar-line chart.

Restriction:	Not supported by Java
Requirement:	This option affects graphics output that is created through the ODS HTML destination only. If specified, a PLOT statement or statements must be specified after the BAR statement.
Supported Statements:	AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE
Supports:	Data tips and drill-down functionality

Syntax

PLOT < / *options(s)* >;

<PLOT –1< / *options* >> <;PLOT–2< / *options* > …>;

Summary of Optional Arguments

Appearance options

CAUTOREF=*reference-line-color*
: specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option.

CAXIS=*axis-color*
: specifies a color for the tick marks and for the axis area frame on the plot (right) response axis.

CREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
: specifies colors for reference lines.

CTEXT=*text-color*
: specifies a color for all text on the plot response axis and legend.

LAUTOREF=*reference-line-type*
: specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option.

LEGEND=LEGEND<1 …99>
: generates a legend and assigns the specified LEGEND definition to the legend.

LREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
: specifies line types for reference lines.

NOLINE
: suppresses the line connecting the PLOT symbols

NOMARKER
: suppresses drawing the marker symbol.

WAUTOREF=*reference-line-width*
: specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option.

WREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
: specifies line widths for reference lines.

Axes options

ASCENDING
: joins the plot points in ascending order of the value of the plot statistic.

AUTOREF
: draws a reference line at each major tick mark on the plot (right) response axis.

AXIS=AXIS<1 …99>
: specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

CLIPREF
: clips the reference lines at the bars.

DESCENDING

joins the plot points in descending order of the value of the plot statistic.

MINOR=*number-of-minor-ticks*
specifies the number of minor tick marks that are drawn between each major tick mark on the PLOT response axis.

NOAXIS
suppresses the right PLOT response axis and displays the midpoint and left BAR axes.

RAXIS=*value-list*99> AXIS= | AXIS<1 – *value-list* | AXIS<1 – 99>
specifies the major tick mark values for the PLOT (right) response axis or assigns an AXIS definition.

REF=*value* | (*value*) | (*value-list*)
draws reference lines at the specified points using the chart response axis.

ODS options

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Statistic options

CFREQ
displays the cumulative frequency statistic above each plot data point.

CPERCENT
displays the cumulative percentage statistic above each plot data point.

FREQ
displays the frequency statistic above each plot data point.

FREQ=*numeric-variable*
specifies a variable whose values weight the contribution of each observation in the computation of the plot statistic.

MEAN
displays the mean statistic above each plot data point.

PERCENT
displays the percentages of observations that have a given value for the plot variable above each plot data point.

STATFMT=*format-specification*
overrides the GBARLINE default format of the displayed statistical value.

SUM
displays the sum statistic above each plot data point.

SUMVAR=*plot-variable*
specifies the variable to plot.

TYPE=*statistic*
specifies the plot statistic.

Optional Arguments

You can specify as many options as you want and list them in any order.

ASCENDING
joins the plot points in ascending order of the value of the plot statistic. By default, plot points are connected from left to right.

AUTOREF
draws a reference line at each major tick mark on the plot (right) response axis. To draw reference lines at specific points on the response axis, use the REF= option. By default, reference lines are drawn in front of the bars. To draw reference lines behind the bars, use the CLIPREF option.

AXIS=AXIS<1 …99>
specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

See "RAXIS=*value-list* | 99> AXIS=AXIS<1 – *value-list* | AXIS<1 – 99>" on page 969

CAUTOREF=*reference-line-color*
specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option. If you do not specify the CAUTOREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE system option is specified. To specify a line type for these reference lines, use the LAUTOREF= option.

Style reference Color attribute of the GraphGridLines element

CAXIS=*axis-color*
specifies a color for the tick marks and for the axis area frame on the plot (right) response axis. If you omit the CAXIS option, the default color is the color defined by the default style or is the first color in the color list.

CFREQ
displays the cumulative frequency statistic above each plot data point. If symbols are used, it appears above each symbol. CFREQ is ignored if the PERCENT or CPERCENT option is specified.

See "About Chart Statistics" on page 931

CLIPREF
clips the reference lines at the bars. Using this option makes the reference lines appear to be behind the bars.

CPERCENT
displays the cumulative percentage statistic above each plot data point. If symbols are used, it appears above each symbol. CPERCENT overrides the FREQ, CFREQ, and PERCENT options.

Alias CPCT

See "About Chart Statistics" on page 931

CREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies colors for reference lines. Specifying a single color without parentheses applies that color to all reference lines, including lines drawn with the AUTOREF and REF= options. The CAUTOREF= option overrides the CREF= reference line

color for reference lines drawn with the AUTOREF option. Specifying a single color in parentheses applies that color only to the first reference line drawn with the REF= option. Specifying a reference color list applies colors in sequence to successive lines drawn with the REF= option. The syntax of the color list requires parentheses and color names separated by spaces (*color1 color2 ...colorN*) or separated by commas (*color1*, *color2* ..., *colorN*). If you do not specify the CREF= option, the GBARLINE procedure uses the color specified by the CAXIS= option. If neither option is specified, then the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified. To specify line types for these reference lines, use the LREF= option.

Alias CRF=

Style reference LineStyle attribute of the GraphGridLines element

CTEXT=*text-color*

specifies a color for all text on the plot response axis and legend. This includes axis labels, tick mark values, legend labels, and legend value descriptions. The GBARLINE procedure looks for the text color in the following order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the PLOT statement.
2. the color specified by the CTEXT= option in the PLOT statement
3. the color specified by the CTEXT= option in a GOPTIONS statement.
4. the color specified in the current style. However, if the NOGSTYLE system option is specified, then black for the ActiveX device and the first color in the color list for all other devices

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, then the CTEXT= color overrides the AXIS statement's COLOR= specification. This means that the CTEXT= color is used for axis labels and values. The COLOR= color is still used for all other axis elements, such as tick marks.

Alias CT=

Style reference GraphLabelText, GraphValueText

Note If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

DESCENDING

joins the plot points in descending order of the value of the plot statistic. By default, plot points are connected from left to right.

FREQ

displays the frequency statistic above each plot data point. If symbols are used, it appears above each symbol. Non-integer values are rounded down to the nearest integer.

Restriction FREQ is ignored if the CFREQ, PERCENT, or CPERCENT option is specified.

See "About Chart Statistics" on page 931

FREQ=*numeric-variable*
specifies a variable whose values weight the contribution of each observation in the computation of the plot statistic. Each observation is counted the number of times that is specified by the value of *numeric-variable* for that observation. If the value of *numeric-variable* is missing, zero, or negative, then the observation is not used in the statistic calculation. Non-integer values of *numeric-variable* are truncated to integers. The FREQ= option is valid with all plot statistics.

You cannot use TYPE=PERCENT, TYPE=CPERCENT, TYPE=FREQ, or TYPE=CFREQ with the SUMVAR= option. You must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum.

The statistics are not affected by applying a format to *numeric-variable*.

Restriction Not supported by ActiveX

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction Not supported by Java or ActiveX

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

Example "Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs" on page 978

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java and ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

LAUTOREF=*reference-line-type*
specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify a color for these reference lines, use the CAUTOREF= option.

LEGEND=LEGEND<1 …99>

generates a legend and assigns the specified LEGEND definition to the legend. The LEGEND= option is ignored if the specified LEGEND definition is not in effect. When you specify the LEGEND option, the BAR statement generates a legend even if the SUBGROUP= option is not specified. This output differs from the output generated by the GCHART procedure where the SUBGROUP option automatically creates a legend by default. In this case, only one bar is represented in the legend.

You can specify both a BAR and PLOT legend on the same graph. If you specify the same location with the LEGEND= option in the BAR statement and the POSITION= option in the LEGEND statement, a single combined legend is drawn. However, ActiveX output displays separate but adjacent legends.

The ActiveX device does not support all LEGEND statement options.

Restriction Not supported by Java. Partially supported by ActiveX.

Note Only one PLOT statement can contain a LEGEND= reference. If you request a PLOT legend, then all of the PLOT lines are displayed in the legend.

See "LEGEND Statement" on page 432

Example "Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs" on page 978

LREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)

specifies line types for reference lines. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. Specifying a line type without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the LAUTOREF= option overrides LREF=*reference-line-type* for reference lines drawn with the AUTOREF option. Specifying a single line type in parentheses applies that line type to the first reference line drawn with the REF= option. Specifying a line type list applies line types in sequence to successive reference lines drawn with the REF= option. The syntax of the line-type list requires parentheses and types separated by spaces (*type1 type2 ...typeN*). If you do not specify the LREF= option, the GBARLINE procedure uses the type specified by the AXIS statement's STYLE= option. If neither option is specified, the default line type is retrieved from the current style. If the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these reference lines, use the CREF= option.

Alias LR=

Style reference GraphReference

Restriction Not supported by Java

MEAN

displays the mean statistic above each plot data point. If symbols are used, it appears above each symbol. MEAN is ignored unless you also use the SUMVAR= option.

See "About Chart Statistics" on page 931

MINOR=*number-of-minor-ticks*
specifies the number of minor tick marks that are drawn between each major tick mark on the PLOT response axis. Minor tick marks are not labeled. The MINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

NOAXIS
suppresses the right PLOT response axis and displays the midpoint and left BAR axes. The axis lines, axis labels, axis values, and all major and minor tick marks are suppressed on the right axis. If you specify an axis definition with the MAXIS= or RAXIS= options, then the axes are generated as defined in the AXIS statement. However, all lines, labels, values, and tick marks are suppressed. Therefore, AXIS statement options such as ORDER=, LENGTH=, and OFFSET= are still used.

To remove only selected axis elements such as lines, values, or labels, use specific AXIS statement options.

NOAXIS does not suppress either the default frame or an axis area fill requested by the CFRAME= option. To remove the axis frame, use the NOFRAME option in the procedure.

Alias NOAXES

NOLINE
suppresses the line connecting the PLOT symbols This is true regardless of what is specified in the SYMBOL statement.

NOMARKER
suppresses drawing the marker symbol. This is true regardless of what is specified in the SYMBOL statement.

PERCENT
displays the percentages of observations that have a given value for the plot variable above each plot data point. If symbols are used, it appears above each symbol.

Alias PCT

Restriction The PERCENT option is ignored if the CPERCENT option is specified.

See “About Chart Statistics” on page 931

RAXIS=*value-list*99> AXIS= | AXIS<1 – *value-list* | AXIS<1 – 99>
specifies the major tick mark values for the PLOT (right) response axis or assigns an AXIS definition. The way you specify *value-list* depends on the type of variable:

- *For numeric variables*, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - *n* < *...n*>
  - *n* TO *n* <BY *increment*>
  - *n* < *...n*> TO *n* <BY *increment* > <*n* < *...n*> >

  If a numeric variable has an associated format, the specified values must be the *unformatted* values.

- *For date-time values*, *value-list* includes any SAS date, time, or datetime value described for the SAS functions INTCK and INTNX, shown here as *SAS-value*:
  - '*SAS-value*'*i* < ...'*SAS-value*'*i*>
  - '*SAS-value*'*i* TO '*SAS-value*'*i*<BY *interval*>

Restriction Any response values that exceed the highest tick mark value are not plotted.

Note The overlay plot line connects only the visible plot response values.

REF=*value* | (*value*) | (*value-list*)
draws reference lines at the specified points using the chart response axis. See "MIDPOINTS=*value-list*" on page 952 for a description of *value-list*.

Values can be listed in any order, but should be within the range of values represented by the PLOT response axis. A warning is written to the SAS log if any of the points are off of the axis, and no reference line is drawn for such points. You can use the AUTOREF option to draw reference lines automatically at all of the major tick marks.

STATFMT=*format-specification*
overrides the GBARLINE default format of the displayed statistical value.

Note: This option is valid starting with SAS 9.4M5.

The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use the STATFMT= option to change the default format that might contain decimal points, percentages, or commas. Here is an example:

```
axis1 label=("Mean Revenue ($B)");
axis2 label=("Total Revenue ($B)");
proc gbarline data=sashelp.electric;
   bar year / discrete width=5 sumvar=Revenue type=mean statfmt=f6.1
      inside=mean raxis=axis1;
   plot / sumvar=Revenue type=sum statfmt=f6. sum raxis=axis2;
run;
quit;
axis;
```

In this example, the STATFMT= option in the PLOT statement specifies format F6. for the SUM values that are displayed above the plot symbols.

Alias SFMT=, SFORMAT=, STATFORMAT=

Note The STATFMT= option has no effect on the format of the response-axis tick values. To format the response-axis values, use a FORMAT statement in the GBARLINE procedure step to apply a format to the variable specified in SUMVAR= or FREQ=.

SUM
displays the sum statistic above each plot data point. If symbols are used, it appears above each symbol.

Restrictions The SUM option is ignored unless you also use the SUMVAR= option.

This option is ignored if the MEAN option is specified.

See "About Chart Statistics" on page 931

SUMVAR=*plot-variable*
specifies the variable to plot. *Plot-variable*, if specified, must be numeric. The GBARLINE procedure calculates the sum or, if requested, the mean of *plot-variable* for each midpoint.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

See "About Response Variables" on page 931

Example "Example 1: Producing a Basic Bar-Line Chart" on page 974

TYPE=*statistic*
specifies the plot statistic.

- If the SUMVAR= option is not used, *statistic* can be one of the following:

FREQ
frequency (default)

CFREQ
cumulative frequency

PERCENT
percentage

CPERCENT
cumulative percentage

- If SUMVAR= is used, *statistic* can be one of the following:

SUM
sum (default)

MEAN
mean

Because you cannot use TYPE=FREQ, TYPE=CFREQ, TYPE=PERCENT, or TYPE=CPERCENT with SUMVAR=, you must use the FREQ= option to calculate percentages or frequencies based on a sum.

See "About Chart Statistics" on page 931

"Calculating Weighted Statistics" on page 933

URL=*character-variable*
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only.

Not supported by GIF, PNG, and ACTIVEX devices.

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored.

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

WAUTOREF=*reference-line-width*
specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CAUTOREF= option.

Style reference LineThickness attribute of the GraphGridLines element

WREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines. Line widths are specified as whole numbers. Specifying a line width without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the WAUTOREF= option overrides WREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the REF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the REF= option. The syntax of the line-width list is (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the CREF= option.

Style reference LineThickness attribute of the GraphReference element

Details

Using the PLOT Statement

PLOT statements are optional, but if specified, they must follow the BAR statement. If you do not specify any PLOT statements, GBARLINE generates only a bar chart and duplicates the chart response axis (left axis) as the right response axis.

To specify a variable to plot, use the SUMVAR= option. If you do not specify a plot variable, GBARLINE uses the chart variable as the plot variable. For more information, see "About Response Variables" on page 931 and the option description for "SUMVAR=*plot-variable*" on page 971.

Description

The PLOT statement specifies one plot request. You can use multiple PLOT statements to generate multiple plots. The PLOT statement automatically does the following:

- scales the plot response (right) axis to include the maximum and minimum data values
- plots data points within the axis and connects them from left to right
- labels the plot response axis and displays each major tick mark value

You can use statement options to specify a plot variable, manipulate the plot response axis, modify the appearance of your graph, and describe catalog entries. You can use SYMBOL definitions to modify plot symbols for the data points, suppress the joining of data points, or specify other types of interpolation. For more information, see the "SYMBOL Statement" on page 470.

In addition, you can use global statements to add a legend, modify the axis, or add titles, footnotes, and notes to the plot.

About SYMBOL Definitions

How SAS/GRAPH Generates and Assigns SYMBOL Definitions

SYMBOL statements control the appearance of plot symbols and lines. They can specify the following attributes:

- the shape, size, and color of the plot symbols that mark the data points
- the plot line style, color, and width
- an interpolation method (either JOIN, NEEDLE, STEP, or NONE) for plotting data
- how missing values are treated in interpolation calculations

SYMBOL definitions are assigned either by default by the GBARLINE procedure or explicitly with a plot request.

If no SYMBOL definition is currently in effect, the GBARLINE procedure produces a join interpolation using the default plot symbol. For multiple PLOT statements where no SYMBOL statements were specified, the procedure rotates through the default symbols for the current device.

See the "SYMBOL Statement" on page 470 for a complete discussion of its features.

About Interpolation Methods

You can produce plot overlays such as step plot overlays by specifying interpolation methods with the SYMBOL statement. For PROC GBARLINE, you can use the SYMBOL statement to do the following tasks:

- connect data points with straight lines (JOIN)
- produce overlay plots with unconnected data points (NONE)
- use a step function to connect the data points (STEP)

For bar-line charts, points on the plot overlays are automatically connected by default, which is equivalent to specifying the JOIN interpolation method.

The "SYMBOL Statement" on page 470 describes the JOIN, STEP, and NONE interpolation methods.

Examples: GBARLINE Procedure

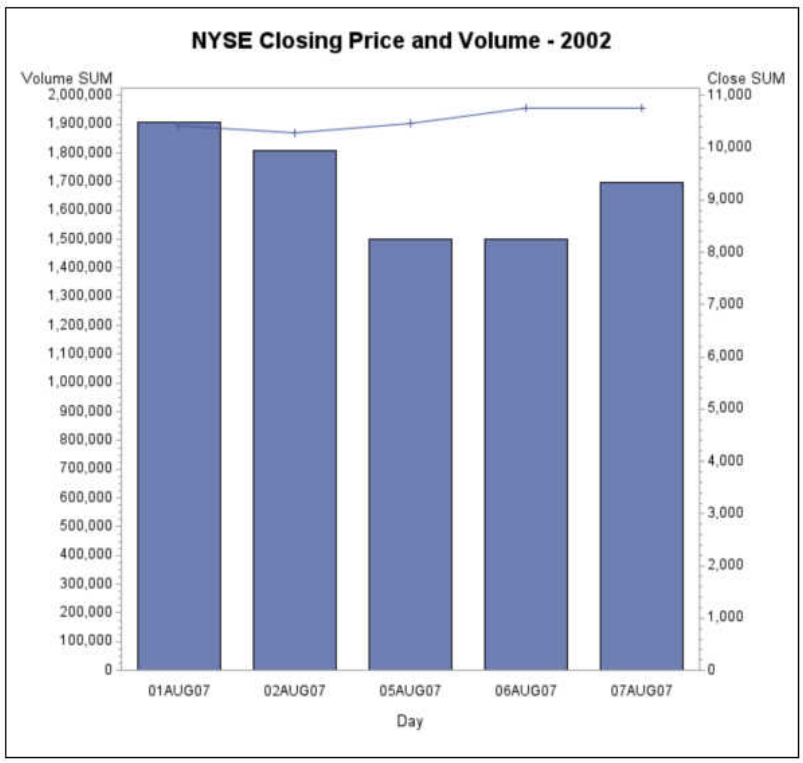
Example 1: Producing a Basic Bar-Line Chart

Features: BAR and PLOT statement options
SUMVAR=

Sample library member: GBLSTOCK

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces a basic bar-line chart showing the volume and closing price for each of five days of trading activity on the New York Stock Exchange. The vertical bars indicate the volume using the left (chart) response axis, and the line plot shows the closing price. This graph uses the statistical style.



Program

```
goptions reset=all border;

title1 "NYSE Closing Price and Volume - 2002";

data nyse;
   format Day date7.;
   format High Low Close comma12.;
   format Volume comma12.;
   input Day date7. High Low Close Volume;

datalines;
01AUG07 10478.76 10346.24 10426.91 1908809
02AUG07 11042.92 10298.44 10274.65 1807543
05AUG07 10498.22 10400.31 10456.43 1500656
06AUG07 10694.47 10636.32 10762.98 1498403
07AUG07 10801.12 10695.13 10759.48 1695602
run;

proc gbarline data=nyse;
bar day / discrete sumvar=volume space=4;
plot / sumvar=close;
run;
quit;
```

Program Description

Set the graphics environment. Some graphics options might override style attributes, so if you are using a style, specify the minimum graphic options needed by your graph.

```
goptions reset=all border;
```

Define the title.

```
title1 "NYSE Closing Price and Volume - 2002";
```

Create the data set NYSE. NYSE contains one observation for each of five workdays. Each observation includes the date, closing price, and volume.

```
data nyse;
   format Day date7.;
   format High Low Close comma12.;
   format Volume comma12.;
   input Day date7. High Low Close Volume;

datalines;
01AUG07 10478.76 10346.24 10426.91 1908809
02AUG07 11042.92 10298.44 10274.65 1807543
05AUG07 10498.22 10400.31 10456.43 1500656
06AUG07 10694.47 10636.32 10762.98 1498403
07AUG07 10801.12 10695.13 10759.48 1695602
run;
```

Produce the bar-line chart. The SUMVAR= option in the BAR statement specifies the variable whose values determine the height of the bars. The SUMVAR= option

in the PLOT statement specifies the variable whose values are used to calculate the overlay plot.

```
proc gbarline data=nyse;
bar day / discrete sumvar=volume space=4;
plot / sumvar=close;
run;
quit;
```

Example 2: Calculating Weighted Statistics

Features: BAR statement options
- AXIS=
- SUMVAR=

PLOT statement options
- AXIS=
- FREQ=
- SUMVAR=

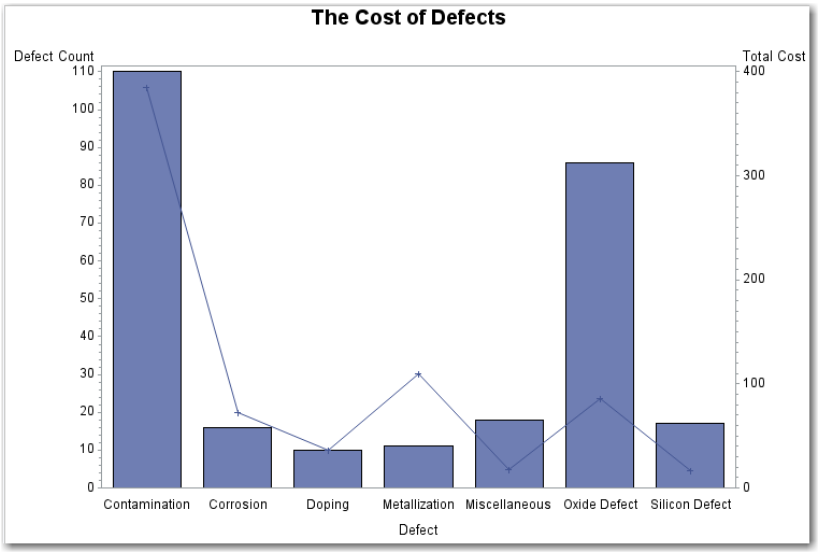
AXIS statement

Data set: SASHELP.FAILURE

Sample library member: GBLWTSTA

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses the FREQ= option to calculate weighted statistics for the line plot. During the manufacture of a metal-oxide semiconductor (MOS) capacitor, various defects and their frequencies were recorded.



Program

```
goptions reset=all border;

data myfail;
   set sashelp.failure;
   label Cause='Defect';
    select (Cause) ;
      when ("Contamination")  Cost=3.5;
      when ("Metallization")  Cost=10;
      when ("Oxide")          Cost=10.5;
      when ("Corrosion")      Cost=4.5;
      when ("Doping")         Cost=3.6;
      when ("Silicon")        Cost=5.4;
      otherwise               Cost=1.0;
    end;
run;

title1 "The Cost of Defects";

axis1 label=("Defect Count");
axis2 label=("Total Cost");

proc gbarline data=myfail;
   bar Cause / sumvar=Count axis=axis1;
   plot / sumvar=Count freq=cost axis=axis2;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

This graph uses the data set entitled FAILURE found in the SASHELP library. Each observation of the FAILURE data set contains the type of manufacturing defect and a count of how many times it occurred. Define data that associates a cost with each defect. Assign a label of 'Defect' in place of the variable name in the data set.

```
data myfail;
   set sashelp.failure;
   label Cause='Defect';
    select (Cause) ;
      when ("Contamination")  Cost=3.5;
      when ("Metallization")  Cost=10;
      when ("Oxide")          Cost=10.5;
      when ("Corrosion")      Cost=4.5;
      when ("Doping")         Cost=3.6;
      when ("Silicon")        Cost=5.4;
      otherwise               Cost=1.0;
    end;
run;
```

Define the title.

```
title1 "The Cost of Defects";
```

Define the labels for the axes.

```
axis1 label=("Defect Count");
axis2 label=("Total Cost");
```

Produce the bar-line chart. The SUMVAR= option in the BAR statement specifies the variable that determines the height of the bars. The SUMVAR= option in the PLOT statement specifies the plot variable, in this case the Cause of Failure, or Defect.. GBARLINE multiplies the value of the FREQ= variable (cost assigned) by the value of the COUNT variable, and uses the result to determine the plot points.

```
proc gbarline data=myfail;
   bar Cause / sumvar=Count axis=axis1;
   plot / sumvar=Count freq=cost axis=axis2;
run;
quit;
```

Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs

Features:

BAR statement options
- DISCRETE
- HTML=
- LEGEND=
- MAXIS=
- RAXIS=
- SUBGROUP=
- SUMVAR=

PLOT statement options
- AXIS=
- HTML=
- LEGEND=

Multiple PLOT statements
- SUMVAR=

ODS statement option
- STYLE=

AXIS statements

LEGEND statements

ODS HTML statement

SYMBOL statement

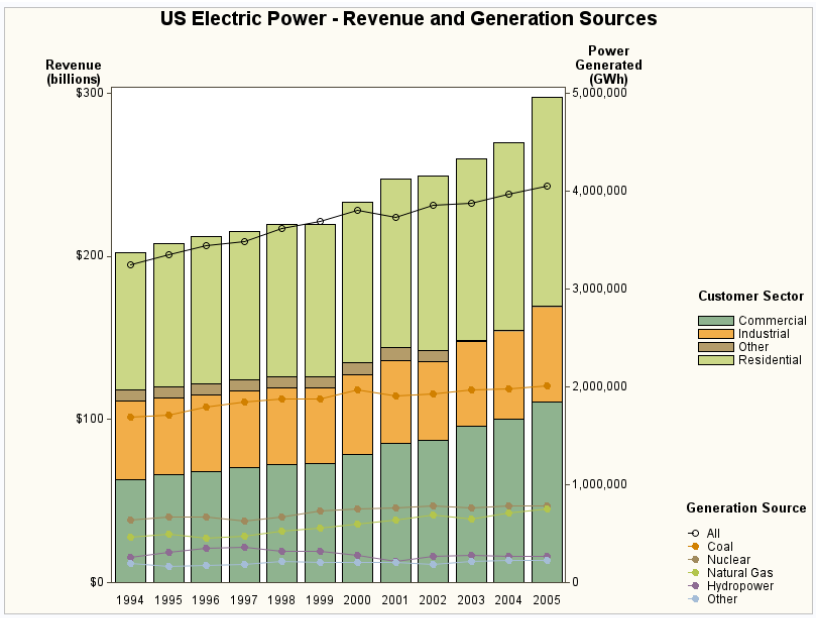
Data set: SASHELP.ELECTRIC

Sample library member: GBLPOWER

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This graph shows the total amount of power generated by six different energy sources in the US during the years 1994 to 2005. It also shows the revenue received from four different customer sectors during these same years. The power generated is graphed as a subgrouped bar chart. The chart variable is YEAR, and the subgroup variable is CUSTOMER, the customer sector. The program also specifies the DISCRETE option, so each year's data is graphed as a separate midpoint. The subgroups create a separate segment in the bar for each year, and the height of each bar represents the total revenue for that year for all customer sectors. The power generated from each energy source is plotted as six different line plots. Each of the six plot lines represents a different energy source.

Separate legends are created for the bar chart and the line plots. By specifying the LEGEND POSITION= option, the legend for the bar chart is displayed at the top middle of the graph. The legend for the plots is displayed at the bottom right of the graph. The colors used for everything except the plot lines is controlled by the style. The example specifies the Analysis style. This example defines data tip text for both the plot symbols and the bar chart segments. It defines drill-down URLs for the entries in the footnotes.



Link to Bar Data: USEIA Energy Customer Sectors
Link to Line Data: USEIA Energy Generation Sources

Program

```
filename odsout ".";

goptions reset=all border;

ods _all_ close;

ods html path=odsout file="electricpowerUS.html" style=analysis gtitle
   nogfootnote;

title1  "US Electric Power - Revenue and Generation Sources";

footnote1  j=l italic
```

```
           link="http://www.eia.doe.gov/cneaf/electricity/epa/
epat7p3.html"
           "Link to Bar Data: USEIA Energy Customer Sectors";

footnote2 j=l italic
          link="http://www.eia.doe.gov/cneaf/electricity/epa/
epat1p1.html"
          "Link to Line Data: USEIA Energy Generation Sources" ;

axis1 label=(j=c "Revenue" j=c "(billions)") minor=none; /* left */
axis2 label=(j=c "Power" j=c "Generated" j=c "(GWh)") minor=none; /*
right */
axis3 label=none; /* bottom */

/* Bar legend */
legend1 position=(middle right outside)  across=1
        label=(position=(top )  j=l "Customer Sector");
/* Line plot legend */
legend2 position=(bottom right outside) across=1 repeat=1
        label=(position=(top) j=l  "Generation Source") ;

symbol1 c=black value=circle;
symbol2 value=dot;

proc gbarline data=sashelp.electric;
   bar year / discrete sumvar=Revenue subgroup=Customer
              raxis=axis1 maxis=axis3 legend=legend1
              html=revtip name="US_Electric_Power"
              des="Chart of US Electricity Generation Sources and
Consumers";

   plot / sumvar=AllPower   html=alltip legend=legend2 axis=axis2;
   plot / sumvar=Coal       html=coaltip;
   plot / sumvar=Nuclear    html=nuketip;
   plot / sumvar=NaturalGas html=gastip;
   plot / sumvar=Hydro      html=hydrotip;
   plot / sumvar=Other      html=othertip;
run;
quit;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Create a file reference for the output. The current working directory is specified in this example.

```
filename odsout ".";
```

Set the graphics environment.

```
goptions reset=all border;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Set the HTML options. The PATH= option specifies the file reference that was defined previously for the output. The FILE= option specifies a name for the HTML output file. The GTITLE option causes the title to be rendered as part of the graph image instead of being created by the HTML code as text. Alternatively, the NOGFOOTNOTE option causes the footnote to be created by the HTML file as text instead of being rendered as an image with the rest of the graph. Notice that, as a result, the TITLE appears within the graph frame, but the footnotes appear outside the frame.

```
ods html path=odsout file="electricpowerUS.html" style=analysis gtitle
   nogfootnote;
```

Define the title and footnotes. The LINK= option in the FOOTNOTE statement defines drill-down URLs for the source of the information.

```
title1  "US Electric Power - Revenue and Generation Sources";

footnote1  j=l italic
           link="http://www.eia.doe.gov/cneaf/electricity/epa/
epat7p3.html"
           "Link to Bar Data: USEIA Energy Customer Sectors";

footnote2 j=l italic
          link="http://www.eia.doe.gov/cneaf/electricity/epa/
epat1p1.html"
          "Link to Line Data: USEIA Energy Generation Sources" ;
```

Define the labels for the axes. The AXIS1 statement defines the axis properties for the bar response (left) axis. The AXIS2 statement defines the properties for the plot response (right) axis. The AXIS3 statement is used to suppress the default label on the midpoint axis.

```
axis1 label=(j=c "Revenue" j=c "(billions)") minor=none; /* left */
axis2 label=(j=c "Power" j=c "Generated" j=c "(GWh)") minor=none; /*
right */
axis3 label=none; /* bottom */
```

Specify options for the bar and plot legends. Use different LEGEND statements and position the legends in different places for the bar chart and the overlay plots. This causes GBARLINE to produce two separate legends instead of combining the legends into one.

```
/* Bar legend */
legend1 position=(middle right outside)  across=1
        label=(position=(top )  j=l "Customer Sector");
/* Line plot legend */
legend2 position=(bottom right outside) across=1 repeat=1
        label=(position=(top) j=l  "Generation Source") ;
```

Define the plot symbols.

```
symbol1 c=black value=circle;
symbol2 value=dot;
```

Produce the bar-line chart. This graph uses the data set entitled ELECTRIC found in the SASHELP library. The SUMVAR= option in the BAR statement specifies the variable that determines the height of the bars. The SUMVAR= option in the PLOT statement specifies the plot variable. The HTML= options associate data tip text with the bars and plot points.

```
proc gbarline data=sashelp.electric;
   bar year / discrete sumvar=Revenue subgroup=Customer
```

```
                raxis=axis1 maxis=axis3 legend=legend1
                html=revtip name="US_Electric_Power"
                des="Chart of US Electricity Generation Sources and
Consumers";

   plot / sumvar=AllPower   html=alltip legend=legend2 axis=axis2;
   plot / sumvar=Coal       html=coaltip;
   plot / sumvar=Nuclear    html=nuketip;
   plot / sumvar=NaturalGas html=gastip;
   plot / sumvar=Hydro      html=hydrotip;
   plot / sumvar=Other      html=othertip;
run;
quit;
```

Close ODS HTML. Closing ODS HTML closes the HTML output file.

```
ods html close;
```

Open ODS HTML; Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```
ods html; /* Not required in SAS Studio */
```

Chapter 36

GCHART Procedure

Overview: GCHART Procedure

About the GCHART Procedure

The GCHART procedure produces six types of charts: block charts, horizontal and vertical bar charts, pie and donut charts, and star charts. These charts graphically represent the value of a statistic calculated for one or more variables in an input SAS data set. The charted variables can be either numeric or character.

The procedure calculates these statistics:

- frequency or cumulative frequency counts
- percentages or cumulative percentages
- sums
- means

Use the GCHART procedure to do the following tasks:

- display and compare exact and relative magnitudes
- examine the contribution of parts to the whole
- analyze where data are out of balance

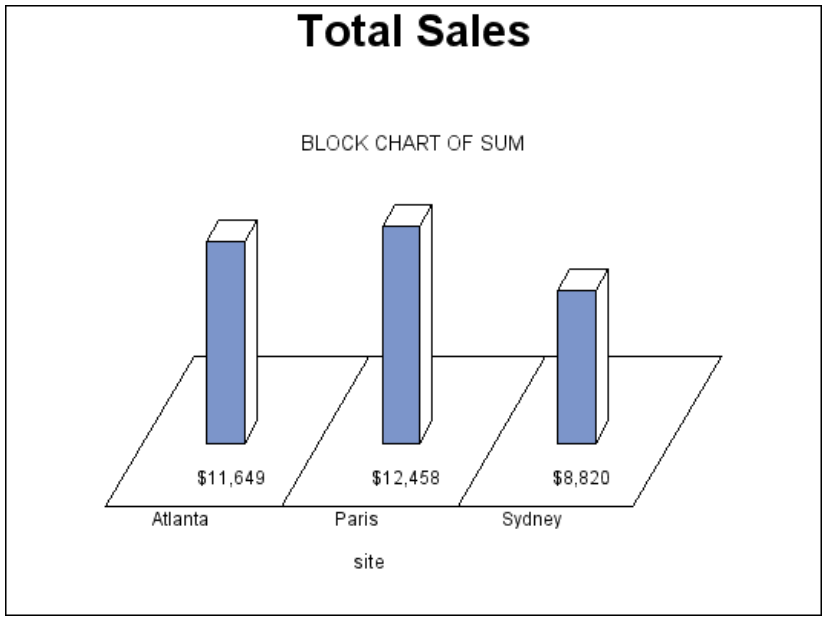
About Block Charts

Block charts display the relative magnitude of data with blocks of varying height, each set in a square that represents a category of data (midpoint). Because block charts do not use axes, they are most useful when the relative magnitude of the blocks is more significant than the exact magnitude of any particular block.

Figure 36.109 on page 985 shows a simple block chart of total sales for three manufacturing sites. Each site is a midpoint and occupies one square. The name of the site (the midpoint value) is printed below the square. By default, midpoint values are arranged in ascending order from left to right. The label below the midpoint grid names the chart variable.

Sales for the site (the chart statistic) are represented by the height of the block; sales amount (the formatted statistic value) is printed below the block. The heading above the blocks describes the type of statistic, in this case SUM.

Figure 36.1 Block Chart (GCHBKSUM)



The program for this chart is in "Example 1: Specifying the Sum Statistic in a Block Chart" on page 1093. For more information about producing block charts, see the "BLOCK Statement" on page 1002.

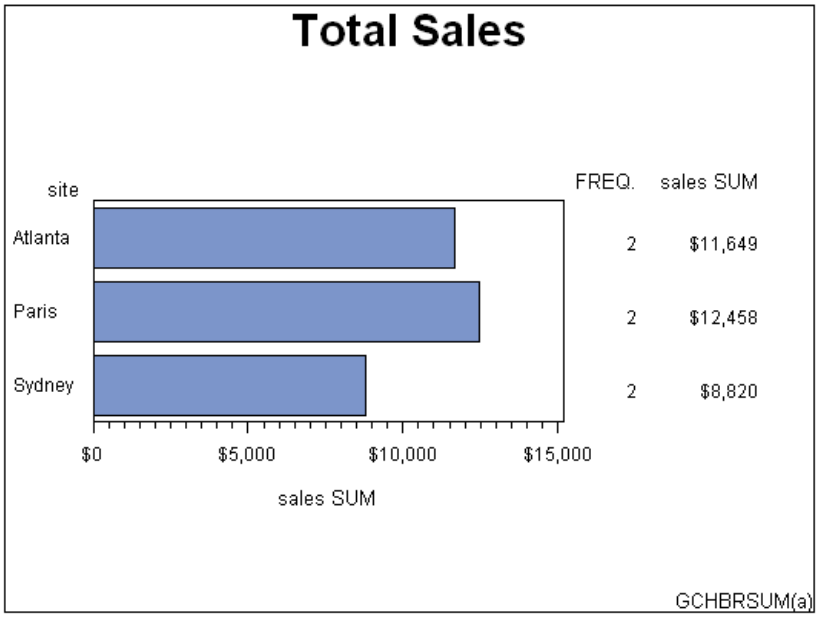
About Bar Charts

Horizontal and vertical bar charts display the magnitude of data with bars, each of which represents a category of data (midpoint). The length (or height) of the bars represents the value of the chart statistic for the corresponding midpoint. Both horizontal and vertical bar charts can be either two-dimensional or three-dimensional shapes, depending on which statement you choose.

Figure 36.110 on page 986 shows a simple two-dimensional, horizontal bar chart of total sales for three manufacturing sites. Each site is a midpoint and is displayed as a bar. The name of the site (the midpoint value) is printed on the midpoint axis beside the bar. By default, midpoint values are arranged in ascending alphabetical or numeric order from top to bottom of the chart and labeled with the name or label of the chart variable.

The chart statistics, in this case total sales for each site, are represented by the length of the bars. The response axis displays the scale of values for the chart statistic. The table of statistics to the right of the bars displays the statistic for each bar. Both a column in the table and the response axis are labeled with the name of the summary variable and the type of statistic.

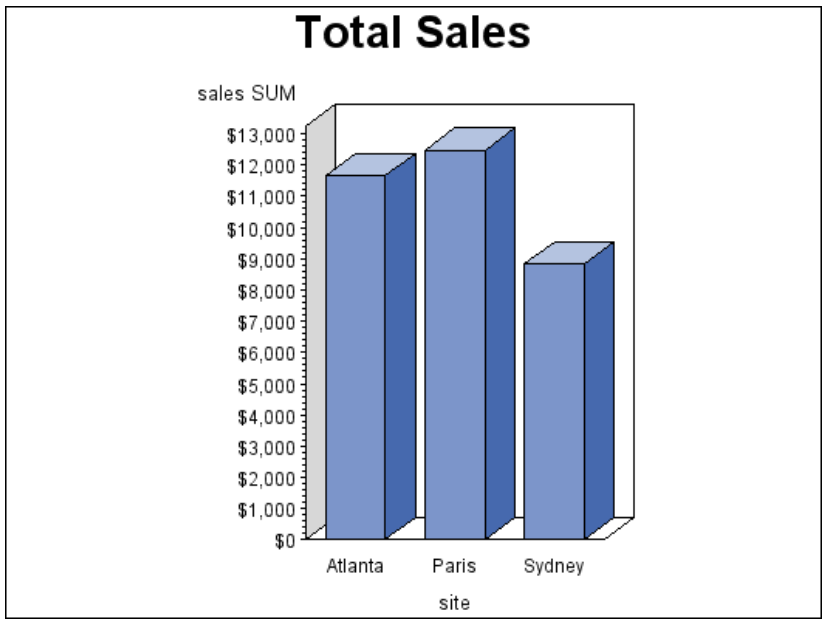
Figure 36.2 Horizontal Bar Chart (GCHBRSUM (a))



The program for this chart is "Example 3: Specifying the Sum Statistic in Bar Charts" on page 1098.

Figure 36.111 on page 986 shows the same data presented as a three-dimensional, vertical bar chart. The two types of bar charts have essentially the same characteristics except for where they display statistical values. By default, horizontal bar charts display a table of statistic values to the right of the bars. You can specify that vertical bar charts display the statistic value above or inside each bar.

Figure 36.3 Vertical (Three-Dimensional) Bar Chart (GCHBRSUM (b))



The program for this chart is “Example 3: Specifying the Sum Statistic in Bar Charts” on page 1098. For more information about producing horizontal and vertical bar charts, see “HBAR, HBAR3D, VBAR, and VBAR3D Statement” on page 1016.

About Pie, Detail Pie, and Donut Charts

Pie and donut charts represent the relative contribution of parts to the whole. They display data as wedge-shaped “slices” of a circle (either a “pie” or “donut”), either in two- or three-dimensional form. Each slice represents a category of data (midpoint). The size of each slice (length of the arc) represents the contribution of the corresponding midpoint to the total chart statistic. Detail pie charts are pie charts with a second pie overlay that shows additional detail about the data that contributes to each of the outer pie's slices. Donut charts look like pie charts except that they have a hole in the middle in which you can place text.

Figure 36.112 on page 987 shows a pie chart of total sales for three manufacturing sites. Each site is a midpoint and is displayed as a slice. By default, the slices are ordered alphabetically, by the midpoint name and counterclockwise beginning at the three o'clock position.

Sales for the site (the chart statistic) are represented by the size of the slice. Both the sales amount (the formatted value of the chart statistic) and the name of the site (the midpoint value) are printed outside of the slice. You can also label pie slices with the percentage of the total statistic value that they represent. The heading above the pie describes the type of statistic (SUM), and names the summary variable (SALES) and the chart variable (SITE).

Figure 36.4 *Pie Chart (GCHPISUM (a))*

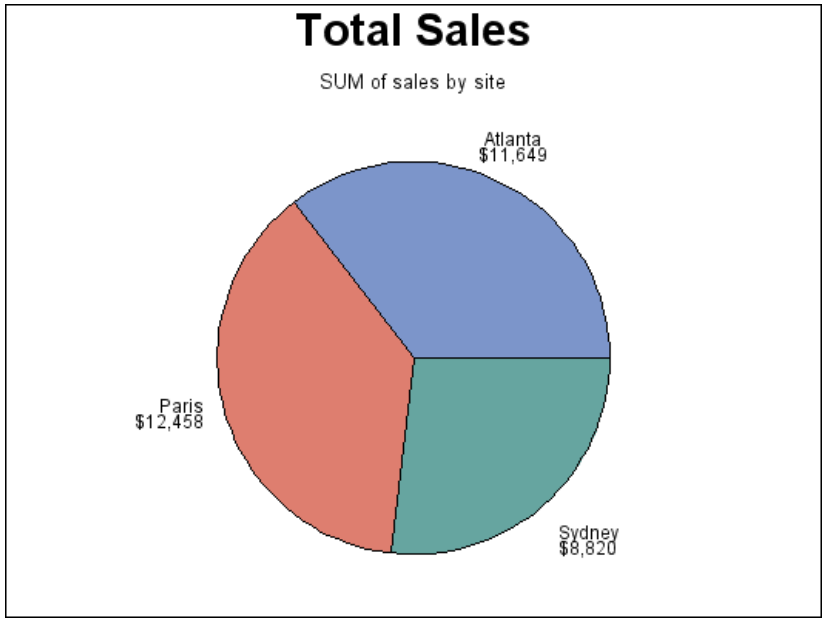


Figure 36.113 on page 988 shows the three-dimensional version of the same pie chart. This version features the exploded slice.

Figure 36.5 Three-Dimensional Pie Chart (GCHPISUM (b))

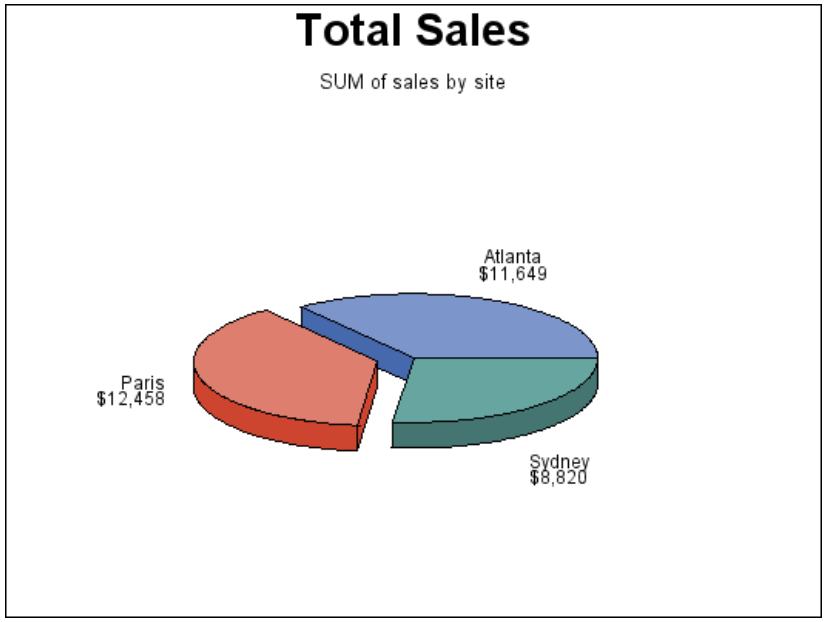
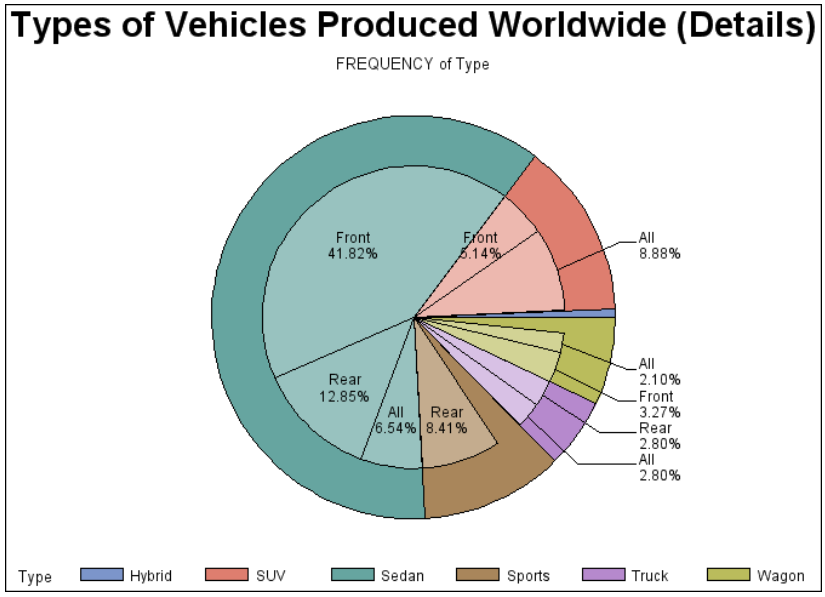


Figure 36.114 on page 988 shows a detail pie chart generated from the same data.

Figure 36.6 Detail Pie Chart (GCHDTPIE)



The programs for these charts are in "Example 7: Specifying the Sum Statistic for a Pie Chart" on page 1109 and "Example 13: Creating a Detail Pie Chart" on page 1123. For more information about producing pie or donut charts, see "PIE, PIE3D, and DONUT Statement" on page 1052.

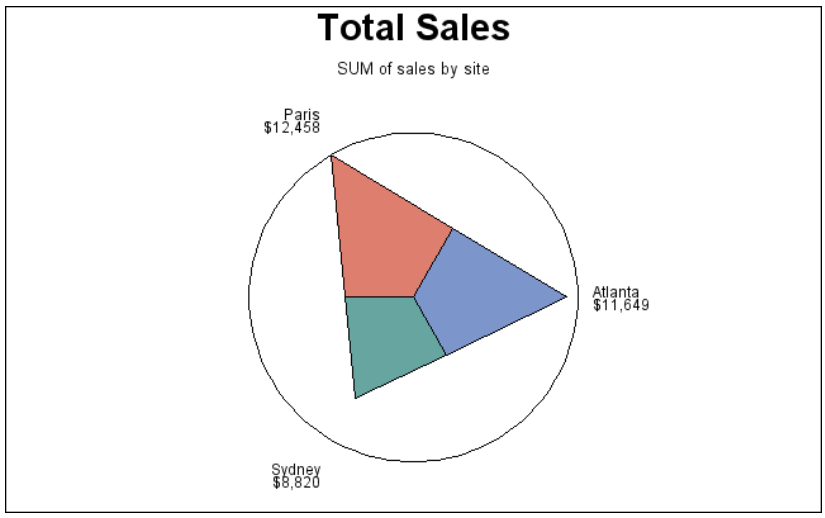
About Star Charts

Star charts display data as lines (“spines”) radiating from the center of a circle toward the perimeter. Each spine represents a category of data (midpoint). The length of a spine represents the magnitude of the chart statistic for that midpoint starting at the center of the circle, which by default represents 0. The radius of the circle is the length of the longest spine (greatest statistic value) in the chart. Instead of spines, star charts can also display the chart statistic as slices, which are enclosed areas formed by connecting the ends of the spines.

Figure 36.115 on page 989 shows the total sales for the three manufacturing sites as a star chart. Each site is a midpoint and is displayed as a spine. By default the ends of the spines are connected and they are ordered counterclockwise beginning at the three o'clock position.

Sales for the site (the chart statistic) are represented by the length of the spine. Both the sales amount (the formatted statistic value) and the name of the site (the midpoint value) are printed outside of the star chart. You can also label star charts with the percentage of the total statistic value that they represent. The heading above the chart describes the type of statistic (SUM), and names the summary variable (SALES) and the chart variable (SITE).

Figure 36.7 *Star Chart (GCHSTSUM)*



The program for this chart is “Example 11: Specifying the Sum Statistic in a Star Chart” on page 1118. For more information about producing star charts, see “STAR Statement” on page 1077. For an alternative way of producing similar types of charts, see Chapter 43, “GRADAR Procedure,” on page 1375.

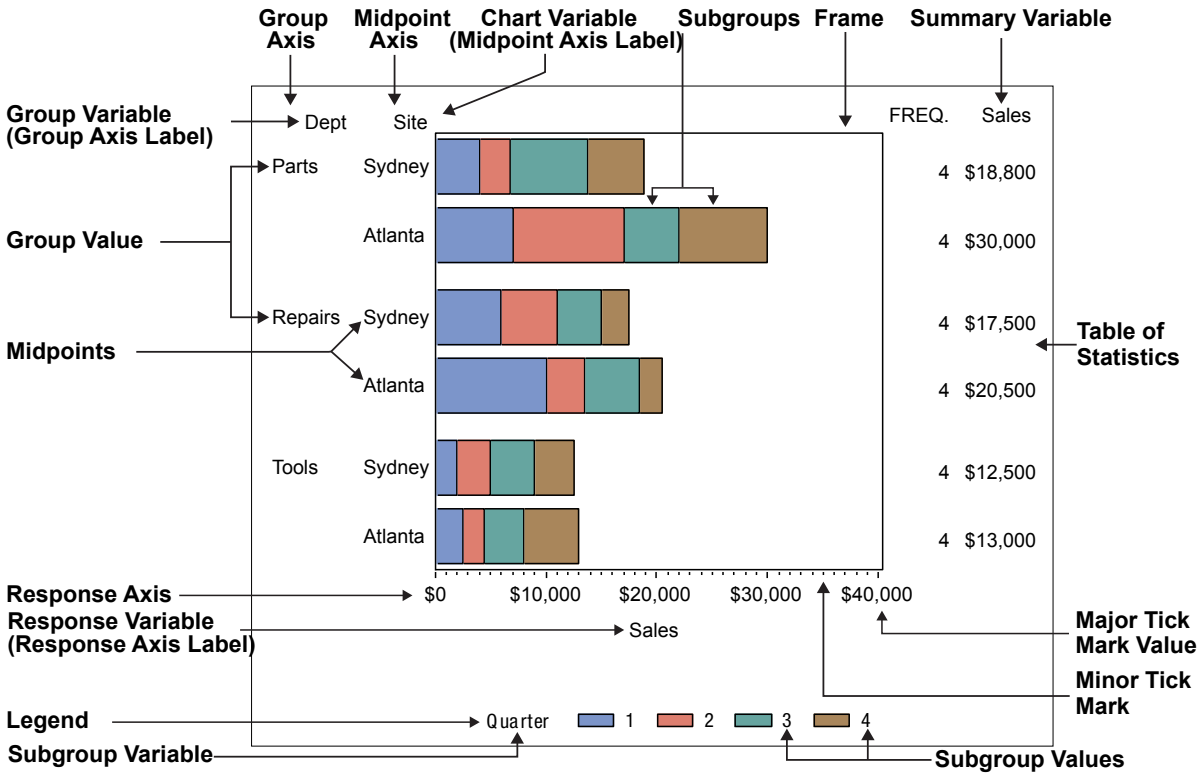
Concepts: GCHART Procedure

Chart Terminology

The GCHART procedure produces charts based on the values of a *chart variable*. These values are represented by a set of *midpoints*. The chart itself displays information about the chart variable in the form of *chart statistics*.

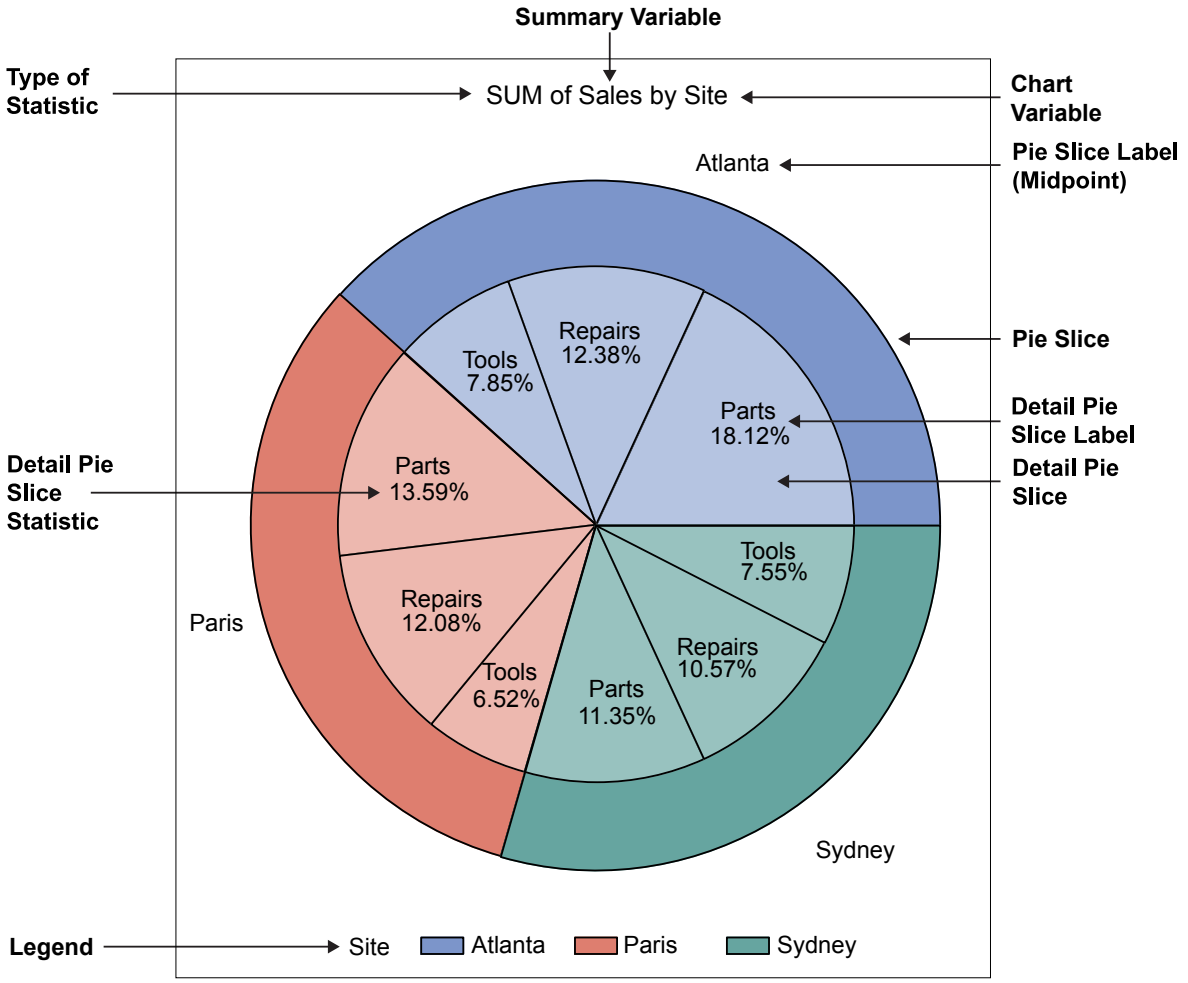
Figure 36.116 on page 990 and Figure 36.117 on page 991 illustrate these terms as well as other terms used with the GCHART procedure.

Figure 36.8 *Terms Used with Bar Charts*



Bar charts have at least two axes: a midpoint axis that shows the categories of data, and a response axis that displays the scale of values for the chart statistic. By default, the response axis is divided into evenly spaced intervals identified with major tick marks that are labeled with the corresponding statistic value. Minor tick marks are evenly distributed between the major tick marks. Each axis is labeled with the chart variable name or label. The response axis is also labeled with the statistic type.

Figure 36.9 *Terms Used with Pie and Donut Charts*



Pie charts show statistics based on values of a variable called the chart variable. Generally, the values of the chart variable are represented by the slices in the chart. Beside each pie slice a number (or character string) appears that identifies the value or range of values assigned to that slice by the GCHART procedure. This number (or character string) is known as the *midpoint* for that slice. The statistic value for each midpoint is displayed beneath the midpoint. Each pie slice represents a different value of a given variable (the chart variable). Because the pie chart forms a circle of 360 degrees, each slice represents a percentage of degrees of the circle. The number of degrees created by each slice represents the statistic value for the midpoint.

Understanding Chart Variables

About Chart Variables

The *chart variable* is the variable in the input data set whose values determine the categories of data represented by the bars, blocks, slices, or spines. The chart variable generates the midpoints to which each observation in the data set contributes.

The chart variable can be either character or numeric. Character chart variables contain character values, which are always discrete. Numeric chart variables fall into two categories: discrete and continuous.

Note: You can apply a format that converts multiple values or a range of values to a single formatted value. In this case the GCHART procedure produces a single midpoint for that single formatted value.

- *Discrete variables* contain a finite number of specific numeric values that are to be represented on the chart. For example, a variable that contains years, such as 1984 or 2001, is a discrete variable.
- *Continuous variables* contain a range of numeric values that are to be represented on the chart. For example, a variable of temperature data that contains real values between 0 and 212 is a continuous variable.

Numeric chart variables are always treated as continuous variables unless the DISCRETE option is used in the action statement, or, unless a format is used to group ranges of values. In most cases it is a good idea to specify the DISCRETE option when using date values.

Missing Values

By default, the GCHART procedure ignores missing midpoint values for the chart variable. If you specify the MISSING option, then missing values are treated as a valid midpoint and are included on the chart. Missing values for the group and subgroup variables are always treated as valid groups and subgroups.

When the value of the variable that is specified in the FREQ= option is missing, 0, or negative, the observation is excluded from the calculation of the chart statistic.

When the value of the variable specified in the SUMVAR= option is missing, the observation is excluded from the calculation of the chart statistic.

Understanding Midpoints

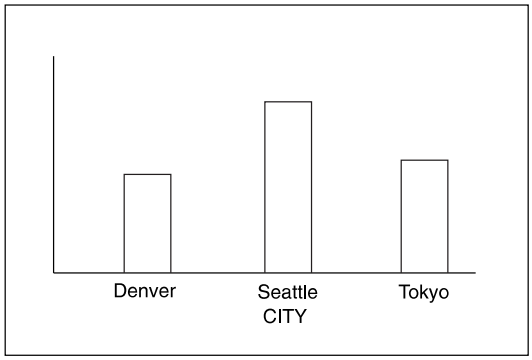
About Midpoints

Midpoints are the values of the chart variable that identify categories of data. By default, midpoints are selected or calculated by the procedure. The way the procedure handles the midpoints depends on whether the values of the chart variable are character, discrete numeric, or continuous numeric.

Character Values

A character chart variable generates a midpoint for each unique value of the variable. For example, if the chart variable CITY contains the names of three different cities, each city is a midpoint, resulting in three midpoints for the chart:

Figure 36.10 *Character Midpoints*



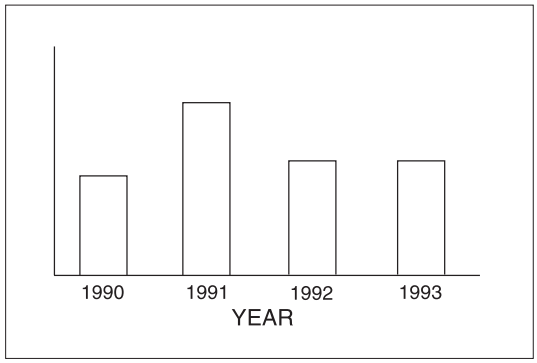
(In pie charts, midpoint values that compose a small percentage of the total for the chart might be placed in the OTHER slice and do not produce a separate midpoint.)

By default, character midpoints are arranged in alphabetic order. If a character variable has an associated format, the values are arranged in order of the formatted values.

Discrete Numeric Values

A numeric chart variable used with the DISCRETE option generates a midpoint for each unique value of the chart variable. For example, the numeric variable YEAR used with the DISCRETE option produces one midpoint for each year:

Figure 36.11 *Discrete Numeric Midpoints*



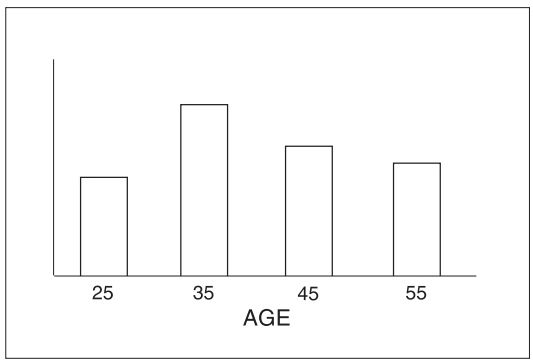
By default, numeric midpoints are arranged in ascending order. The DISCRETE option is very useful for working with dates and numeric values with text user-defined formats. If the numeric variable has an associated format, each formatted value generates a separate midpoint. Formatted numeric variables are arranged in ascending order according to their unformatted numeric values.

Continuous Numeric Values

A continuous numeric variable generates midpoints that represent ranges of values. By default, the GCHART procedure determines the ranges, calculates the median value of each range, and displays the appropriate median value at each midpoint on the chart. A value that falls exactly halfway between two midpoints is placed in the higher range.

For example, the numeric variable AGE produces four midpoints, each of which represents a ten-year age range; the median value of the range is displayed at each midpoint:

Figure 36.12 *Continuous Numeric Midpoints*



By default, midpoints of ranges are arranged in ascending order.

Selecting and Ordering Midpoints

For character or discrete numeric values, you can use the MIDPOINTS= option to rearrange the midpoints or to exclude midpoints from the chart. For example, to change the default alphabetic order of the midpoints in Figure 36.118 on page 993, specify the following:

```
midpoints="Tokyo" "Denver" "Seattle"
```

To exclude the midpoint for Denver, specify the following:

```
midpoints="Tokyo" "Seattle"
```

In this case, values excluded by the option are not included in the calculation of the chart statistic.

You can order or select discrete numeric midpoint values just as you do character values, but you omit the quotation marks when specifying numeric values.

For continuous numeric variables, use the LEVELS= or MIDPOINTS= option to

- change the number of midpoints
- control the range of values each midpoint represents
- change the order of the midpoints.

To control the range of values each midpoint represents, use the MIDPOINTS= option to specify the median value of each range. For example, to select the ranges 20–29, 30–39, and 40–49, specify the following:

```
midpoints=25 35 45
```

Alternatively, to select the number of midpoints that you want and let the procedure calculate the ranges and medians, use the LEVELS= option.

After you specify the number of midpoints that you want, the range for each midpoint is calculated automatically, using the algorithm in Terrell and Scott (1985).

You can also use formats to control the ranges of continuous numeric variables, but in that case the values are no longer continuous but discrete.

Note: You cannot use the MIDPOINTS= option to exclude continuous numeric values from the chart. Values below or above the ranges specified by the option are automatically included in the first and last midpoints, respectively. To exclude continuous numeric values from a chart, use a WHERE statement in a DATA step or the WHERE= data set option.

See also the description of the LEVELS= and MIDPOINTS= options for the appropriate statement.

About Chart Statistics

Definition: Chart Statistics

The *chart statistic* is the statistical value calculated for the chart variable and represented by each block, bar, or slice. The GCHART procedure calculates six chart statistics; the default statistic is frequency.

The examples given in the descriptions of these statistics assume a data set with two variables, CITY and SALES. The values of CITY are **Denver**, **Seattle**, and **Tokyo**. There are 21 observations: seven for Denver, nine for Seattle, and five for Tokyo.

Frequency

The frequency statistic is the total number of observations in the data set for each midpoint. For example, seven observations of the chart variable, CITY, contain the value **Denver**, so the frequency for the **Denver** midpoint is 7.

Cumulative Frequency

The cumulative frequency statistic adds the frequency for the current midpoint to the frequency of all of the preceding midpoints. For example, the frequency for the **Denver** midpoint is 7, and the frequency for the next midpoint, **Seattle**, is 9, so the cumulative frequency for **Seattle** is 16.

You cannot request cumulative frequency with the DONUT, PIE, PIE3D, or STAR statements.

Percentage

The percentage statistic is calculated by dividing the frequency for each midpoint by the total frequency count for all midpoints in the chart or group and multiplying it by 100. For example, the frequency count for the **Denver** midpoint is 7 and the total frequency count for the chart is 21, so the percentage statistic for **Denver** is 33.3%.

Cumulative Percentage

The cumulative percentage statistic adds the percentage for the current midpoint to the percentage for all of the preceding midpoints in the chart or group. For example, the percentage for the `Denver` midpoint is 33.3, and the percentage for the next midpoint, `Seattle`, is 42.9, so the cumulative percentage for `Seattle` is 76.2.

You cannot request cumulative percentage with the DONUT, PIE, PIE3D, or STAR statements.

Sum

The sum statistic is the total of the values for the SUMVAR= variable for each midpoint. For example, if you specify SUMVAR=SALES, and the values of the SALES variable for the seven `Denver` observations are `8734`, `982`, `1504`, `3207`, `4502`, `624`, and `918`, then the sum statistic for the `Denver` midpoint is 20,471.

You must use the SUMVAR= option to specify the variable for which you want the sum statistic.

Mean

The mean statistic is the average of the values for the SUMVAR= variable for each midpoint. For example, if TYPE=MEAN and SUMVAR=SALES, the mean statistic for the `Denver` midpoint is 2924.42.

You must use the SUMVAR= option to specify the variable for which you want the mean statistic.

Calculating Weighted Statistics

By default, each observation is counted only once in the calculation of the chart statistic. To calculate weighted statistics in which an observation can be counted more than once, use the FREQ= option. This option identifies a variable whose values are used as a multiplier for the observation in the calculation of the statistic. If the value of the FREQ= variable is missing, 0, or negative, the observation is excluded from the calculation.

If you use the SUMVAR= option, then the SUMVAR= variable value for an observation is multiplied by the FREQ= variable value for that observation when calculating the chart statistic.

For example, to use a variable called COUNT to produce weighted statistics, assign FREQ=COUNT. If you also assign the variable HEIGHT to the SUMVAR= option, then the following table shows how the values of COUNT and HEIGHT would affect the statistic calculation:

Value of COUNT	Value of HEIGHT	Number of Times the Observation is Used	Value Used for HEIGHT
1	55	1	55
5	65	5	325
.	63	0	-
-3	60	0	-

By default, the percentage and cumulative percentage statistics are calculated based on the frequency. If you want to chart a percentage or cumulative percentage based on a sum, you can use the FREQ= option to specify a variable to use for the "sum" calculation. You can also specify the PCT statistic, as shown in this example:

```
freq=count type=pct
```

Because the variable that is used by the FREQ= option determines the number of times an observation is counted, the value of COUNT is the equivalent of the sum statistic.

See also the descriptions of the TYPE=, SUMVAR=, and FREQ= options for the action statements.

About Patterns

Pattern Categories

When a chart needs one or more patterns, the procedure uses either one of the following:

- default patterns and outlines that are automatically generated by SAS/GRAPH
- patterns, colors, outlines, and images that are defined by PATTERN statements, graphics options, and procedure options

The following sections summarize pattern behavior for the GCHART procedure. For more information, see the "PATTERN Statement" on page 455.

Default Patterns and Outlines

The GCHART procedure uses default patterns and outlines when you do not do the following:

- specify *any* PATTERN statements
- use the CPATTERN= graphics option

- use the COLORS= graphics options
- use the COUTLINE= option in the action statement

The default patterns, colors, and outlines are generated from the current style. If all of the above conditions are true, and the GSTYLE option is in effect, then the GCHART procedure does the following:

- selects the default fill, which is always solid, and rotates it through the color list of the current style, generating one solid pattern for each color. If the first color in the style's color list is black (or white), the procedure skips that color and begins generating patterns with the next color.
- uses the style outline color to outline every patterned area.

If all of the above conditions are true, and the NOGSTYLE option is specified, then the GCHART procedure does the following:

- selects the first default fill, which is always solid, and rotates it through the device's color list, generating one solid pattern for each color. If the first color in the device's color list is black (or white), the procedure skips that color and begins generating patterns with the next color.
- uses the foreground color to outline every patterned area.
- if the procedure needs additional patterns, GCHART selects the next default pattern fill that is appropriate to the type of chart. GCHART then rotates the pattern fill through the color on the list, skipping the foreground color as before. The procedure continues in this manner until it has generated enough patterns for the chart.

Changing any of the above conditions changes or overrides the default behavior:

- If you specify a color list with the COLORS= option in a GOPTIONS statement. If the list contains more than one color, the procedure produces a solid pattern through that list, using every color, even if the foreground color is black (or white). The default outline color remains the style outline color.
- If you specify either COLORS=(*one-color*) or the CPATTERN= graphics option, the default fill pattern changes from solid to the list of appropriate hatch patterns. The procedure uses the specified color to generate one pattern definition for each hatch pattern in the list. The default outline color remains the style outline color. (The Java and ActiveX devices do not support hatch patterns.)

For a description of these graphics options, see “Graphics Options and Device Parameters Dictionary” on page 577.

User-Defined Patterns, Outlines, and Images

You can use PATTERN statements to specify patterns, including color or fill type or both. You can also specify images to fill the bars of two-dimensional bar charts. For complete information about all patterns, see the “PATTERN Statement” on page 455. See also the section on controlling patterns and colors for each chart type.

When you use PATTERN statements, the procedure uses the specified patterns until all of the PATTERN definitions that they generate have been used. Then, if more patterns are required, it returns to the default pattern rotation.

To change the outline color of any pattern, whether it is a default or user-defined pattern, use the COUTLINE= option in the action statement that generates the chart.

The GCHART procedure uses the device colors for outlines in block, bar, pie, and star charts regardless of the ODS Style specified. Similarly, the color of the midpoint grid lines in particular charts depend on the device specified. For example, when you use ODS Style HighContrast with its black background, change from a black or dark-colored outline or axis color to a more visible color. Use the appropriate options available in the action statement that generates the chart. The COUTLINE= option changes the outline color of a block, bar, pie, or star chart. The CAXIS= option changes the axis color in a block or bar chart.

Two-dimensional bar charts created with the HBAR and VBAR statements can use the PATTERN statement to fill specified bars with specified images. For details, see the option description for "IMAGE=*fileref* | "*external-file*"" on page 456. Other means of including images in charts include adding background images to bar charts. The IBACK= option specifies an image file that fills the entire area behind the graph. The IFRAME= option specifies an image file that fills the area within the axes of the graph.

For additional information, including a list of recognized image file types, see "Image File Types Supported by SAS/GRAPH" on page 384.

Syntax: GCHART Procedure

Requirement: At least one BLOCK, HBAR, HBAR3D, VBAR, VBAR3D, PIE, PIE3D, DONUT, or STAR statement is required.

Supported Statements: AXIS, FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

Supports: RUN-group processing

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, "Plotting a Cloud Analytic Services (CAS) In-Memory Table," on page 153.

The procedure can include the SAS/GRAPH statements BY on page 424 and NOTE on page 507, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See "SAS/GRAPH Statements" on page 397 and *SAS DATA Step Statements: Reference* for more information.

Tip: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, `ODS PDF; procedure-code; ODS PDF CLOSE;`). See Chapter 7, "Using Run-Group Processing," on page 75 for more information.

PROC GCHART DATA=*input-data-set*
< ANNOTATE=*Annotate-data-set* >
< GOUT=<*libref.*>*output-catalog* >
< IMAGEMAP=*output-data-set* >;

BLOCK *chart-variable(s)* < / *options* >;

HBAR | HBAR3D | VBAR | VBAR3D *chart-variable(s)* < / *options* >;

PIE | PIE3D | DONUT *chart-variable(s)* < / *options* >;
STAR *chart-variable(s)* < / *options* >;

PROC GCHART Statement

Identifies the data set containing the chart variables. Can specify annotation and an output catalog.

Requirement: An input data set is required. If not specified, an input data set must have been created before issuing the PROC GCHART statement.

Syntax

PROC GCHART DATA=*input-data-set*
< ANNOTATE=*Annotate-data-set* >
< GOUT=<*libref.*>*output-catalog* >
< IMAGEMAP=*output-data-set* >;

Required Argument

DATA=*input-data-set*
specifies the SAS data set that contains the variable or variables to chart. If not specified, an input data set must have been created before issuing the PROC GCHART statement.

Default By default, the procedure uses the most recently created SAS data set.

See "The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

"Understanding Chart Variables" on page 992

Optional Arguments

PROC GCHART statement options affect all graphs produced by the procedure.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

Note Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block, pie, donut, or star charts.

See Chapter 28, "Using Annotate Data Sets," on page 721

GOUT=<*libref*.>*output-catalog*
specifies the SAS catalog in which to save the graphics output that is produced by the GCHART procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

Restriction Not supported by Java or ActiveX

See "Specifying the Catalog Name and Entry Name for Your GRSEGs " on page 138

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255 and "Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" on page 262

BLOCK Statement

Creates block charts in which the height of the blocks represents the value of the chart statistic for each category of data.

Requirement:	At least one chart variable is required.
Supported Statements:	FOOTNOTE, LEGEND, PATTERN, TITLE
Supports:	Drill-down functionality

Syntax

BLOCK *chart-variable(s)* < / *options* >;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GCHART procedure.

BLOCKMAX=*max-value*
specifies the chart statistic value of the tallest block on the chart.

CAXIS=*grid-color*

specifies the color for the midpoint grid.

COUTLINE=*block-outline-color* | SAME
outlines all blocks or all block segments and legend values in the subgroup legend (if it appears) using the specified color.

CTEXT=*text-color*
specifies a color for all text on the grid and legend.

LEGEND=LEGEND<1 …99>
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.

NOHEADING
suppresses the heading describing the type of statistic.

NOLEGEND
suppresses the legend automatically generated by the SUBGROUP= option.

PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP
specifies how fill patterns are assigned.

WOUTLINE=*block-outline-width*
specifies the width of the block outline in pixels.

Catalog entry description options

DESCRIPTION="*description*"
specifies a description of the output.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Midpoint options

DISCRETE
treats a numeric chart variable as a discrete variable rather than as a continuous variable.

GROUP=*group-variable*
organizes the data according to the values of *group-variable*.

LEVELS=*number-of-midpoints* | ALL
specifies the number of midpoints to be graphed for the chart variable.

MIDPOINTS=OLD
generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

MIDPOINTS=*value-list*
specifies the midpoint values for the blocks.

MISSING
accepts a missing value as a valid midpoint for the chart variable.

SUBGROUP=*subgroup-variable*
divides the blocks into segments according to the values of *subgroup-variable*.

ODS options

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Statistic options

FREQ=*numeric-variable*
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

G100
calculates the percentage and cumulative percentage statistics separately for each group.

STATFMT=*format-specification*
overrides the GCHART default format of the displayed statistical value.

SUMVAR=*numeric-variable*
specifies a numeric variable for sum or mean calculations.

TYPE=*statistic*
specifies the chart statistic.

Required Argument

chart-variable(s)
specifies one or more variables that define the categories of data to chart. Each chart variable draws a separate chart. All variables must be in the input data set. Separate multiple chart variables with blanks. The values of a chart variable used with the BLOCK statement have a maximum length of 13.

See "Understanding Chart Variables" on page 992

Optional Arguments

Options in a BLOCK statement affect all graphs produced by that statement. You can specify as many options as you want and list them in any order. For details about specifying colors, see Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363. For a complete description of the graphics options, see "Graphics Options and Device Parameters Dictionary" on page 577.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

Note Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block, pie, donut, or star charts.

See Chapter 28, "Using Annotate Data Sets," on page 721

BLOCKMAX=*max-value*
specifies the chart statistic value of the tallest block on the chart. This option lets you produce a series of block charts using the same scale. All blocks are rescaled using *max-value* as if it were the maximum value on the chart.

Restriction Not supported by Java or ActiveX

CAXIS=*grid-color*

specifies the color for the midpoint grid. By default, the midpoint grid uses the color of the current device. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

Style reference Color attribute of the GraphAxisLines element

Note Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

COUTLINE=*block-outline-color* | SAME

outlines all blocks or all block segments and legend values in the subgroup legend (if it appears) using the specified color. SAME specifies that the outline color of a block or a block segment or a legend value is the same as the interior pattern color.

The default outline color depends on the use of the PATTERN statement:

- If you do not specify a PATTERN statement, the default outline color is the color of the current device.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

Alias CO=

Style reference Color attribute of the GraphOutlines element

Restriction Partially supported by Java and ActiveX

Note Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

See “Controlling Block Chart Patterns and Colors” on page 1014

“About Patterns” on page 998

Example “Example 2: Grouping and Subgrouping a Block Chart” on page 1095

CTEXT=*text-color*

specifies a color for all text on the grid and legend. The GCHART procedure looks for the text color in the following order:

1. colors specified for labels and values on an assigned LEGEND statement, which override the CTEXT= option specified in the BLOCK statement.
2. the color specified by the CTEXT= option in the BLOCK statement.
3. the color specified by the CTEXT= option in a GOPTIONS statement.
4. the color specified in the current style. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

Alias	CT=
Style reference	Color attribute of the GraphValueText and the GraphLabelText elements
Note	If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" on page 251 for more information
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open
- the description and the properties for the output in the Results window
- the description and properties for the catalog entry in the SAS Explorer
- the **Description** field of the PROC GREPLAY window

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on statements such as TITLE, FOOTNOTE, and NOTE. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias	DES=
Default	BLOCK CHART OF *chart-variable*
Style reference	Color attributes of the GraphValueText and the GraphLabelText elements

DISCRETE

treats a numeric chart variable as a discrete variable rather than as a continuous variable. The GCHART procedure creates a separate midpoint and, hence, a separate grid square and block for each unique value of the chart variable. If the chart variable has a format that is associated with it, then each formatted value is treated as a midpoint.

The LEVELS=*number-of-midpoints* option is ignored when you use the DISCRETE option. The MIDPOINTS= option overrides the DISCRETE option.

FREQ=*numeric-variable*

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times specified by the value of *numeric-variable* for that observation. If the value of *numeric-variable* is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of *numeric-variable* are truncated to integers.

FREQ= is valid with all chart statistics.

You cannot use the PERCENT, CPERCENT, FREQ, or CFREQ statistics with the SUMVAR= option. As a result, you must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum.

The statistics are not affected by applying a format to *numeric-variable*.

See "Calculating Weighted Statistics" on page 997

G100
calculates the percentage and cumulative percentage statistics separately for each group. When you use G100, the individual percentages reflect the contribution of the midpoint to the group and total 100% for each group. G100 is ignored unless you also use the GROUP= option.

By default, the individual percentages reflect the contribution of the midpoint to the entire chart and total 100% for the entire chart.

GROUP=*group-variable*
organizes the data according to the values of *group-variable*. *Group-variable* can be either character or numeric and is always treated as a discrete variable. The group variable can have up to 12 different values.

GROUP= produces a group grid that contains a separate row of blocks for each unique value of the group variable. Each row contains a square for each midpoint. The groups are arranged from front to back in ascending order of the group variable values. These values are printed to the left of each row; the group variable name or label is printed above the list of group values.

By default, each group includes all midpoints, even if no observations for the group fall within the midpoint range. Missing values for *group-variable* are treated as a valid group.

Example "Example 2: Grouping and Subgrouping a Block Chart" on page 1095

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java or ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

LEGEND=LEGEND<1 …99>

assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option. The LEGEND= option itself does *not* generate a legend.

LEGEND= is ignored if the following is true:

- SUBGROUP= is not used.
- The specified LEGEND definition is not in effect.
- The NOLEGEND option is used.
- The PATTERNID= option is set to any value other than SUBGROUP. That is, the value of PATTERNID= is BY or GROUP or MIDPOINT.

To create a legend based on the chart midpoints instead of the subgroups, use the chart variable as the subgroup variable:

```
block city / subgroup=city;
```

The Java and ActiveX devices do not support all LEGEND statement options. See the "LEGEND Statement" on page 432 for more information.

Restriction Partially supported by Java and ActiveX

See "SUBGROUP=*subgroup-variable*" on page 1011

"LEGEND Statement" on page 432

Example "Example 2: Grouping and Subgrouping a Block Chart" on page 1095

LEVELS=*number-of-midpoints* | ALL

specifies the number of midpoints to be graphed for the chart variable. After you specify the number of midpoints that you want, the range for each midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985). When you specify LEVELS=ALL, up to 25 unique numeric or character midpoint values are graphed. LEVELS=ALL is ignored if more than 25 values are specified. The LEVELS=*number-of-midpoints* option is ignored if any of the following is true:

- The chart variable is character type.
- The DISCRETE option is used.
- The MIDPOINTS= option is used.

MIDPOINTS=*value-list*

specifies the midpoint values for the blocks. The way you specify *value-list* depends on the type of variable:

- For numeric chart variables, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - *n* < *...n*>
  - *n* TO *n* <BY *increment*>
  - *n* < *...n*> TO *n* <BY *increment*> <*n* < *...n*>>

If a numeric variable has an associated format, the specified values must be the *unformatted* values.

If you omit the DISCRETE option, then numeric values are treated as continuous, which means that the following is true by default:

- The lowest midpoint consolidates all data points from negative infinity to the median of the first two midpoints.
- The highest midpoint consolidates all data points from the median of the last two midpoints up to infinity.
- All other values in *value-list* specify the median of a range of values, and the GCHART procedure calculates the midpoint values.

  If you include the DISCRETE option, then each value in *value-list* specifies a unique numeric value.

- For character chart variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:

  'value-1' < *...'value-n'*>

  If a character variable has an associated format, the specified values must be the *formatted* values.

For a complete description of *value-list*, see the "*value-list*" on page 408 in the AXIS statement.

If *value-list* for either type of variable specifies so many midpoints that the axis values overwrite each other, then the values might be unreadable. In this case the procedure writes a warning to the SAS log. On many devices, you can correct crowded values by increasing the number of cells in your graphics is displayed using the HPOS= and VPOS= graphics options.

See "Understanding Midpoints" on page 993

Example "Example 2: Grouping and Subgrouping a Block Chart" on page 1095

MIDPOINTS=OLD

generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976). The MIDPOINTS=OLD option is ignored unless the chart variable is numeric.

MISSING

accepts a missing value as a valid midpoint for the chart variable. By default, observations with missing values are ignored. Missing values are always valid for the group and subgroup variables.

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.

- The first character is always represented in uppercase, and all other characters are represented in lowercase.
- If the name begins with a number, an underscore is prepended to the name.
- If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

For the graphics output file name:

- The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
- All characters are represented in lowercase.
- If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
- If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

  Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GCHART

NOHEADING

suppresses the heading describing the type of statistic. For the Java and ActiveX devices, NOHEADING is the default. For other devices, by default the heading is printed at the top of each block chart.

Restriction Not supported by Java or ActiveX

Example "Example 2: Grouping and Subgrouping a Block Chart" on page 1095

NOLEGEND

suppresses the legend automatically generated by the SUBGROUP= option. NOLEGEND is ignored if the SUBGROUP= option is not used.

PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP

specifies how fill patterns are assigned. By default, PATTERNID=SUBGROUP. Values for PATTERNID= are as follows:

BY
changes patterns each time the value of the BY variable changes. All blocks use the same pattern if the GCHART procedure does not include a BY statement.

GROUP
changes patterns every time the value of the group variable changes. All blocks in each group (row) use the same pattern, but a different pattern is used for each group.

MIDPOINT
changes patterns every time the midpoint value changes. If you use the GROUP= option, the respective midpoint patterns are repeated for each group.

SUBGROUP
changes patterns every time the value of the subgroup variable changes. The blocks must be subdivided by the SUBGROUP= option for the SUBGROUP value to have an effect. Without the SUBGROUP= option, all block faces have the same pattern.

Note If you use the SUBGROUP= option and specify a PATTERNID= value other than SUBGROUP, the block segments use the same pattern and are indistinguishable.

See "Controlling Block Chart Patterns and Colors" on page 1014

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

STATFMT=*format-specification*
overrides the GCHART default format of the displayed statistical value.

Note: This option is valid starting with SAS 9.4M5.

The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use the STATFMT= option to change the default format that might contain decimal points, percentages, or commas. Here is an example:

```
proc gchart data=sashelp.electric(where=(year ge 2000));
   block year / discrete sumvar=Revenue
      type=sum statfmt=dollar8.2;
run;
quit;
```

In this example, the STATFMT= option in the BLOCK statement specifies format DOLLAR8.2 for the SUM values.

Alias SFMT=, SFORMAT=, STATFORMAT=

SUBGROUP=*subgroup-variable*
divides the blocks into segments according to the values of *subgroup-variable*. *Subgroup-variable* can be either character or numeric and is always treated as a discrete variable. SUBGROUP= creates a separate segment within each block for every unique value of the subgroup variable for that midpoint.

If PATTERNID=SUBGROUP (the default setting), each segment is filled with a different pattern, and a legend providing a key to the patterns is automatically generated. If the value of PATTERNID= is anything other than SUBGROUP, the

segments are all the same color, the legend is suppressed, and the subgrouping effect is lost.

By default the legend appears at the bottom of the chart. To modify the legend, assign a LEGEND definition with the LEGEND= option. To suppress the legend, specify NOLEGEND.

See "LEGEND Statement" on page 432

Example "Example 2: Grouping and Subgrouping a Block Chart" on page 1095

SUMVAR=*numeric-variable*

specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of *numeric-variable* for each midpoint. The resulting statistics are represented by the height of the blocks in each square. The values of a numeric variable used with the BLOCK statement have a maximum length of 8.

When you use the SUMVAR= option, the TYPE= option value must be either SUM or MEAN. With SUMVAR=, the default is TYPE=SUM.

Example "Example 1: Specifying the Sum Statistic in a Block Chart" on page 1093

TYPE=*statistic*

specifies the chart statistic.

- If the SUMVAR= option is not used, *statistic* can be one of the following:

FREQ
: frequency (the default)

CFREQ
: cumulative frequency

PERCENT PCT
: percentage

CPERCENT CPCT
: cumulative percentage

- If SUMVAR= is used, *statistic* can be either:

SUM
: sum (the default)

MEAN
: mean

You cannot specify the statistics PERCENT, CPERCENT, FREQ, or CFREQ in conjunction with the SUMVAR= option. As a result, you must use FREQ= to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum. See also "Calculating Weighted Statistics" on page 997.

If you specify TYPE=MEAN and use the SUBGROUP= option, the height of the block represents the mean for the entire midpoint. The subgroup segments are proportional to the subgroup's contribution to the sum for the block.

See "About Chart Statistics" on page 996

Example "Example 2: Grouping and Subgrouping a Block Chart" on page 1095

URL=*character-variable*
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction This option affects graphics output that is created through the ODS HTML destination only.

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

WOUTLINE=*block-outline-width*
specifies the width of the block outline in pixels.

Style reference LineThickness attribute of the GraphOutlines element

Restriction Not supported by Java

Details

Description

The BLOCK statement specifies the variable or variables that define the categories of data to chart. This statement automatically does the following actions:

- determines the midpoints.
- calculates the chart statistic for each midpoint (the default is FREQ).
- scales the blocks according to the statistic value.
- assigns patterns and colors to the block faces and the grid. The default block pattern is solid.

You can use statement options to select or order the midpoints (blocks), to change the type of chart statistic, and to modify the appearance of the chart. You can also specify additional variables by which to group, subgroup, or sum the data.

Block charts enable grouping, which organizes the blocks into rows based on the values of a group variable. Block charts also enable subgrouping, which subdivides the blocks into segments based on the values of a subgroup variable.

In addition, you can use global statements to modify the block patterns and the legend, as well as add titles, footnotes, and notes to the chart. You can also use an Annotate data set to enhance the chart.

Note: If you get a message that the chart is too large to display on your terminal or printer, try one or both of the following:

- reduce the size of the character cells defined for the output device by specifying larger values for the HPOS= and VPOS= graphics options

- decrease the size of the chart text with the HTEXT= graphics option

See "The Graphics Output and Device Display Areas" on page 80 for details.

Controlling Block Chart Patterns and Colors

Default Patterns and Outlines

Each block in a block chart is filled with a pattern, but only the front faces of the blocks display the patterns. Because the system option, GSTYLE, is in effect by default, the procedure uses the style's default patterns and outlines when producing output. By default, the procedure does the following:

- fills the bars with bar or block patterns, beginning with the default fill, SOLID, and uses each color in the color list available in the default style. When these colors are exhausted, the procedure rotates through a slightly modified version of the original list of colors. It continues in this fashion until all of the chart variables have been assigned a unique pattern.

  If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, the procedure uses all the colors in the list to generate the patterns.
- outlines blocks and block segments using the color defined by the style.
- colors the midpoint grid with the color of the current style.

See "About Patterns" on page 998 for more information about how the GCHART procedure assigns default patterns and outlines.

User-Defined Patterns

To override the default patterns and select fills and colors for the blocks or block segments, use the PATTERN statement. Only bar or block patterns are valid; all other pattern fills are ignored. For a complete description of all bar or block patterns, see the description of the PATTERN statement option " VALUE=*bar/block-pattern*" on page 457.

Whenever you use PATTERN statements, the default pattern outline color is that of the current style. Only when the EMPTY pattern is used does the pattern change to SAME. That is, the outline color is the same as the fill color. To specify the outline color, use the "COUTLINE=*block-outline-color* | SAME" on page 1005.

When Patterns Change

The PATTERNID= option controls when the pattern changes. By default, PATTERNID=SUBGROUP. Therefore, when you use the SUBGROUP= option to subdivide the blocks, the pattern automatically changes each time the subgroup value changes. Each subdivision of the block displays a different pattern. As a result, the number of values for the SUBGROUP= variable determines the number of block patterns on the chart. If you do not subdivide the blocks, all blocks use the same pattern.

Instead of changing the pattern for each subgroup, you can change the pattern for each midpoint, each group, or each BY group, by changing the value of the PATTERNID= option. See the "About Patterns" on page 998 for details.

Grid Color

By default, grid elements use the color specified in the current style. To change the grid color, use the CAXIS= option. To change the grid text color, use the CTEXT= option.

Controlling Block Chart Text

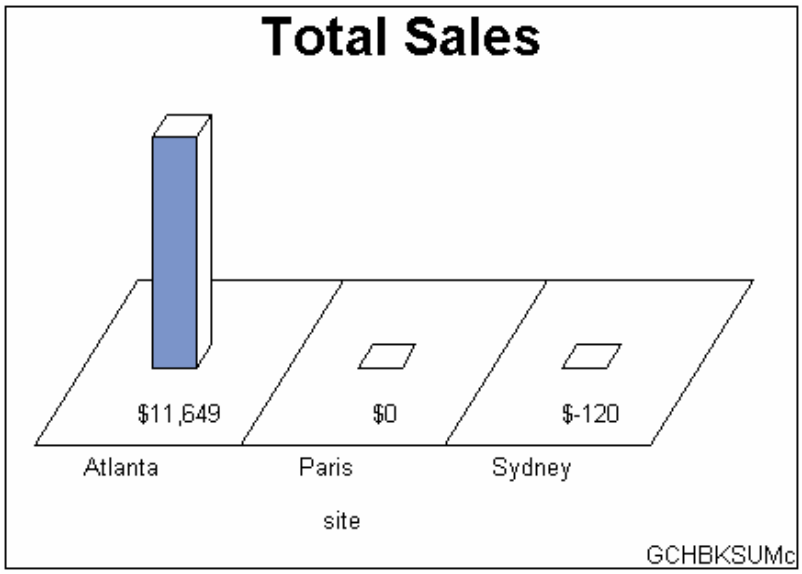
To control the font and size of text on the chart, use the graphic options "FTEXT Graphics Option" on page 626 and "HTEXT Graphics Option" on page 650.

Because block charts do not use AXIS statements, you must use a LABEL statement instead to suppress the label for the midpoint variable. See "Example 2: Grouping and Subgrouping a Block Chart" on page 1095.

Displaying Negative or Zero Values

The relative block heights in the chart represent the scaled value of the chart statistic value for the midpoint. The statistic can have a value of 0 or, in the case of sum and mean, a negative value. In these cases the base of the block is drawn in the square for the corresponding midpoint. Figure 36.121 on page 1015 shows an example of a chart with 0 and negative statistic values.

Figure 36.13 *Block Chart with 0 and Negative Statistic Values*



HBAR, HBAR3D, VBAR, and VBAR3D Statement

Create horizontal or vertical bar charts in which the length or height of the bars represents the value of the chart statistic for each category of data.

Requirement:	At least one chart variable is required.
Supported Statements:	AXIS, FOOTNOTE, LEGEND, PATTERN, TITLE
Supports:	Drill-down functionality

Syntax

HBAR | HBAR3D | VBAR | VBAR3D *chart-variable(s)* < / *options* >;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate all graphs that are produced by the GCHART procedure.

CAUTOREF=*reference-line-color*
: specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option.

CAXIS=*axis-color*
: specifies a color for the response and midpoint axis lines and for the default axis area frame.

CERROR=*error-bar-color*
: specifies the color of error bars in bar charts.

CFRAME=*background-color*
: specifies the color with which to fill the axis area in two-dimensional bar charts or the floor, back wall, and side wall of three-dimensional bar charts.

CGROUPREF (HBAR and VBAR only)
: specifies the color of the reference line dividing each group of bars produced with the HBAR and VBAR statements.

CLM=*confidence-level*
: specifies the confidence intervals to use when drawing error bars.

CMIDPOINTREF (HBAR and VBAR only)
: specifies the color of the midpoint reference lines between bars produced with the HBAR and VBAR statements.

COUTLINE=*bar-outline-color* | SAME
: outlines all bars or bar segments and legend values in the subgroup legend (if it appears) using the specified color.

CREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)

specifies colors for reference lines.

CTEXT=*text-color*
specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions.

FRAME

NOFRAME
specifies whether a line is drawn around the axis area of a two-dimensional bar chart or a back wall is drawn in a three-dimensional bar chart.

GROUPREF (HBAR and VBAR only)
draws a reference line dividing each group of bars produced by the HBAR and VBAR statements.

GSPACE=*group-spacing*
specifies the amount of extra space between groups of bars.

IFRAME=*fileref* | '*external-file*'
identifies the image file that you want to apply to a two-dimensional chart's axis area or a three-dimensional chart's back wall.

IMAGESTYLE=TILE | FIT
specifies whether to use multiple instances of an image, or stretch a single image.

LAUTOREF=*reference-line-type*
specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option.

LEGEND=LEGEND<1 …99>
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.

LGROUPREF=*reference-line-type*
specifies line types for reference lines dividing each group of bars produced by the HBAR and VBAR statements.

LMIDPOINTREF=*reference-line-type*
specifies line types for midpoint reference lines between bars produced by HBAR and VBAR statements.

LREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines.

MIDPOINTREF (HBAR and VBAR only)
draws a reference line at the midpoint between each bar produced by the HBAR and VBAR statements.

NOLEGEND
suppresses the legend that is automatically generated by the SUBGROUP= option.

NOSYMBOL
suppresses the legend that is automatically generated by the SUBGROUP= option.

PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP
specifies how fill patterns are assigned.

SHAPE=*three-dimensional-bar-shape* (HBAR3D and VBAR3D only)
specifies the shape of the bars in charts that are produced with the HBAR3D and VBAR3D statements.

SPACE=*bar-spacing*
specifies the amount of space between individual bars or between the bars within each group if you also use the GROUP= option.

SUBOUTSIDE=*statistic* (VBAR and VBAR3D only)

displays a list of the subgroup values of the specified statistic above the bars.

WAUTOREF=*reference-line-width*
specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option.

WGROUPREF=*reference-line-width*
specifies the line width for reference lines between groups of bars produced by the HBAR and VBAR statements.

WIDTH=*bar-width*
specifies the width of the bars.

WMIDPOINTREF=*reference-line-width*
specifies the line width for midpoint reference lines between bars produced by the HBAR and VBAR statements.

WOUTLINE=*bar-outline-width*
specifies the width of the outline in pixels.

WREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines.

Axis options

ASCENDING
arranges the bars in ascending order of the value of the chart statistic.

AUTOREF
draws a reference line at each major tick mark on the response axis.

AXIS=AXIS<1 …99>
specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

CLIPREF
clips the reference lines at the bars.

DESCENDING
arranges the bars in descending order of the value of the chart statistic.

FRONTREF
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

GAXIS=AXIS<1 …99>
assigns the specified AXIS definition to the group axis.

MAXIS=AXIS<1 …99>
assigns the specified AXIS definition to the midpoint axis.

MINOR=*number-of-minor-ticks*
specifies the number of minor tick marks between each major tick mark on the response axis.

NOAXIS
suppresses all axes, including axis lines, axis labels, axis values, and all major and minor tick marks.

NOBASEREF
suppresses the zero reference line when the SUM or MEAN chart statistic has negative values.

NOZERO
suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar.

RANGE
displays on the axis of the chart the range of numeric values represented by each bar.

RAXIS=*value-list* | AXIS<1 – 99>

AXIS=*value-list* | AXIS<1 – 99>

specifies values for the major tick mark divisions on the response axis or assigns the specified AXIS definition to the axis.

REF=*value* | (*value*) | (*value-list*)

draws reference lines at the specified points on the response axis.

Catalog entry description options

DESCRIPTION="*description*"

specifies a description of the output.

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Midpoint options

DISCRETE

treats a numeric chart variable as a discrete variable rather than as a continuous variable.

GROUP=*group-variable*

organizes the data according to values of *group-variable*.

LEVELS=*number-of-midpoints* | ALL

specifies the number of midpoints to be graphed for a chart variable.

MIDPOINTS=OLD

generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

MIDPOINTS=*value-list*

specifies the midpoint values for the bars.

MISSING

accepts a missing value as a valid midpoint for the chart variable.

SUBGROUP=*subgroup-variable*

divides the bars into segments according to the values of *subgroup-variable*.

ODS options

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*

identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*

specifies a character variable whose values are URLs.

Statistic options

CFREQ

displays the cumulative frequency statistic in the table of statistics and above vertical bars.

CFREQLABEL='*column-label*' | NONE (HBAR and HBAR3D only)

specifies the text of the column label for the CFREQ statistic in the table of statistics.

CPERCENT

displays the cumulative percentage statistic in the table of statistics and above vertical bars.

CPERCENTLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the CPERCENT statistic in the table of statistics.

ERRORBAR=BARS | BOTH | TOP
draws confidence levels.

FREQ
displays the frequency statistic in the table of statistics and above vertical bars.

FREQ=*numeric-variable*
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

FREQLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the FREQ statistic in the table of statistics.

G100
calculates the percentage and cumulative percentage statistics separately for each group.

INSIDE=*statistic*
displays the values of the specified statistic inside the bars.

MEAN
displays the mean statistic in the table of statistics and above vertical bars.

MEANLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the MEAN statistic in the table of statistics.

NOSTATS (HBAR and HBAR3D only)
suppresses the table of statistics.

OUTSIDE=*statistic*
displays the values of the specified statistic above the bars.

PERCENT
prints the percentages of observations having a given value for the chart variable in the table of statistics and above vertical bars.

PERCENTLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the PERCENT statistic in the table of statistics.

PERCENTSUM
calculates a percent of the sum variable for horizontal or vertical bar charts.

STATFMT=*format-specification*
overrides the GCHART default format of the displayed statistical value.

SUM
displays the sum statistic in the table of statistics and above vertical bars.

SUMLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the SUM statistic in the table of statistics.

SUMVAR=*numeric-variable*
specifies a numeric variable for sum or mean calculations.

TYPE=*statistic*
specifies the chart statistic.

Required Argument

chart-variable(s)

specifies one or more variables that define the categories of data to chart. Each chart variable draws a separate chart. All variables must be in the input data set. Multiple chart variables must be separated with blanks.

See "Understanding Chart Variables" on page 992

Optional Arguments

Options in an HBAR, HBAR3D, VBAR, or VBAR3D statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order. For details about specifying colors, see Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363. For details about specifying images, see Chapter 24, "Adding Images to SAS/GRAPH Output," on page 383. For a complete description of the graphics options, see "Graphics Options and Device Parameters Dictionary" on page 577.

ANNOTATE=*Annotate-data-set*

specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

Note Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block, pie, donut, or star charts.

See Chapter 28, "Using Annotate Data Sets," on page 721

ASCENDING

arranges the bars in ascending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. The ASCENDING option reorders the bars from shortest to longest. In horizontal bar charts the ordering is top to bottom; in vertical bar charts the ordering is left to right.

If you also use the GROUP= option, the reordering is performed separately for each group, so the order of the midpoints might be different for each group.

The ASCENDING option overrides any midpoint order specified with the MIDPOINTS= option or specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

AUTOREF

draws a reference line at each major tick mark on the response axis. To draw reference lines at specific points on the response axis, use the REF= option.

By default, reference lines in two-dimensional bar charts are drawn in front of the bars. To draw reference lines behind the bars, use the CLIPREF option.

By default, reference lines in three-dimensional bar charts are drawn on the back plane of the axis. To draw reference lines in front of the bars, use the FRONTREF option.

Example "Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart" on page 1103

AXIS=AXIS<1 …99>

specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

See "RAXIS=*value-list* | AXIS<1 – 99> |AXIS=*value-list* | AXIS<1 – 99>" on page 1041

CAUTOREF=*reference-line-color*

specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option. If you do not specify the CAUTOREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified. To specify a line type for these reference lines, use the LAUTOREF= option.

Style reference Color attribute of the GraphGridLines element

CAXIS=*axis-color*

specifies a color for the response and midpoint axis lines and for the default axis area frame. If you omit the CAXIS= option, PROC GCHART searches for a color specification in this order:

1. the COLOR= option in AXIS definitions
2. the color specified in the current device. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

This option also specifies the default color for all reference lines.

Style reference Color attribute of the GraphAxisLines attribute

Note Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

CERROR=*error-bar-color*

specifies the color of error bars in bar charts. The default is the color of the response axis, which is controlled by the CAXIS= option.

Style reference Color attribute of the GraphError element

CFRAME=*background-color*

specifies the color with which to fill the axis area in two-dimensional bar charts or the floor, back wall, and side wall of three-dimensional bar charts. The axis area color does not affect the frame color, which is always the same as the midpoint axis line color and controlled by the CAXIS= option. By default, the axis area in two-dimensional bar charts is not filled.

Alias CFR=

Style reference Color attribute of the GraphWalls element

Restriction CFRAME= is overridden by the NOFRAME and IFRAME= options. If the IFRAME= option is in effect, an image appears in the axis area instead of the color.

Note If the color of the background, the bar, and the outline are the same, you might not be able to distinguish the bars.

See "FRAME|NOFRAME" on page 1028

"IFRAME=*fileref* | '*external-file*'" on page 1031

Example "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

CFREQ

displays the cumulative frequency statistic in the table of statistics and above vertical bars. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ option is specified.

Restriction The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

Interaction Specifying the CFREQ option when creating an HBAR chart with an ActiveX device causes the cumulative frequencies to display at the end of each bar instead of the frequencies.

See "About Chart Statistics" on page 996

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

CFREQLABEL='*column-label*' | NONE (HBAR and HBAR3D only)

specifies the text of the column label for the CFREQ statistic in the table of statistics. *Column-label* can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify CFREQLABEL=NONE.

Restriction Not supported by Java or ActiveX

CGROUPREF (HBAR and VBAR only)

specifies the color of the reference line dividing each group of bars produced with the HBAR and VBAR statements. To draw reference lines between specific groups of bars produced with the HBAR and VBAR statements, use the GROUPREF= option.

CLIPREF

clips the reference lines at the bars. This makes the reference lines appear to be behind the bars. Because the CLIPREF option is the default for three-dimensional bar charts, it affects only two-dimensional charts.

Example "Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart" on page 1103

CLM=*confidence-level*

specifies the confidence intervals to use when drawing error bars. Values for *confidence-level* must be greater than or equal to 50 and strictly less than 100. The default is 95. See "ERRORBAR=BARS | BOTH | TOP" on page 1027 for details about how error bars are computed and drawn.

CMIDPOINTREF (HBAR and VBAR only)
specifies the color of the midpoint reference lines between bars produced with the HBAR and VBAR statements. To draw the reference lines between specific midpoints of bars produced with the HBAR and VBAR statements, use the MIDPOINTREF= option.

COUTLINE=*bar-outline-color* | SAME
outlines all bars or bar segments and legend values in the subgroup legend (if it appears) using the specified color. SAME specifies that the outline color of a bar or a bar segment or a legend value is the same as the interior pattern color.

The default outline color depends on the use of the PATTERN statement:

- If you do not specify a PATTERN statement, the default outline color is the color of the current device.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

Alias CO=

Style reference Color attribute of the GraphOutlines element

Note Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

See “Controlling Bar Chart Patterns, Colors, and Images” on page 1050

“About Patterns” on page 998

Examples “Example 3: Specifying the Sum Statistic in Bar Charts” on page 1098

“Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 1103

“Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 1107

CPERCENT
displays the cumulative percentage statistic in the table of statistics and above vertical bars. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ, CFREQ, or PERCENT option is specified.

Alias CPCT=

Restriction The table of statistic values is not printed when creating an HBAR chart with an ActiveX device

Interaction Specifying the CPERCENT option when creating an HBAR chart with an ActiveX device causes the cumulative percentages to display at the end of each bar instead of the frequencies.

See "About Chart Statistics" on page 996

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

CREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies colors for reference lines. Specifying a single color without parentheses applies that color to all reference lines, including lines drawn with the AUTOREF and REF= options. Note that the CAUTOREF= option overrides the CREF= reference color for reference lines drawn with the AUTOREF option. Specifying a single color in parentheses applies that color to only the first reference line drawn with the REF= option. Specifying a reference color list applies colors in sequence to successive lines drawn with the REF= option. The syntax of the color list requires parentheses and color names separated by spaces (*color1 color2 ...colorN*) or separated by commas (*color1, color2 ..., colorN*). If you do not specify the CREF= option, the GCHART procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified. To specify line types for these reference lines, use the LREF= option.

Alias CR=

Style reference LineStyle attribute of the GraphGridLines element

CPERCENTLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the CPERCENT statistic in the table of statistics. *Column-label* can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify CPERCENTLABEL=NONE.

Restriction Not supported by Java or ActiveX

CTEXT=*text-color*
specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions. The GCHART procedure searches for a color specification in this order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the BAR statement.
2. the color specified by the CTEXT= option in the BAR statement.
3. the color specified by the CTEXT= option in the GOPTIONS statement.
4. the color specified in the current style. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. In this

instance the COLOR= color is still used for all other axis colors, such as tick marks.

Alias CT=

Style reference Color attributes of the GraphValueText and the GraphLabelText elements

Note If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

DESCENDING

arranges the bars in descending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. The DESCENDING option reorders the bars from longest to shortest. In horizontal bar charts the ordering is top to bottom; in vertical bar charts the ordering is left to right. If you also use the GROUP= option, the reordering is performed separately for each group, so the order of the midpoints might be different for each group.

The DESCENDING option overrides any midpoint order that is specified with the MIDPOINTS= option. The DESCENDING option also overrides any midpoint order that is specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the SAS Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default BAR CHART OF *chart-variable*

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

DISCRETE

treats a numeric chart variable as a discrete variable rather than as a continuous variable. The GCHART procedure creates a separate midpoint and, hence, a

separate bar for each unique value of the chart variable. If the chart variable has a format that is associated with it, each formatted value is treated as a midpoint.

The LEVELS=*number-of-midpoints* option is ignored when you use the DISCRETE option. The MIDPOINTS= option overrides the DISCRETE option. The ORDER= option in an AXIS statement that is assigned to the midpoint axis can rearrange or exclude discrete midpoint values.

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

ERRORBAR=BARS | BOTH | TOP

draws confidence intervals for either of the following:

- the mean of the SUMVAR= variable for each midpoint if you specify TYPE=MEAN
- the percentage of observations assigned to each midpoint if you specify TYPE=PCT with no SUMVAR= option

The ERRORBAR= option cannot be used with values of the TYPE= option other than MEAN or PCT. Valid values for the ERRORBAR= option are:

BARS
draws error bars as bars half the width of the main bars.

BOTH
draws error bars as two ticks joined by a line (default).

TOP
draws the error bar as a tick for the upper confidence limit that is joined to the top of the bar by a line.

By default, the ERRORBAR= option uses a confidence level of 95%. You can specify different confidence levels with the CLM= option.

When you use the ERRORBAR= option with TYPE=PCT, the confidence interval is based on a normal approximation. Let TOTAL be the total number of observations, and PCT be the percentage assigned to a given midpoint. The standard error of the percentage is approximated as follows:

```
APSTDERR=100 * SQRT((PCT/100) * (1-(PCT/100)) / TOTAL);
```

Let LEVEL be the confidence level specified using the CLM= option, with a default value of 95. The upper confidence limit for the percentage is computed as follows:

```
UCLP = PCT + APSTDERR * PROBIT( 1-(1-LEVEL/100)/2 );
```

The lower confidence limit for the percentage is computed as follows:

```
LCLP = PCT - APSTDERR * PROBIT( 1-(1-LEVEL/100)/2 );
```

When you use the ERRORBAR= option with TYPE=MEAN, the sum variable must have at least two nonmissing values for each midpoint. Let N be the number of observations assigned to a midpoint, MEAN be the mean of those observations, and STD be the standard deviation of the observations. The standard error of the mean is computed as follows:

```
STDERR = STD / SQRT(N);
```

Let LEVEL be the confidence level specified using the CLM= option, with a default value of 95. The upper confidence limit for the mean is computed as follows:

```
UCLM = MEAN + STDERR * TINV( 1-(1-LEVEL/100)/2, N-1);
```

The lower confidence limit for the mean is computed as

```
LCLM = MEAN - STDERR * TINV( 1-(1-LEVEL/100)/2, N-1);
```

In some instances, you want the error bars to represent a given number C of standard errors instead of a confidence interval. If the number of observations assigned to each midpoint is the same, then you can find the appropriate value for the CLM= option by running a DATA step. For example, suppose you want error bars that represent one standard error (C=1) with a sample size of N. Run the following DATA step to compute the appropriate value for the CLM= option and assign that value to a macro variable &LEVEL:

```
data _null_;
c = 1; n = 10;
level = 100 * (1 - 2 * (1 - probt( c, n-1)));
put _all_;
call symput("level",put(level,best12.));
run;
```

Then when you run the GCHART procedure, you can specify CLM=&LEVEL.

Note that this trick does not work precisely if different midpoints have different numbers of observations. However, choosing an average value for N can yield sufficiently accurate results for graphical purposes if the sample sizes are large or do not vary much.

If you want to create a chart with a legend and error bars, use the SUBGROUP= option with the ERRORBAR= option, and set the SUBGROUP= option to the midpoint variable.

FRAME | NOFRAME

specifies whether a line is drawn around the axis area of a two-dimensional bar chart or a back wall is drawn in a three-dimensional bar chart. The default is FRAME. Specifying the NOFRAME option removes the axis area frame from two-dimensional charts, including any background color or image. For three-dimensional charts, NOFRAME removes the back wall color or image, and leaves the vertical and horizontal axis planes and axes. To remove these planes, use the NOPLANE option in the AXIS statement. To remove one or more axis elements, use either the AXIS statement or the NOAXIS option.

The color of the frame is the color of the midpoint axis, which is determined by the CAXIS= option.

Alias FR | NOFR

Restriction The NOFRAME option overrides the CFRAME= and IFRAME= options and the goption “IBACK Graphics Option” on page 651.

See “CFRAME=*background-color*” on page 1022

“IFRAME=*fileref* | '*external-file*'” on page 1031

Examples “Example 2: Creating a Drill-Down HTML Graph” on page 269

“Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 1107

FREQ

displays the frequency statistic in the table of statistics and above vertical bars. Non-integer values are rounded down to the nearest integer. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this

option is ignored if the bars are too narrow to avoid overlapping values. This option overrides the CFREQ, PERCENT, CPERCENT, SUM, and MEAN options.

See "About Chart Statistics" on page 996

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

Example "Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart" on page 1103

FREQLABEL='*column-label*' | NONE (HBAR and HBAR3D only)

specifies the text of the column label for the FREQ statistic in the table of statistics. *column-label* can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify FREQLABEL=NONE.

Restriction Not supported by Java or ActiveX

Examples "Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart" on page 1103

"Example 6: Generating Error Bars in a Horizontal Bar Chart" on page 1107

FREQ=*numeric-variable*

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times that is specified by the value of *numeric-variable* for that observation. If the value of *numeric-variable* is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of *numeric-variable* are truncated to integers. FREQ= is valid with all chart statistics.

Because you cannot use TYPE=PERCENT, TYPE=CPERCENT, TYPE=FREQ, or TYPE=CFREQ with the SUMVAR= option, you must use FREQ= to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum.

The statistics are not affected by applying a format to *numeric-variable*.

Restriction The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

Interaction When specifying the ACTIVEX device with an HBAR chart, the frequencies are displayed at the end of each bar by default.

See "Calculating Weighted Statistics" on page 997

FRONTREF

specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars. By default, reference lines in three-dimensional bar charts are drawn on the back plane of the axis.

G100

calculates the percentage and cumulative percentage statistics separately for each group. When you use the G100 option, the individual percentages reflect the contribution of the midpoint to the group and total 100% for each group. The G100 option is ignored unless you also use the GROUP= option.

By default, the individual percentages reflect the contribution of the midpoint to the entire chart and total 100% for the entire chart.

GAXIS=AXIS<1 …99>

assigns the specified AXIS definition to the group axis. (A group axis is created when you use the GROUP= option.) You can use the AXIS definition to modify the order of the groups, the text of the labels, and appearance of the axis. The GAXIS= option is ignored if the specified AXIS definition does not exist.

The AXIS statement options MAJOR= and MINOR= are ignored in AXIS definitions assigned to the group axis because the axis does not use tick marks. A warning message is written to the SAS log if these options appear in the AXIS definition.

The Java and ActiveX devices do not support all AXIS statement options. See the "AXIS Statement" on page 399 for more information.

To remove groups from the chart, use the ORDER= option in the AXIS statement.

To suppress the brackets drawn around the values on the group axis in vertical bar charts, use the NOBRACKETS option in the AXIS statement.

Restriction Partially supported by Java and ActiveX

See "AXIS Statement" on page 399

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

GROUP=*group-variable*

organizes the data according to values of *group-variable*. *Group-variable* can be either character or numeric and is always treated as a discrete variable.

The GROUP= option produces a separate group of bars for each unique value of the group variable. Missing values for *group-variable* are treated as a valid group. The groups are arranged in ascending order of the group variable values.

By default, each group includes all midpoints, even if no observations for the group fall within the midpoint range, meaning that no bar is drawn at the midpoint. Use the NOZERO option to suppress midpoints with no observations.

The GROUP= option also produces a *group axis* that lists the values that distinguish the groups. The group axis has no axis line but displays up to 256 characters of the group variable name or label. To modify the group axis, assign an AXIS definition with the GAXIS= option.

In horizontal bar charts, the group axis is to the left of the midpoint axis. The groups are arranged from top to bottom, starting with the lowest value at the top.

In vertical bar charts, the group axis is below the midpoint axis and the groups are arranged from left to right starting with the lowest value at the left. The group label in a vertical bar chart can be narrower than all the bars in the group. In this case brackets are added to the label to emphasize which bars belong in each group. Group brackets are not displayed if the space between the group values is less than one and one-half character cells. Use the NOBRACKETS option in the AXIS statement to suppress the group brackets.

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

GROUPREF (HBAR and VBAR only)
draws a reference line dividing each group of bars produced by the HBAR and VBAR statements. To draw reference lines at specific points on the response axis, use the REF= option.

GSPACE=*group-spacing*
specifies the amount of extra space between groups of bars. *Group-space* can be any nonnegative number. Units are character cells. Use GSPACE=0 to leave no extra space between adjacent groups of bars. In this case, the same space appears between groups of bars as between the bars in the same group.

The GSPACE= option is ignored unless you also use the GROUP= option. By default, the GCHART procedure calculates group spacing based on the size of the axis area and the number of bars in the chart.

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java or ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

IFRAME=*fileref* | '*external-file*'
identifies the image file that you want to apply to a two-dimensional chart's axis area or a three-dimensional chart's back wall. *fileref* must be a valid SAS fileref up to eight characters long and must have been previously assigned with a FILENAME statement. *external-file* must specify the complete filename of the image file that you want to use. The format of *external-file* varies across operating environments.

Restriction Not supported by Java

Interactions This option is overridden by the NOIMAGEPRINT goption. For information about the NOIMAGEPRINT goption, see "IMAGEPRINT, NOIMAGEPRINT Graphics Options" on page 653.

This option is ignored if you specify the NOFRAME option or if you specify the STYLE=0 option in the AXIS statement.

See the goption "IBACK Graphics Option" on page 651 to fill the background surrounding two- and three-dimensional bar charts

"IMAGESTYLE=TILE | FIT" on page 1032

"Displaying Images on Data Elements" on page 389

"CFRAME=*background-color*" on page 1022

IMAGESTYLE=TILE | FIT

specifies whether to use multiple instances of an image, or stretch a single image. Use multiple instances of an image to fill the axis area or back wall (TILE). You can choose to stretch a single instance of an image to fill the axis area or back wall (FIT).The TILE value is the default.

Restriction Not supported by Java

See "IFRAME=*fileref* | '*external-file*'" on page 1031

INSIDE=*statistic*

displays the values of the specified statistic inside the bars. For the Java and ActiveX devices, this option is valid for both horizontal and vertical bar charts. However, Java and ActiveX devices do not support INSIDE=PERCENTSUM. For other devices, this option is valid only for vertical bar charts. For subgrouped bar charts generated with the Java and ActiveX devices, you can display only one statistic for each bar. These devices do not create both inside and outside bar labels. For graphs generated with the Java and ActiveX devices, the OUTSIDE= option overrides the INSIDE= option.

Statistic can be one of the following:

- FREQ
- CFREQ
- CPERCENT | CPCT
- MEAN
- PERCENT | PCT
- PERCENTSUM | PCTSUM
- SUM

If the bars are subgrouped, only the following statistics are valid:

- FREQ
- PERCENT | PCT
- SUBPCT
- PERCENTSUM | PCTSUM
- SUM

With subgroups, PERCENT displays the percent contribution of each subgroup to the midpoint value of the bar, based on frequency. The PERCENT values for each subgroup total the percent contribution of the bar to the whole. For example, if the percent contribution of the whole bar is 60%, the PERCENT

statistic for all the subgroups in that bar are 60% total. To calculate PERCENT based on the SUMVAR= variable, use the FREQ= and TYPE= options. For details, see "Calculating Weighted Statistics" on page 997.

SUBPCT displays the percent contribution of each subgroup to the total bar. The SUBPCT values for each subgroup total the percent contribution to the whole bar. Because of rounding, the total of the percents might not equal 100.

See "About Chart Statistics" on page 996

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

Examples "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

"Example 2: Creating a Drill-Down HTML Graph" on page 269

LAUTOREF=*reference-line-type*

specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify a color for these reference lines, use the CAUTOREF= option.

Style reference LineStyle attribute of the GraphGridLines element

LEGEND=LEGEND<1 …99>

assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option. The LEGEND= option itself does *not* generate a legend.

LEGEND= is ignored if any of the following is true:

- The SUBGROUP= option is not used.
- The specified LEGEND definition is not in effect.
- The NOLEGEND option is used.
- The PATTERNID= option is set to any value other than SUBGROUP. That is, the value of PATTERNID= is BY or GROUP or MIDPOINT.

To create a legend based on the chart midpoints instead of the subgroups, use the chart variable as the subgroup variable:

```
hbar city / subgroup=city;
```

Restriction The Java and ActiveX devices do not support all LEGEND statement options.

See "LEGEND Statement" on page 432

"SUBGROUP=*subgroup-variable*" on page 1043

Example "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

LEVELS=*number-of-midpoints* | ALL
specifies the number of midpoints to be graphed for a chart variable. After you specify the number of midpoints that you want, the range for each midpoint is calculated automatically, using the algorithm in Terrell and Scott (1985).

If you specify LEVELS=ALL, then all unique numeric or character midpoint values are graphed. You might have instances where the data contains a large number of unique midpoint values (more than 200). In this case you can use the XPIXELS and YPIXELS GOPTIONS to enable the device driver to render a larger (and more readable) graph.

The LEVELS=*number-of-midpoints* option is ignored if any of the following are true:

- The chart variable is character type.
- The DISCRETE option is used.
- The MIDPOINTS= option is used.

LGROUPREF=*reference-line-type*
specifies line types for reference lines dividing each group of bars produced by the HBAR and VBAR statements. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify a color for these reference lines, use the CGROUPREF= option. To specify a width for these reference lines, use the WGROUPREF= option.

Style reference LineStyle attribute of the GraphGridLines element

LMIDPOINTREF=*reference-line-type*
specifies line types for midpoint reference lines between bars produced by HBAR and VBAR statements. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify a color for these reference lines, use the CMIDPOINTREF= option. To specify a width for these reference lines, use the WMIDPOINTREF= option.

Style reference LineStyle attribute of the GraphGridLines element

LREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. Specifying a line type without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the LAUTOREF= option overrides LREF=*reference-line-type* for reference lines drawn with the AUTOREF option. Specifying a single line type in parentheses applies that line type to the first reference line drawn with the REF= option. Specifying a line type list applies line types in sequence to successive reference lines drawn with the REF= option. The syntax of the line-type list requires parentheses and line types separated by spaces (*type1 type2 ...typeN*). If you do not specify the LREF= option, the GCHART procedure uses the type specified by the AXIS statement's STYLE= option. If neither option is specified, the default line type is retrieved from the current style. However, if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these reference lines, use the CREF= option.

Alias LR=

Style reference LineStyle attribute of the GraphGridLines element

Restriction Not supported by Java

MAXIS=AXIS<1 …99>
assigns the specified AXIS definition to the midpoint axis. The MAXIS= option is ignored if the specified AXIS definition does not exist.

Restriction The Java and ActiveX devices do not support all AXIS statement options.

See "AXIS Statement" on page 399

"Understanding Midpoints" on page 993

Example "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

MEAN
displays the mean statistic in the table of statistics and above vertical bars. By default, the column heading in the table includes the name of the variable for which the mean is calculated. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ, CFREQ, PERCENT, CPERCENT, or SUM option is specified. The MEAN option is ignored unless you also use the SUMVAR= option.

Restriction The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

Interaction Specifying the MEAN option when creating an HBAR chart with an ActiveX device causes the means to display at the end of each bar instead of the frequencies.

See "About Chart Statistics" on page 996

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

MEANLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the MEAN statistic in the table of statistics. *column-label* can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify MEANLABEL=NONE.

Restriction Not supported by Java and ActiveX

Example "Example 6: Generating Error Bars in a Horizontal Bar Chart" on page 1107

MIDPOINTREF (HBAR and VBAR only)
draws a reference line at the midpoint between each bar produced by the HBAR and VBAR statements. To draw reference lines at specific points on the response axis, use the REF= option.

MIDPOINTS=*value-list*

specifies the midpoint values for the bars. The way you specify *value-list* depends on the type of the chart variable.

- For numeric chart variables, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - *n* < ...*n*>
  - *n* TO *n* <BY *increment*>
  - *n*< ...*n*> TO *n* <BY *increment*> <*n* < ...*n*>>

  If a numeric variable has an associated format, the specified values must be the *unformatted* values.

  If you omit the DISCRETE option, then numeric values are treated as continuous, which means that the following is true by default:

  - The lowest midpoint consolidates all data points from negative infinity to the median of the first two midpoints.
  - The highest midpoint consolidates all data points from the median of the last two midpoints up to infinity.
  - All other values in *value-list* specify the median of a range of values, and the GCHART procedure calculates the midpoint values.

  If you include the DISCRETE option, then each value in *value-list* specifies a unique numeric value.

- For character chart variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:

  'value-1' < ...*'value-n'*>

  If a character variable has an associated format, the specified values must be the *formatted* values.

  You are able to specify up to 256 characters for each variable value.

For a complete description of *value-list*, see " ORDER=(*value-list*)" on page 407 in the AXIS statement.

If the *value-list* for either type of variable specifies so many midpoints that the axis values overwrite each other, the values might be unreadable. In this case the procedure writes a warning to the SAS log. On many devices, this problem can be corrected by adjusting the size of the text with the HTEXT= graphics option. You can also correct the problem by increasing the number of cells in your graphics display. Do this by using the HPOS= and VPOS= graphics options.

The ORDER= option in the AXIS statement overrides the order specified in the MIDPOINTS= option. The bar chart statement options ASCENDING and DESCENDING also override both the MIDPOINTS= and ORDER= options in the AXIS statement.

Note Any character bar variable value with a long text string has the potential to cause a scaling issue and might produce a wider bar chart than desired.

See "Understanding Midpoints" on page 993

Example "Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart" on page 1103

MIDPOINTS=OLD
generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976). The MIDPOINTS=OLD option is ignored unless the chart variable is numeric.

MINOR=*number-of-minor-ticks*
specifies the number of minor tick marks between each major tick mark on the response axis. The MINOR= option in a bar chart statement overrides the number of minor tick marks specified in the MINOR= option in an AXIS definition. The AXIS definition is assigned to the response axis with the RAXIS= option.

MISSING
accepts a missing value as a valid midpoint for the chart variable. By default, observations with missing values are ignored. Missing values are always valid for group and subgroup variables.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 135.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output file name. See “About Filename Indexing” on page 137.
  - If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See “About Filename Indexing” on page 137.

Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GCHART

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

NOAXIS

suppresses all axes, including axis lines, axis labels, axis values, and all major and minor tick marks. If you specify an axis definition with the GAXIS, MAXIS=, or RAXIS= options, then the axes are generated as defined in the AXIS statement. But in this case all lines, labels, values, and tick marks are suppressed. Therefore, AXIS statement options such as ORDER=, LENGTH, and OFFSET= are still used.

To remove only selected axis elements such as lines, values or labels, use specific AXIS statement options.

The NOAXIS option does not suppress either the default frame or an axis area fill requested by the CFRAME= option. To remove the frame around the two-dimensional axis area or the three-dimensional back wall, use the NOFRAME option in the procedure. To remove the horizontal or vertical axis planes, use the NOPLANE option in the AXIS statement.

NOBASEREF

suppresses the zero reference line when the SUM or MEAN chart statistic has negative values.

NOLEGEND

suppresses the legend that is automatically generated by the SUBGROUP= option.

Alias NOSYMBOL

Interaction The NOLEGEND option is ignored if the SUBGROUP= option is not used.

NOSTATS (HBAR and HBAR3D only)

suppresses the table of statistics. The NOSTATS option suppresses both the default statistics and specific statistics requested by the FREQ, CFREQ, PERCENT, CPERCENT, SUM, and MEAN options.

Restriction Not supported by Java

NOSYMBOL

suppresses the legend that is automatically generated by the SUBGROUP= option.

Alias NOLEGEND

Interaction The NOSYMBOL option is ignored if the SUBGROUP= option is not used.

NOZERO

suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar. The NOZERO option is typically used with the GROUP= option. This combination suppresses midpoints when not all values of the chart variable are present for every group or if the chart statistic for the bar is 0.

Note There are instances when a bar is omitted and bar labels are specified with the VALUE= option in an AXIS statement. This might result in the labels being shifted and not displaying with the correct bar.

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

OUTSIDE=*statistic*

displays the values of the specified statistic above the bars.

For example, OUTSIDE=*PERCENT* displays the percentages of observations having a given value for the chart variable. For the Java and ActiveX devices, this option is valid for both horizontal and vertical bar charts. However, a restriction exists when specifying OUTSIDE=PERCENTSUM. Although it is supported by the ACTXIMG device it is not supported by the ACTIVEX and JAVAIMG devices. For other devices, this OUTSIDE= option is valid only for vertical bar charts. For subgrouped bar charts generated with the ACTIVEX device, you can display only one statistic for each bar. This device does not create both inside and outside bar labels. For graphs generated with the ACTIVEX device, the OUTSIDE= option overrides the INSIDE= option.

Statistic can be one of the following:

- FREQ
- CFREQ
- PERCENT | PCT
- CPERCENT | CPCT
- PERCENTSUM | PCTSUM
- SUM
- MEAN

See "About Chart Statistics" on page 996 for definitions of each of these statistics.

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

Examples "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

"Example 2: Creating a Drill-Down HTML Graph" on page 269

PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP

specifies how fill patterns are assigned. By default, PATTERNID=SUBGROUP. Values for PATTERNID= are as follows:

BY

changes patterns each time the value of the BY variable changes. All bars use the same pattern if the GCHART procedure does not include a BY statement.

GROUP

changes patterns every time the value of the group variable changes. All bars in each group use the same pattern, but a different pattern is used for each group.

MIDPOINT
changes patterns every time the midpoint value changes. If you use the GROUP= option, the respective midpoint patterns are repeated for each group.

SUBGROUP
changes patterns every time the value of the subgroup variable changes. The bars must be subdivided by the SUBGROUP= option for the SUBGROUP value to have an effect. Without the SUBGROUP= option, all bars have the same pattern.

Note If you use the SUBGROUP= option and specify a PATTERNID= value other than SUBGROUP, the bar segments use the same pattern and are indistinguishable.

See "Controlling Bar Chart Patterns, Colors, and Images" on page 1050

Examples "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

"Example 2: Creating a Drill-Down HTML Graph" on page 269

PERCENT
prints the percentages of observations having a given value for the chart variable in the table of statistics and above vertical bars. If a chart variable is not specified, percentages are calculated from the weighted contribution of each observation (frequency). Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ or CFREQ option is specified.

Alias PCT=

Restriction The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

Interaction Specifying the PERCENT option when creating an HBAR chart with an ActiveX device causes the percentages to display at the end of each bar instead of the frequencies.

See "About Chart Statistics" on page 996

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

PERCENTLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the PERCENT statistic in the table of statistics. *column-label* can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify PERCENTLABEL=NONE.

Restriction Not supported by Java and ActiveX

PERCENTSUM
calculates a percent of the sum variable for horizontal or vertical bar charts. The PERCENTSUM option is ignored if the SUMVAR= option is not specified.

Alias PCTSUM

Restriction When PERCENTSUM is the statistic specified by the INSIDE= option, it is not supported by any Java or ActiveX device. The specification of OUTSIDE=PERCENTSUM is supported by the ACTXIMG device but is not supported by the ACTIVEX and JAVAIMG devices.

See "About Chart Statistics" on page 996

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

RANGE

displays on the axis of the chart the range of numeric values represented by each bar. In the graphics output:

- The starting value of each range is indicated with the less-than symbol (<).
- The ending value is indicated with the greater-than-or-equal-to symbol (>=).
- The default midpoint axis label is "*chart-variable* RANGE."

The RANGE option has no effect on axes that represent character data. By default, the values shown on the axis are determined by the value of the "MIDPOINTS=*value-list*" on page 1036.

Interactions If specified, the "DISCRETE" option overrides the RANGE option.

During the creation of vertical bar charts, when options RANGE and LEVELS= are both used, RANGE displays the endpoints of the range for each vertical bar. It displays this rather than just the numerical midpoint of the range as would be displayed with the LEVELS= option alone. If necessary, RANGE automatically angles or rotates the values and they might be less readable.

If the RANGE, LEVELS=, and AXIS VALUE= options are all specified, the RANGE option is ignored and the LEVELS= option displays the numeric midpoint of each vertical bar.

RAXIS=*value-list* | AXIS<1 – 99>
AXIS=*value-list* | AXIS<1 – 99>

specifies values for the major tick mark divisions on the response axis or assigns the specified AXIS definition to the axis. See the "MIDPOINTS=*value-list*" on page 1036 for a description of *value-list*. By default, the GCHART procedure scales the response axis automatically and provides an appropriate number of tick marks.

You can specify negative values. But note that negative values are reasonable only when TYPE=SUM or TYPE=MEAN and one or more of the sums or means are less than 0. Frequency and percentage values are never less than 0.

For lists of values, a separate major tick mark is created for each individual value. A warning message is written to the SAS log if the values are not evenly spaced.

If the values represented by the bars are larger than the highest tick mark value, the bars are truncated at the highest tick mark.

If you use a BY statement with the PROC GCHART statement, then the same response axes are produced for each BY group when RAXIS=*value-list* is used. The same is true if there is an ORDER= list in the AXIS statement assigned to the response axis.

Restriction The Java and ActiveX devices do not support all AXIS statement options.

See the "AXIS Statement" on page 399 for more information.

Examples "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

"Example 2: Creating a Drill-Down HTML Graph" on page 269

REF=*value* | (*value*) | (*value-list*)
draws reference lines at the specified points on the response axis.

See the "MIDPOINTS=*value-list*" on page 1036 for a description of *value-list*.

Values can be listed in any order, but should be within the range of values represented by the response axis. A warning is written to the SAS log if any of the points are off of the axis, and no reference line is drawn for such points. You can use the AUTOREF option to draw reference lines automatically at all of the major tick marks.

By default, reference lines in three-dimensional bar charts are drawn on the back plane of the axis. To draw the reference lines in front of the bars, use the FRONTREF option.

SHAPE=*three-dimensional-bar-shape* (HBAR3D and VBAR3D only)
specifies the shape of the bars in charts that are produced with the HBAR3D and VBAR3D statements. *three-dimensional-bar-shape* can be one of the following:

- BLOCK | B (the default)
- CYLINDER | C
- HEXAGON | H
- PRISM | P
- STAR | S

Example "Example 2: Creating a Drill-Down HTML Graph" on page 269

SPACE=*bar-spacing*
specifies the amount of space between individual bars or between the bars within each group if you also use the GROUP= option. *Bar-spacing* can be any nonnegative number, including decimal values. Units are character cells. You cannot specify a unit of measure, such as inches or percent. By default, the GCHART procedure calculates spacing based on the size of the axis area and the width of the bars on the chart. Use SPACE=0 to leave no space between adjacent bars.

The SPACE= option is ignored if the following is true:

- You specify the WIDTH= option and are using the Java or ActiveX devices.
- The specified spacing requests a chart that is too large to fit in the space available for the midpoint axis. In this case, a warning message is issued.

Examples "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

"Example 2: Creating a Drill-Down HTML Graph" on page 269

STATFMT=*format-specification*
overrides the GCHART default format of the displayed statistical value.

Note: This option is valid starting with SAS 9.4M5.

The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use the STATFMT= option to change the default format that might contain decimal points, percentages, or commas.

If you specify options TYPE=MEAN, STATFMT=f6.2, INSIDE=SUM, and OUTSIDE=MEAN, the STATFMT=f6.2 option applies only to the calculated TYPE=MEAN statistical values. Here is an example:

```
footnote "Total revenue is shown inside each bar.";
proc gchart data=sashelp.electric(where=(year ge 2000));
   vbar year / discrete width=8 sumvar=Revenue
      type=mean statfmt=f6.2 inside=sum outside=mean;
run;
quit;
footnote;
```

In this case, the INSIDE= option values are displayed in their default format. They are unaffected by the STATFMT= option. The OUTSIDE= option values are displayed in the format that is specified in the STATFMT= option.

If you change the previous example to specify option TYPE=SUM, then the STATFMT=f6.2 option applies only to the calculated TYPE=SUM statistical values. In that case, the OUTSIDE= option values are displayed in their default format, and the INSIDE= option values are displayed in the format that is specified in the STATFMT= option.

Alias SFMT=, SFORMAT=, STATFORMAT=

Note The STATFMT= option overrides the format of the response-axis tick values.

SUBGROUP=*subgroup-variable*
divides the bars into segments according to the values of *subgroup-variable*. *Subgroup-variable* can be either character or numeric and is always treated as a discrete variable. SUBGROUP= creates a separate segment within each bar for every unique value of the subgroup variable for that midpoint.

If PATTERNID=SUBGROUP (the default setting), each segment is filled with a different pattern and a legend that provides a key to the patterns is automatically generated. If the value of PATTERNID= is anything other than SUBGROUP, the segments are all the same color, the legend is suppressed, and the subgrouping effect is lost.

By default the legend appears at the bottom of the chart. To modify the legend, assign a LEGEND definition with the LEGEND= option. To suppress the legend, specify NOLEGEND.

See "LEGEND Statement" on page 432

Examples "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

"Example 2: Creating a Drill-Down HTML Graph" on page 269

"Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart" on page 1103

SUBOUTSIDE=*statistic* (VBAR and VBAR3D only)

displays a list of the subgroup values of the specified statistic above the bars. The values are listed vertically above each bar in the order in which they occur on the bar from the top down. This option is valid for VBAR and VBAR3D charts only. The Java and ActiveX devices do not support this option. For all of the other devices:

- SUBOUTSIDE= overrides OUTSIDE=.
- If you specify both the INSIDE= and SUBOUTSIDE= options, use the same statistic for both. If you specify a different statistic for the SUBOUTSIDE= option, SAS/GRAPH uses the INSIDE= statistic for SUBOUTSIDE= instead.

Statistic can be one of the following:

- FREQ
- CFREQ
- PERCENT | PCT
- CPERCENT | CPCT
- SUM

Restriction Not supported by Java and ActiveX

See "About Chart Statistics" on page 996

"Displaying Statistics in Vertical Bar Charts" on page 1049

SUM

displays the sum statistic in the table of statistics and above vertical bars. By default, the column heading in the table includes the name of the variable for which the sum is calculated. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ, CFREQ, PERCENT, or CPERCENT option is specified. SUM is ignored unless you also use the SUMVAR= option.

Restriction The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

Interaction Specifying the SUM option when creating an HBAR chart with an ActiveX device causes the sums to display at the end of each bar instead of the frequencies.

See "About Chart Statistics" on page 996

"Displaying Statistics in Horizontal Bar Charts" on page 1049

"Displaying Statistics in Vertical Bar Charts" on page 1049

SUMLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the SUM statistic in the table of statistics. *Column-label* can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify SUMLABEL=NONE.

Restriction Not supported by Java and ActiveX

SUMVAR=*numeric-variable*
specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of *numeric-variable* for each midpoint. The resulting statistics are represented by the length of the bars along the response axis, and they are displayed at major tick marks.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

Examples "Example 3: Specifying the Sum Statistic in Bar Charts" on page 1098

"Example 6: Generating Error Bars in a Horizontal Bar Chart" on page 1107

TYPE=*statistic*
specifies the chart statistic.

- If the SUMVAR= option is not used, *statistic* can be one of the following:

FREQ
frequency (the default)

CFREQ
cumulative frequency

PERCENT PCT
percentage

CPERCENT CPCT
cumulative percentage

- If the SUMVAR= option is used, *statistic* can be either of the following:

SUM
sum (the default)

MEAN
mean

You cannot use TYPE=FREQ, TYPE=CFREQ, TYPE=PERCENT, or TYPE=CPERCENT with the SUMVAR= option. Therefore, you must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum. See also "Calculating Weighted Statistics" on page 997.

If you specify TYPE=MEAN and use the SUBGROUP= option, the height or length of the bar represents the mean for the entire midpoint. The subgroup segments are proportional to the subgroup's contribution to the sum for the bar. See also the option description for "SUBGROUP=*subgroup-variable*" on page 1043.

See "About Chart Statistics" on page 996 for a complete description of statistic types

Example "Example 6: Generating Error Bars in a Horizontal Bar Chart" on page 1107

URL=*character-variable*

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction This option affects graphics output that is created through the ODS HTML destination only.

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored.

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

WAUTOREF=*reference-line-width*

specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CAUTOREF= option.

Style reference LineThickness attribute of the GraphGridLines element

WGROUPREF=*reference-line-width*

specifies the line width for reference lines between groups of bars produced by the HBAR and VBAR statements. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CGROUPREF= option. To specify a line type for these reference lines, use the LGROUPREF= option.

Style reference LineThickness attribute of the GraphGridLines element

WMIDPOINTREF=*reference-line-width*

specifies the line width for midpoint reference lines between bars produced by the HBAR and VBAR statements. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CMIDPOINTREF= option. To specify a line type for these reference lines, use the LMIDPOINTREF= option.

Style reference LineThickness attribute of the GraphGridLines element

WIDTH=*bar-width*

specifies the width of the bars. By default, the GCHART procedure selects a bar width that accommodates the midpoint values displayed on the midpoint axis using a hardware font and a height of one cell. Units for *bar-width* are character cells. The value for *bar-width* must be greater than 0, but it does not have to be an integer, for example:

```
vbar site / width=1.5;
```

Sometimes the requested bar width results in a chart that is too large to fit in the space available for the midpoint axis. In this case the procedure issues a warning in the log and ignores the WIDTH= option. If the specified width is too narrow, the procedure displays the midpoint values vertically.

Example "Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart" on page 1100

WOUTLINE=*bar-outline-width*
specifies the width of the outline in pixels. The WOUTLINE= option affects both the bar and subgroup outlines.

Style reference LineThickness attribute of the GraphOutlines element

Restriction Not supported by Java

WREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines. Line widths are specified as whole numbers. Specifying a line width without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the WAUTOREF= option overrides WREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the REF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the REF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the CREF= option.

Style reference LineThickness attribute of the GraphReference element

Restriction Not supported by Java

Details

Description

The HBAR, HBAR3D, VBAR, and VBAR3D statements specify the variable or variables that define the categories of data to chart. These statements automatically do the following:

- determine the midpoints.
- calculate the chart statistic for each midpoint (the default is FREQ).
- scale the response axis and the bars according to the statistic value.
- determine bar width and spacing.
- assign patterns to the bars; the default bar or block pattern is SOLID.
- draw a frame around the axis area using a color determined by the current style. Or, if the NOGSTYLE option is specified, use the first color in the device's color list.

You can use statement options to do the following:

- select or order the midpoints (bars)
- control the tick marks on the response axis
- change the type of chart statistic
- display specific statistics
- modify the appearance of the chart.

You can also specify additional variables by which to group, subgroup, or sum the data.

All bar charts allow grouping, which uses an additional category to organize the bars into groups, and subgrouping, which divides the bars into segments.

In addition, you can do the following:

- use global statements to modify the axes (including requesting a logarithmic axis), the bar patterns, and the legend. See "SAS/GRAPH Statements" on page 397 for more information.
- add titles, footnotes, and notes to the chart. See the "TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement" on page 507 for more information.
- use an Annotate data set to enhance the chart. See Chapter 28, "Using Annotate Data Sets," on page 721 for more information.
- display an image in the axis area or back wall. See the option description for "IFRAME=*fileref* | '*external-file*'" on page 1031.
- display images in the bars of an HBAR or VBAR chart, and with a device of ACTIVEX or ACTXIMG, in the bars of an HBAR3D or VBAR3D chart. See the option description for "IMAGE=*fileref* | "*external-file*"" on page 456 in the PATTERN statement.

The Chart Statistic and the Response Axis

Where Chart Statistic Values Are Displayed

In bar charts, the scale of values of the chart statistic is displayed on the response axis. By default, the response axis is divided into evenly spaced intervals identified with major tick marks that are labeled with the corresponding statistic value. Minor tick marks are evenly distributed between the major tick marks unless a log axis has been requested. For sum and mean statistics, the major tick marks are labeled with values of the SUMVAR= variable (formatted if the variable has an associated format). The response axis is also labeled with the statistic type.

Specifying Logarithmic Axes

Logarithmic axes can be specified with the AXIS statement. See the "AXIS Statement" on page 399 for a complete discussion.

Displaying Statistics in Horizontal Bar Charts

Details

For graphs generated with the Java and ActiveX devices, default statistics are not generated, but you can display one statistic at the end of each bar. To specify the statistic, specify the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, or MEAN option.

For graphs generated with other devices, the HBAR and HBAR3D statements print a table of statistic values to the right of the bars. When the value of TYPE= is FREQ, CFREQ, PERCENT, or CPERCENT, the frequency, cumulative frequency, percentage, and cumulative percentage statistics are printed next to the bars by default. When TYPE=SUM, the frequency and sum statistic values are printed by default. When TYPE=MEAN, the frequency and mean statistic values are printed by default. However, if you use the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, or MEAN options to request specific statistics, the default statistics are not printed.

For sum and mean, the name of the SUMVAR= variable is added to the heading for the column of values.

Specifying the Table of Statistics

You can use the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, and MEAN options to select only certain statistics. Without the SUMVAR= option, only the frequency, cumulative frequency, percentage, and cumulative percentage statistics can be printed. With the SUMVAR= option, all statistics, including the sum and mean, can be printed. You can suppress all statistics with the NOSTATS option.

To change the column labels for any statistic in the table, use one or more of the statistic column label options: FREQLABEL=, CFREQLABEL=, PERCENTLABEL=, CPERCENTLABEL=, SUMLABEL=, and MEANLABEL=.

To control the font and size of the text in the table of statistics, use the graphics options “FTEXT Graphics Option” on page 626 and “HTEXT Graphics Option” on page 650.

Displaying Statistics in Vertical Bar Charts

Statistic values on vertical bar charts are not printed by default. You must explicitly request a statistic with the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, MEAN, INSIDE=, OUTSIDE=, or SUBOUTSIDE= option.

For graphs generated with the Java and ActiveX devices, you can display one statistic for each bar. For graphs generated with other devices, you can display up to two statistics. Statistics can be displayed either above the bars or inside the bars.

To specify a statistic that you want to display above the bars, specify the statistic option (FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, or MEAN) or specify OUTSIDE=*statistic*. When subgrouping is used, to display a list of the subgroup statistics above each bar, specify SUBOUTSIDE=*statistic*. To specify a statistic that you want to display inside the bars, specify INSIDE=*statistic*.

For graphs generated with the Java and ActiveX devices, the OUTSIDE= option overrides INSIDE=, and INSIDE= overrides the FREQ, CFREQ, PERCENT,

CPERCENT, PERCENTSUM, SUM, and MEAN options. For graphs generated with other devices, the individual statistic options and the SUBOUTSIDE= option override the OUTSIDE= option.

If more than one statistic option is specified, only the highest priority statistic is displayed. The priority order, from highest to lowest, is as follows:

1. FREQ
2. CFREQ
3. PERCENT
4. CPERCENT
5. PERCENTSUM
6. SUM
7. MEAN

The bars must be wide enough to accommodate the text. You can adjust the width of the bars with the WIDTH= option. To control the font and size of the text, use the graphics options "FTEXT Graphics Option" on page 626 and "HTEXT Graphics Option" on page 650.

Ordering and Selecting Midpoints

To rearrange character or discrete numeric midpoint values or to select ranges for numeric values, use the MIDPOINTS= option. Changing the number of midpoints for numeric variables can change the range of values for individual midpoints. Note that it does not change the range of values for the chart as a whole. For details see "Understanding Midpoints" on page 993.

Like the MIDPOINTS= option, the ORDER= option in the AXIS statement can rearrange the order of the midpoints or suppress the display of discrete numeric or character values. However, the ORDER= option cannot calculate the midpoints for a continuous numeric variable, or exclude values from the calculations. For details, see the description of the "*value-list*" on page 408 in the AXIS statement.

Controlling Bar Chart Patterns, Colors, and Images

Default Patterns and Outlines

Each bar in a bar chart is filled with a pattern. Because the system option, GSTYLE, is in effect by default, the procedure uses the style's default patterns and outlines when producing output. By default, the procedure does the following:

- fills the bars with bar patterns, beginning with the default fill, SOLID, and rotates it through the list of colors available in the default style. When these colors are exhausted, the procedure a slightly modified version of the original color list. It continues in this fashion until each of the chart variables have been assigned a unique pattern.

  If you use the default style colors and the first color in the list is either black or white, then the procedure does not create a pattern in that color. If you specify a

color list with the COLORS= graphics option, the procedure uses all the colors in the list to generate the patterns.

- outlines bars and bar segments using the color defined by the style.

See "About Patterns" on page 998 for more information about how the GCHART procedure assigns default patterns and outlines.

User-Defined Patterns

To override the default patterns and select fills and colors for the bars or bar segments, use the PATTERN statement. Only bar or block patterns are valid; all other pattern fills are ignored. For a complete description of all bar or block patterns, see " VALUE=*bar/block-pattern*" on page 457 in the PATTERN statement.

Whenever you use PATTERN statements, the default pattern outline color is that of the current style. Only when the EMPTY pattern is used does the pattern change to SAME. That is, the outline color is the same as the fill color. To specify the outline color, use "COUTLINE=*bar-outline-color* | SAME" on page 1024.

When Patterns Change

The PATTERNID= option controls when the pattern changes. By default, PATTERNID=SUBGROUP. Therefore, when you use the SUBGROUP= option to subdivide the bars, the pattern automatically changes each time the subgroup value changes. Also, each subdivision of the bar displays a different pattern. As a result, the number of values for the SUBGROUP= variable determines the number of bar patterns on the chart. If you do not subdivide the bars, all bars use the same pattern.

Instead of changing the pattern for each subgroup, you can change the pattern for each midpoint, each group, or each BY group by changing the value of PATTERNID=. See the description for option "PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP" on page 1039 for details.

Axis Color

By default, axis elements use the colors specified in the current style or the colors that are specified by AXIS statement color options. However, axis statement options can also control the color of the axis lines, text, and frame.

To change the color of ...	Use this option ...
the axis text	CTEXT=
the axis lines	CAXIS=
the area within the frame	CFRAME=

Adding Images to Bar Charts

You can apply images to the bars and to the axis area of two-dimensional bar charts developed with the HBAR and VBAR statements. In three–dimensional bar charts, you can apply images to the back wall, and to the bars when using certain interactive devices. For details, see Chapter 24, “Adding Images to SAS/GRAPH Output,” on page 383.

PIE, PIE3D, and DONUT Statement

create pie or donut charts. The size of a pie slice represents the value of the chart statistic for that category of data in relation to the total chart statistic for all categories.

Requirement:	At least one chart variable is required.
Supported Statements:	FOOTNOTE, LEGEND, PATTERN, TITLE
Supports:	Drill-down functionality

Syntax

PIE | PIE3D | DONUT *chart-variable(s)* < / *options* >;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate all graphs that are produced by the GCHART procedure.

CFILL=*fill-color*
: specifies one color for all patterns in the chart, regardless of whether the fill is solid or hatch.

COUTLINE=*slice-outline-color* | SAME
: outlines all slices, rings (subgroups), and legend values (if a legend appears) in the specified color.

DETAIL_RADIUS=*percent* (PIE and DONUT only)
: determines the size of the detail pie.

EXPLODE=*value-list* | ALL
: pulls the specified slices slightly out from the rest of the pie for added emphasis.

FILL=SOLID | X
: specifies the fill pattern for all slices in the chart.

INVISIBLE=*value-list*
: makes the specified slices invisible, as if they had been removed from the pie.

NOHEADING

suppresses the heading that is normally printed at the top of each page or display of output for all devices except Java and ActiveX.

RADIUS=*value*
specifies the radius of the pie and donut in their respective charts.

WOUTLINE=*slice-outline-width*
specifies the width of the outline in pixels.

Catalog entry description options

DESCRIPTION="*description*"
specifies a description of the output.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Detail pie options

DETAIL_THRESHOLD=*percent* (PIE and DONUT only)
determines whether a detail slice is included in the inner pie.

DETAIL=*variable* (PIE and DONUT only)
produces an inner pie overlay whose slices show the major components that comprise the outer pie's slice.

Detail pie slice-labeling options

DETAIL_PERCENT=BEST | NONE (PIE and DONUT only)
specifies the algorithm to use for displaying the percentage values for the detail pie slices.

DETAIL_SLICE=BEST | NONE (PIE and DONUT only)
specifies the algorithm to use for displaying the detail variable labels for the inner pie slices.

DETAIL_VALUE=BEST | NONE (PIE and DONUT only)
specifies the algorithm to use for displaying the data values for the detail pie slices.

Donut-labeling options

DONUTPCT=*percent* (DONUT only)
specifies the size of the donut hole in percent of the radius of the whole chart.

LABEL=(*text argument(s)*) (DONUT only)
defines the text that is displayed in the donut hole.

Grouping and subgrouping options

ACROSS=*number-of-columns*
draws *number-of-columns* pies across the procedure output area.

DOWN=*number-of-rows*
draws *number-of-rows* pies vertically in the procedure output area.

GROUP=*group-variable*
organizes the data according to values of *group-variable* and produces a separate pie (or donut) chart for each unique value of *group-variable*.

NOGROUPHEADING
suppresses the headings that are normally printed above each pie when you use the GROUP= option.

SUBGROUP=*subgroup-variable*

divides the chart into concentric rings according to the values of *subgroup-variable*.

Midpoint options

DISCRETE
: treats a numeric chart variable as a discrete variable rather than as a continuous variable.

LEVELS=*number-of-midpoints* | ALL
: specifies the number of midpoints to be graphed for a chart variable.

MIDPOINTS=OLD
: generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

MIDPOINTS=*value-list*
: specifies the midpoint values for the slices.

MISSING
: accepts a missing value as a valid midpoint for the chart variable.

OTHER=*percent-of-total*
: collects all midpoints with chart statistic values less than or equal to *percent-of-total* into a generic midpoint named OTHER.

ODS options

HTML_LEGEND=*variable*
: identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
: identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
: specifies a character variable whose values are URLs.

Slice-labeling options

CTEXT=*text-color*
: specifies a color for all text on the axes and legend.

LEGEND | LEGEND=LEGEND<1 …99>
: generates a legend for the slice names (midpoint values) instead of printing them beside the slices.

MATCHCOLOR
: uses the slice pattern color for all slice labels.

NOLEGEND
: suppresses the legend that is automatically generated by the SUBGROUP= option.

OTHERCOLOR=*color*
: specifies the color to use for the OTHER slice.

OTHERLABEL='*text-string*'
: specifies a text string up to 16 characters for the label for the OTHER slice.

PERCENT=ARROW | INSIDE | NONE | OUTSIDE
: prints the percentage represented by each slice using the specified labeling method.

PLABEL=(*text-options*)
: modifies the font, height, and color of the pie slice labels.

PPERCENT=(*text-options*)

modifies the font, height, and color of the displayed percentages associated with the pie slice labels.

SLICE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice name (midpoint value) for each slice.

VALUE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice value (chart statistic) for each slice.

Slice-ordering options

ANGLE=*degrees*
starts the first slice at the specified angle.

ASCENDING
arranges the slices in ascending order of the value of the chart statistic.

CLOCKWISE
draws the slices clockwise starting at the twelve o'clock position.

DESCENDING
arranges the slices in descending order of the value of the chart statistic.

JSTYLE
arranges the midpoints in descending order of the statistic value and draws the slices clockwise starting at the twelve o'clock position.

Statistic options

FREQ=*numeric-variable*
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

STATFMT=*format-specification*
overrides the GCHART default format of the displayed statistical value.

SUMVAR=*numeric-variable*
specifies a numeric variable for sum or mean calculations.

TYPE=*statistic*
specifies the chart statistic.

Required Argument

chart-variable(s)
specifies one or more variables that define the categories of data to chart. Each chart variable draws a separate chart. All variables must be in the input data set. Separate multiple chart variables with blanks.

See "Understanding Chart Variables" on page 992

Optional Arguments

Options in a PIE, PIE3D, or DONUT statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order. For details about specifying colors, see Chapter 23, "Using Colors in SAS/GRAPH Programs,"

on page 363. For a complete description of the graphics options, see "Graphics Options and Device Parameters Dictionary" on page 577.

ACROSS=*number-of-columns*
draws *number-of-columns* pies across the procedure output area. ACROSS is ignored unless you also use the GROUP= option.

If *number-of-columns* calls for more pies than fit horizontally in the graphics output area, no pies are drawn and an error message is written to the SAS log.

If you also use the DOWN= option, the pies are drawn in left-to-right and top-to-bottom order.

ANGLE=*degrees*
starts the first slice at the specified angle. A value of 0 for *degrees* corresponds to the three o'clock position. *Degrees* can be either positive or negative. Positive values move the starting position in the counterclockwise direction; negative values move the starting position clockwise. By default, ANGLE=0. Successive slices are drawn counterclockwise from the starting slice.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

Note Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block, pie, donut, or star charts.

See Chapter 28, "Using Annotate Data Sets," on page 721

ASCENDING
arranges the slices in ascending order of the value of the chart statistic. By default, slices are arranged in ascending order of midpoint value, without regard to size. The ASCENDING option reorders the slices from smallest to largest. The OTHER slice is still last regardless of its size.

Interactions If you also use the GROUP= option, the reordering is performed separately for each group, so the order of the midpoint values might be different for each pie or donut.

The ASCENDING option overrides any midpoint order that is specified with the MIDPOINTS= option.

CFILL=*fill-color*
specifies one color for all patterns in the chart, regardless of whether the fill is solid or hatch. For the PIE3D statement, the fill is always solid. For the PIE and DONUT statements, it is possible that no pattern is specified in the pattern statement or with the FILL= option. In this case the procedure starts with the default solid fill and then, beginning with P2N0, uses each default pie hatch pattern with the specified color. For the outline color, the procedure uses the default color, which is retrieved from the current style. However, if the NOGSTYLE option is specified, the procedure uses the first color in the device's color list. Use the COUTLINE= option to specify a different outline color. The CFILL= option overrides any other pattern color specification and controls the color of all slices.

Style reference Color attribute of the GraphData1 element

See "Controlling Bar Chart Patterns, Colors, and Images" on page 1050

"About Patterns" on page 998

Example "Example 9: Ordering and Labeling Slices in a Pie Chart" on page 1115

CLOCKWISE
draws the slices clockwise starting at the twelve o'clock position. Although this position implies ANGLE=90, you can use the ANGLE= option to specify a different starting angle.

COUTLINE=*slice-outline-color* | SAME
outlines all slices, rings (subgroups), and legend values (if a legend appears) in the specified color. SAME specifies that the outline color of a slice or a slice segment or a legend value is the same as the interior pattern color.

The default outline color depends in the PATTERN statement:

- If you do not specify a PATTERN statement, the default outline color is the color of the current device.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

Alias CO=

Style reference Color attribute of the GraphOutlines element

Note Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

See "Controlling Slice Patterns and Colors" on page 1074

"About Patterns" on page 998

Examples "Example 7: Specifying the Sum Statistic for a Pie Chart" on page 1109

"Example 8: Subgrouping a Donut or Pie Chart" on page 1112

CTEXT=*text-color*
specifies a color for all text on the axes and legend. This includes axis labels, tick mark values, legend labels, and legend value descriptions. The GCHART procedure looks for the text color in the following order:

1. the colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified on either the PIE or DONUT statement.
2. the color specified by the CTEXT= option in the PIE or DONUT statement.
3. the color specified by the CTEXT= option in a GOPTIONS statement.

4 the color specified in the current style. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. The COLOR= color is still used for all other axis colors, such as tick marks.

Alias	CT=
Style reference	Color attributes of the GraphValueText and the GraphLabelText elements
Note	If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.
Example	"Example 8: Subgrouping a Donut or Pie Chart" on page 1112

DESCENDING

arranges the slices in descending order of the value of the chart statistic. By default, slices are arranged in ascending order of alphabetical or numeric midpoint value, without regard to size or summary statistic. DESCENDING reorders the slices from largest to smallest. The OTHER slice is still last, regardless of its size.

If you also use the GROUP= option, the reordering is performed separately for each group, so the order of midpoint values might be different for each pie or donut.

DESCENDING overrides any midpoint order that is specified with the MIDPOINTS= option.

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the SAS Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256

characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default Based on the chart type, either PIE CHART or DONUT CHART OF *chart-variable*

DETAIL=*variable* (PIE and DONUT only)
produces an inner pie overlay whose slices show the major components that comprise the outer pie's slice. *Variable* is the variable whose values are used to construct the detail pie. Detail pie slices are always displayed in descending order. If you specify the DETAIL= option and either GROUP= or SUBGROUP=, then the DETAIL= option is ignored.

DETAIL_PERCENT=BEST | NONE (PIE and DONUT only)
specifies the algorithm to use for displaying the percentage values for the detail pie slices. NONE turns off the display of the percentage values.

DETAIL_RADIUS=*percent* (PIE and DONUT only)
determines the size of the detail pie. *Percent* specifies the percent of the outer pie radius to use as the detail pie radius. The valid range is 25 to 90. The default is 75.

DETAIL_SLICE=BEST | NONE (PIE and DONUT only)
specifies the algorithm to use for displaying the detail variable labels for the inner pie slices. NONE turns off the display of the detail variable labels.

DETAIL_THRESHOLD=*percent* (PIE and DONUT only)
determines whether a detail slice is included in the inner pie. Any detail slice comprising *percent* or more percent of the whole pie is included. The valid range for *percent* is 0 to 75. The default is 4.

DETAIL_VALUE=BEST | NONE (PIE and DONUT only)
specifies the algorithm to use for displaying the data values for the detail pie slices. NONE turns off the display of the data values.

DISCRETE
treats a numeric chart variable as a discrete variable rather than as a continuous variable. The GCHART procedure creates a separate midpoint and, hence, a separate slice for each unique value of the chart variable. If the chart variable has a format that is associated with it, each formatted value is treated as a midpoint.

The LEVELS=*number-of-midpoints* option is ignored when you use the DISCRETE option. The MIDPOINTS= option overrides the DISCRETE option.

DONUTPCT=*percent* (DONUT only)
specifies the size of the donut hole in percent of the radius of the whole chart. Values of *percent* range from 0 to 99. By default, DONUTPCT=25.

Example “Example 8: Subgrouping a Donut or Pie Chart” on page 1112

DOWN=*number-of-rows*
draws *number-of-rows* pies vertically in the procedure output area. The DOWN= option is ignored unless you also use the GROUP= option.

If *number-of-rows* calls for more pies than fit vertically in the graphics area of the output device, no pies are drawn and an error message is written to the SAS log.

If you also use the ACROSS= option, the pies are drawn in left-to-right and top-to-bottom order.

EXPLODE=*value-list* | ALL

pulls the specified slices slightly out from the rest of the pie for added emphasis. *Value-list* is the list of midpoint values for the slices to be exploded. For a description of *value-list*, see "MIDPOINTS=*value-list*" on page 1064.

The values in the value list must match the existing midpoints exactly, including the case of character midpoints. Any values in the list that do not correspond to existing midpoints are ignored.

Using EXPLODE=ALL pulls all of the slices outward from the center of the pie.

When you use the EXPLODE= option, the radius is reduced to allow room for exploded slices.

Restriction EXPLODE=ALL is not supported by Java and ActiveX for the slice labeled 'Other'.

Interaction When used with subgroups, the EXPLODE= option is supported only by the ActiveX and Java devices.

Example "Example 7: Specifying the Sum Statistic for a Pie Chart" on page 1109

FILL=SOLID | X

specifies the fill pattern for all slices in the chart:

SOLID

rotates a solid fill through the color list of the current style as many times as necessary. SOLID is the default.

Alias S

X

rotates a single hatch pattern through the list of colors defined in the current style. If the NOGSTYLE option is specified, it rotates the hatch pattern through the device color list as many times as necessary. If you do not specify the colors= goption, the fill skips the first color in the color list.

By default, the outline color is the color defined by the current style. If the NOGSTYLE option is specified, the outline color is the first color in the device's color list . If PATTERN statements are used to specify colors, the slice outline color matches the slice fill color.

Style reference Color attribute of the GraphData1 element

Restriction Partially supported by Java and ActiveX

Interactions FILL= overrides any pattern that is specified in PATTERN statements.

If any PATTERN statements have been defined, the colors in the PATTERN definitions are used, in order, before the default style color rotation.

See "Controlling Bar Chart Patterns, Colors, and Images" on page 1050

"PATTERN Statement" on page 455

FREQ=*numeric-variable*

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times specified by the value of *numeric-variable* for that observation. If the value of *numeric-variable* is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of *numeric-variable* are truncated to integers.

FREQ= is valid with all chart statistics.

Because you cannot use TYPE=PERCENT or TYPE=FREQ with the SUMVAR= option, you must use the FREQ= option to calculate percentages and frequencies based on a sum.

The statistics are not affected by applying a format to *numeric-variable*.

See "Calculating Weighted Statistics" on page 997

GROUP=*group-variable*

organizes the data according to values of *group-variable* and produces a separate pie (or donut) chart for each unique value of *group-variable*. *Group-variable* can be either character or numeric and is always treated as a discrete variable. Missing values for *group-variable* are treated as a valid group. By default, each group includes only those midpoints with nonzero chart statistic values.

By default, the charts are produced in ascending order of group variable value and each is drawn on a separate page or display. Therefore, the effect of the GROUP= option is essentially the same as using a BY statement. The exception is that the GROUP= option causes the midpoints with the same value to use the same color and fill pattern. To place more than one pie on a page or display, use the ACROSS= or DOWN= options, or both.

See "BY Statement" on page 424

Example "Example 10: Grouping and Arranging Pie Charts" on page 1116

HTML=*variable*

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restrictions Not supported by Java and ActiveX

If either subgroups or the DETAIL= option are specified, then the HTML_LEGEND= option is ignored.

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

INVISIBLE=*value-list*
makes the specified slices invisible, as if they had been removed from the pie. Labels are not printed for invisible slices. *Value-list* is the list of midpoint values for the invisible slices. See the "MIDPOINTS=*value-list*" on page 1064 for a description of *value-list*.

The values in the value list must match the existing midpoints exactly, including the case of character midpoints. Any values in the list that do not correspond to existing midpoints are ignored.

JSTYLE
arranges the midpoints in descending order of the statistic value and draws the slices clockwise starting at the twelve o'clock position. The JSTYLE option has the same effect as specifying both the DESCENDING and CLOCKWISE options.

LABEL=(*text argument(s)*) (DONUT only)
defines the text that is displayed in the donut hole. *Text-argument(s)* defines the text or the appearance of the label, or both. *Text-argument(s)* can be one or more of the following:

'text-string'
provides the text of the label. Enclose each string in quotation marks. Separate multiple strings with blanks.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVAL*n*
#BYVAL(*BY-variable-name*)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples "Example 7: Using BY-group Processing to Generate a Series of Charts" on page 551

"Example 8: Combining Graphs and Reports in a Web Page" on page 557

#BYVAR*n*
#BYVAR(*BY-variable-name*)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the

text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 1076

text-description-suboption
modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be

- ANGLE=*degrees*
- COLOR=*color*
- FONT=*font*
- HEIGHT=*text-height* <*units*>
- JUSTIFY=LEFT | CENTER | RIGHT
- ROTATE=*degrees*

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Restrictions The Java and ActiveX devices do not support all of the suboptions. See "Text Description Suboptions for Donut" on page 1071 for a complete description.

#BYVAL or #BYVAR substitution in a text string is not available in the DATA Step Graphics Interface or in the Annotate facility. This is because BY lines are not created in a DATA step.

Example "Example 8: Subgrouping a Donut or Pie Chart" on page 1112

LEGEND | LEGEND=LEGEND<1 …99>
generates a legend for the slice names (midpoint values) instead of printing them beside the slices. The legend displays each slice name and its associated pattern. This option also suppresses the display of the chart statistic values. To display the chart statistics, use the VALUE= option.

If you use the SUBGROUP= option, the legend is automatically generated. However, because patterning is always by midpoint, the legend still describes the midpoint values, not the subgroups.

Specifying LEGEND=LEGEND*n* assigns the specified LEGEND statement to the legend.

Restriction The Java and ActiveX devices do not support all LEGEND statement options. See the "LEGEND Statement" on page 432 for more information.

Note If you request a legend and the slices use hatch patterns, the patterns in the slices are oriented to be visually equivalent to the legend.

See "LEGEND Statement" on page 432

Example "Example 8: Subgrouping a Donut or Pie Chart" on page 1112

LEVELS=*number-of-midpoints* | ALL
specifies the number of midpoints to be graphed for a chart variable. After you specify the number of midpoints that you want, the range for each numeric midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985).

If you specify LEVELS=ALL, then all unique numeric or character midpoint values are graphed. You might have instances where the data contains a large number of unique midpoint values (more than 200). In this case you can use the XPIXELS and YPIXELS GOPTIONS to enable the device driver to render a larger (and more readable) graph.

The LEVELS=*number-of-midpoints* option is ignored if any of the following is true:

- The chart variable is character type.
- The MIDPOINTS= option is used.

MATCHCOLOR
uses the slice pattern color for all slice labels. MATCHCOLOR overrides the color that is specified in the CTEXT= option.

MIDPOINTS=*value-list*
specifies the midpoint values for the slices. The way you specify *value-list* depends on the type of variable:

- For numeric chart variables, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - *n* < ...*n*>
  - *n* TO *n* <BY *increment*>
  - <*n* ...> *n* TO *n* <BY *increment*> <*n* < ...*n*>>

  If a numeric variable has an associated format, the specified values must be the *unformatted* values.

  If you omit the DISCRETE option, then numeric values are treated as continuous, which means that the following is true by default:

  - The lowest midpoint consolidates all data points from negative infinity to the median of the first two midpoints.
  - The highest midpoint consolidates all data points from the median of the last two midpoints up to infinity.
  - All other values in *value-list* specify the median of a range of values, and the GCHART procedure calculates the midpoint values.

    If you include the DISCRETE option, then each value in *value-list* specifies a unique numeric value.

- For character chart variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:

  'value-1' < ...'value-n'>

  If a character variable has an associated format, the specified values must be the *formatted* values.

  You are able to specify up to 256 characters for each variable value.

For a complete description of *value-list*, see the "*value-list*" on page 408 in the AXIS statement.

Midpoints that represent small percentages are collected into a generic midpoint named OTHER. See the descriptions of "OTHER=*percent-of-total*" on page 1066 and "OTHERLABEL='*text-string*'" on page 1067 for more information.

See "Understanding Midpoints" on page 993

Example "Example 9: Ordering and Labeling Slices in a Pie Chart" on page 1115

MIDPOINTS=OLD
generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976). The MIDPOINTS=OLD option is ignored unless the chart variable is numeric

MISSING
accepts a missing value as a valid midpoint for the chart variable. By default, observations with a missing value are ignored. Missing values are always valid for the group and subgroup variable.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
  - All characters are represented in lowercase.

- If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
- If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

  Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GCHART

NOGROUPHEADING
suppresses the headings that are normally printed above each pie when you use the GROUP= option.

NOHEADING
suppresses the heading that is normally printed at the top of each page or display of output for all devices except Java and ActiveX. For the Java and ActiveX devices, NOHEADING is the default.

Restriction Not supported by Java and ActiveX

Example "Example 8: Subgrouping a Donut or Pie Chart" on page 1112

NOLEGEND
suppresses the legend that is automatically generated by the SUBGROUP= option. NOLEGEND is ignored if the SUBGROUP= option is not used.

OTHER=*percent-of-total*
collects all midpoints with chart statistic values less than or equal to *percent-of-total* into a generic midpoint named OTHER. The value of *percent-of-total* can be 0 to 100; the default value is 4. Therefore, any slice that represents 4% or less of the total is put in the OTHER category.

The OTHER slice is the last slice in the pie, regardless of the order of the slices. (In other words, it is the slice immediately before the starting slice.)

If only one midpoint falls into the OTHER category, its slice is displayed in its normal position in the pie and retains its original label. For example, suppose a pie has these slices and percent values: Coal 35%, Gas 15%, Hydro 5%, and Oil 45%. If you specify OTHER=5, Hydro remains the third slice instead of becoming the last slice.

Note If you specify a small value for *percent-of-total*, the GCHART procedure might not be able to label all of the small slices.

OTHERCOLOR=*color*
specifies the color to use for the OTHER slice. If you omit the OTHERCOLOR= option, GCHART searches for a color specification in this order:

1. the CFILL= option

2 the COLOR= option in a PATTERN statement

3 the COLOR= option in a GOPTIONS statement

4 the color of the current style, or, the first color in the device's color list if the NOGSTYLE option is specified

Style reference Color attribute of the GraphData1 to GraphDataN element, depending on the number of slices in the pie

See For more information, see "Controlling Slice Patterns and Colors" on page 1074.

OTHERLABEL='*text-string*'
specifies a text string up to 16 characters for the label for the OTHER slice. The default label is OTHER.

PERCENT=ARROW | INSIDE | NONE | OUTSIDE
prints the percentage represented by each slice using the specified labeling method. For a description of the option values, see "Selecting and Positioning Slice Labels" on page 1073. By default, PERCENT=NONE (percentage is not displayed).

Whether the slice percent is displayed with or without decimal places, depends on the range of values across the chart. The only way to control the appearance of these values is to calculate the percentage with a DATA step. You can also calculate the percentage with a statistical procedure and use the resulting data set as input to the GCHART procedure. Assign the variable that contains the calculated percentages to the SUMVAR= option.

Examples "Example 9: Ordering and Labeling Slices in a Pie Chart" on page 1115

"Example 10: Grouping and Arranging Pie Charts" on page 1116

PLABEL=(*text-options*)
modifies the font, height, and color of the pie slice labels. Text-options can be one or more of the following:

- COLOR=*color*
- FONT=*font*
- HEIGHT=*text-height* <*units*>

Specify as many options as you want, but enclose them all in one set of parentheses, even if you specify only one option.

Style reference Font Attributes of the GraphValueText element

Restriction The Java and ActiveX devices do not support all of the text options

Interaction If there is no PPERCENT= option specification, the PLABEL= option controls the attributes for both the pie slice label and its associated percentage.

See "Text Description Suboptions for Donut" on page 1071 for a complete description

"PPERCENT=(*text-options*)" on page 1068

PPERCENT=(*text-options*)
modifies the font, height, and color of the displayed percentages associated with the pie slice labels. This option enables you to differentiate the attributes of the percentage from the attributes of its associated text label. For example, you can specify a particular font and an orange color for the percentage with the PPERCENT= option, and a different font and a blue color for the label with the PLABEL= option:

```
ppercent=(font='Albany AMT' h=1.3 color=orange);
plabel=(font='Arial' h=1 color=blue);
```

Text-options can be one or more of the following:

- COLOR=*color*
- FONT=*font*
- HEIGHT=*text-height* <*units*>

Specify as many options as you want, but enclose them all in one set of parentheses, even if you specify only one option.

Default When PPERCENT is not specified, the percent displayed on the graph will share the same attributes as those of the pie slice label.

Style reference Font Attributes of the GraphValueText element

Restriction The Java and ActiveX devices do not support all of the text options

See "Text Description Suboptions for Donut" on page 1071 for a complete description

RADIUS=*value*
specifies the radius of the pie and donut in their respective charts. *Value* is the pie radius in character cells.

SLICE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice name (midpoint value) for each slice.

Default SLICE=OUTSIDE (the name is outside of the slice)

See "Selecting and Positioning Slice Labels" on page 1073 for a description of the option values

Examples "Example 9: Ordering and Labeling Slices in a Pie Chart" on page 1115

"Example 10: Grouping and Arranging Pie Charts" on page 1116

STATFMT=*format-specification*
overrides the GCHART default format of the displayed statistical value.

Note: This option is valid starting with SAS 9.4M5.

The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE=

option. Use the STATFMT= option to change the default format that might contain decimal points, percentages, or commas. Here is an example:

```
proc gchart data=sashelp.electric;
   pie year / discrete sumvar=Revenue
      type=sum statfmt=dollar8.2 percent=inside;
run;
quit;
```

In this example, option STATFMT= in the PIE statement specifies format DOLLAR8.2 for the SUM values.

Alias SFMT=, SFORMAT=, STATFORMAT=

Restriction Do not use the STATFMT=PERCENTw.d in conjunction with the TYPE=PERCENT option when you need to produce a pie with percentages. Choose one or the other. Using both yields undesired results. When you use both, the TYPE= option adds a percentage sign '%', and the STATFMT= option also adds a percentage sign and multiplies the calculated statistic by 100. A calculated percentage of 52.16% becomes 5216%%.

SUBGROUP=*subgroup-variable*

divides the chart into concentric rings according to the values of *subgroup-variable*. The width of the rings, which is the same for each subgroup, is determined by the radius of the pie and the size of the donut hole, if any.

By default, the subgroup rings are ordered from the outside in, alphabetically (if character) or numerically (if numeric). If the JSTYLE option is also used, the order of the slices within the subgroups is determined by the outermost subgroup. There might be an inner subgroup that contains a value that is not in the outer subgroup. In this case the new slice for that value is placed either last or just before the "other" slice, if one is present. That slice order is continued for any remaining subgroups.

Each ring is labeled with its subgroup value; labels are placed to the right of the chart. If the GROUP= option is also used and if all groups contain the same subgroups, then only the first (upper left) chart on each page is labeled. If any group differs in the number of subgroups that it contains, then all charts are labeled.

By default the subgroups are outlined in the foreground color. To specify an outline color, use the COUTLINE= option.

The SUBGROUP= option automatically generates a legend for the midpoint values (not the subgroup values) and suppresses display of the chart statistic. By default the legend appears at the bottom of the chart. To modify the legend, assign a LEGEND definition. To suppress the legend, specify NOLEGEND. To display the chart statistic, use the VALUE= option.

Interaction If EXPLODE is also used, it is ignored.

See "Controlling Bar Chart Patterns, Colors, and Images" on page 1050

"LEGEND Statement" on page 432

Examples "Example 8: Subgrouping a Donut or Pie Chart" on page 1112

"Example 9: Ordering and Labeling Slices in a Pie Chart" on page 1115

SUMVAR=*numeric-variable*

specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of *numeric-variable* for each midpoint. The resulting statistics are represented by the size of the slice and displayed beside of each slice.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

Example "Example 7: Specifying the Sum Statistic for a Pie Chart" on page 1109

TYPE=*statistic*

specifies the chart statistic.

- If the SUMVAR= option is not used, *statistic* can be one of the following:

FREQ
: frequency (the default)

PERCENT PCT
: percentage

- If SUMVAR= is used, *statistic* can be one of the following:

SUM
: sum (the default)

MEAN
: mean

Because you cannot use TYPE=FREQ or TYPE=PERCENT with the SUMVAR= option, you must use FREQ= to calculate percentages or frequencies based on a sum.

Restriction Do not use the TYPE=PERCENT option in conjunction with the STATFMT=PERCENTw.d when you need to produce a pie with percentages. Choose one or the other. Using both yields undesired results. When you use both, the TYPE= option adds a percentage sign '%', and the STATFMT= option also adds a percentage sign and multiplies the calculated statistic by 100. A calculated percentage of 52.16% becomes 5216%%.

See "About Chart Statistics" on page 996

"Calculating Weighted Statistics" on page 997

URL=*character-variable*

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction This option affects graphics output that is created through the ODS HTML destination only.

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

VALUE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice value (chart statistic) for each slice.

Default VALUE=OUTSIDE (the value is outside the slice)

See "Selecting and Positioning Slice Labels" on page 1073 for a description of the option values

Example "Example 9: Ordering and Labeling Slices in a Pie Chart" on page 1115

WOUTLINE=*slice-outline-width*
specifies the width of the outline in pixels. WOUTLINE= affects both the slice outlines and the subgroup outlines.

Style reference LineThickness attribute of the GraphOutlines element

Restriction Not supported by Java and ActiveX

Text Description Suboptions for Donut

The LABEL= option in the DONUT statement and both the PLABEL= and PPERCENT= options in the PIE statement use text description suboptions to change the attributes of the text string or strings that follow.

ANGLE=*degrees*
specifies the angle at which the baseline of the text string or strings is rotated with respect to the horizontal. A positive value for degrees moves the baseline counterclockwise; a negative value moves it clockwise. By default, ANGLE=0 (horizontal).

Valid in DONUT

Alias A=

Restriction Not supported by Java

COLOR=*color*
specifies the color for the text string or strings. The COLOR= suboption stays in effect until another COLOR= specification is encountered. If you omit the COLOR= option, the LABEL= option uses the color defined by the current style.

Valid in DONUT, PIE, PIE3D

Alias C=

Restriction Not supported by Java

Interaction COLOR= suboption ignores the CTEXT= graphics option

See Chapter 23, "Using Colors in SAS/GRAPH Programs," on page 363 for details about specifying *color*

FONT=*font*
specifies the font for the text string or strings. If you omit the FONT= option, the LABEL= option uses the font that is specified by the FTEXT= graphics option. If no font is specified, it uses the default hardware font, NONE.

Valid in DONUT, PIE, PIE3D

Alias F=

Restriction The Java and ActiveX devices do not support all fonts

See Chapter 22, “Specifying Fonts in SAS/GRAPH Programs,” on page 349 for details about specifying *font*

“FTEXT Graphics Option” on page 626

HEIGHT=*text-height* <*units*>
specifies the height of the text string or strings. *Text-height* is the number of units. If you omit the HEIGHT= option, the LABEL= option uses the height that is specified by the HTEXT= graphics option. If no text height is specified and if the default text height is too large for the donut hole, the size of the label is reduced to fit. *Units* can be CELLS | CM | IN | PCT | PT. If you omit *units*, then the HEIGHT= option uses the unit that is specified by the GUNIT= graphics option, or the default unit, CELLS.

Valid in DONUT, PIE, PIE3D

Alias H=

Restriction Not supported by Java and ActiveX

JUSTIFY=LEFT | CENTER | RIGHT
specifies the alignment of the text string or strings. By default, JUSTIFY=CENTER.

Alias J=

Restriction Not supported by Java and ActiveX

ROTATE=
specifies the angle at which each character is rotated with respect to the baseline of the text string. The angle is measured from the current text baseline angle specified by the ANGLE= suboption. A positive value for *degrees* rotates the character counterclockwise; a negative value rotates it clockwise. By default, ROTATE=0 (parallel to the baseline).

Valid in DONUT

Restriction Not supported by Java

Details

Description

The PIE, PIE3D, and DONUT statements specify the variable or variables that define the categories of data to chart. These statements automatically do the following:

- determine the midpoints.

- calculate the chart statistic for each midpoint (the default is FREQ).
- scale each slice to represent its chart statistic. No slice is drawn if the chart statistic for the midpoint is 0.
- order the slices by midpoint value in ascending order starting at the three o'clock position and proceeding counterclockwise around the pie.
- print the slice name (midpoint value) and slice value (chart statistic) beside each slice.
- assign patterns and colors to the slices. The default pie pattern is PSOLID.

You can use statement options to select or order the midpoints (slices), to change the type of chart statistic, and to modify the appearance of the chart. This includes changing the content and position of the slice labels, and the patterns used by the slices. You can also specify additional variables by which to group, subgroup, or sum the data. Statement options can also produce special effects, such as exploded or invisible slices.

Donut and pie charts allow grouping and subgrouping. Grouping creates two or more separate pie or donut charts that are displayed in rows or columns on one graph. Subgrouping creates a separate ring of slices within the circle for each value of the subgroup variable. The concentric rings of the subgrouped pie or donut chart make it easy to compare slice values between subgroups.

In addition, you can use global statements to modify patterns and legends, as well as add titles, footnotes, and notes to the chart. You can also use an Annotate data set to enhance the chart.

Selecting and Positioning Slice Labels

By default, each slice is labeled with its midpoint value (slice name) and its chart statistic value (slice value), which are printed outside of the slice. You can control where and how these labels are displayed with the SLICE= and VALUE= options, respectively. In addition, each slice can display the percentage its midpoint contributes to the total chart statistic (slice percent). Use the PERCENT= option to request slice percent.

The SLICE=, VALUE=, and PERCENT= options use the same values:

ARROW
: places the text outside the slice and connects the text to the slice with a line. This labeling method reduces the radius of the pie. The arrow uses the color that is specified by the CTEXT= option in the PIE, PIE3D, or DONUT statement. If the CTEXT= option is omitted, the arrow uses the color defined by the current style.

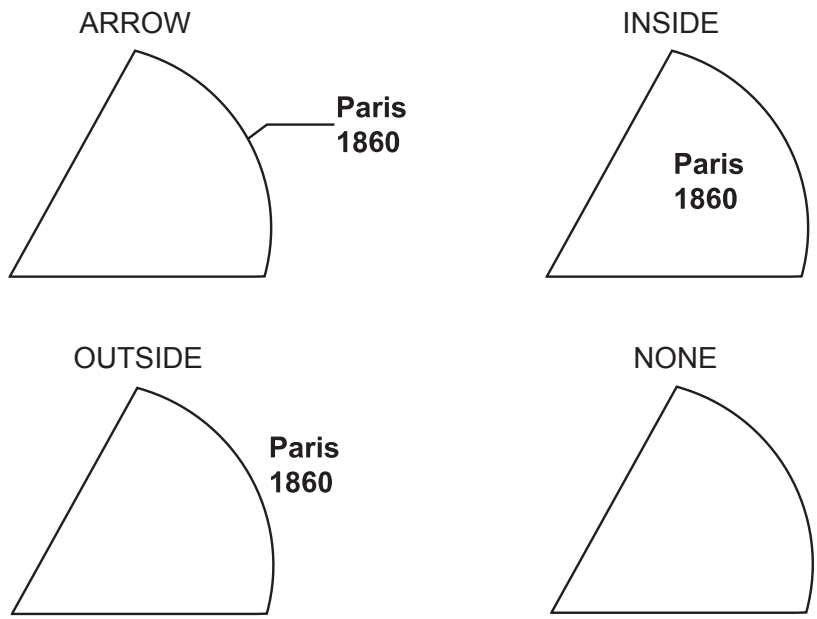
INSIDE
: places the text inside the slice. The label overlays the slice fill patterns. This labeling method increases the radius of the pie.

NONE
: suppresses the text.

OUTSIDE
: places the text outside of the slice.

Figure 36.122 on page 1074 illustrates these values.

Figure 36.14 Slice Labeling Methods



The SLICE= and VALUE= options are dependent on each other. If you specify only VALUE= or only SLICE=, the other option automatically uses the same labeling method. The PERCENT= option is independent of these two.

Be careful about the combinations that you specify. For example, if you specify PERCENT=ARROW and VALUE=OUTSIDE, the line that connects the percentage information to each slice might overlay the statistic value.

If your pie has many slices, the labels might overlap, particularly if there are several small slices together. You can correct the overlapping labels by using any of the following options:

- the HTEXT= graphics option to decrease the size of the labels.
- the GRSEG Graphics Editor to adjust the labels by moving or resizing the text.
- the ANGLE= option to change the orientation of the pie.
- the MIDPOINTS= option to rearrange slices so that small slices are not together.
- the OTHER= option to group more midpoints into the OTHER category.
- the HPOS= and VPOS= graphics options to increase the number of cells in your display. See “The Graphics Output and Device Display Areas” on page 80 for details.

Controlling Slice Patterns and Colors

How to Control Slice Patterns and Colors

Pie and donut charts are always patterned by midpoint. Even when you specify subgrouping, the patterning method does not change from midpoint to subgroup.

Default Patterns and Outlines

Each slice in a pie or donut chart is filled with a pattern. Because the system option GSTYLE is in effect by default, the procedure uses the current style's default patterns and outlines when producing output. By default, the procedure does the following:

- fills the slices with pie patterns, beginning with the default fill, PSOLID, and rotates it through the list of colors available in the current style. When these colors are exhausted, the procedure rotates through a slightly modified version of the original list of colors. It continues in this fashion until all of the chart variables have been assigned a unique pattern.

  Note: PIE3D always uses solid patterns.

  If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, the procedure uses all the colors in the list to generate the patterns.
- outlines slices and subgroup segments using the color defined by the style. To change the outline color, use the COUTLINE= option.

See "About Patterns" on page 998 for more information about how the GCHART procedure assigns default patterns and outlines.

Controlling Patterns

You can control slice patterns and their outlines in several ways.

- To select a different fill for the slices, such as empty or hatched, you can do the following:
  - request a single hatched fill pattern for all slices by specifying the FILL=X option on the PIE or DONUT statement. The pattern specified by FILL=X uses the colors in the color list as many times as needed to generate all of the patterns that are required by the chart. If you specify a single color with either CFILL= or the graphics option, CPATTERN=, all slices use the same color as well as the same pattern.
  - specify a pattern with the VALUE= option in the PATTERN statement. Only pie patterns are valid; all other pattern specifications are ignored. For a complete description of all pie patterns, see "VALUE=*pie/star-pattern* " on page 461 in the PATTERN statement.

    If no color options are specified, the procedure rotates each specified fill once through the list of colors available in the current style. Otherwise, the PATTERN statement generates one pattern definition for the specified pattern and color. When all of the specified patterns are exhausted, the procedure starts rotating through the default pie patterns, beginning with PSOLID.
- To select colors for the slices, you can do the following:
  - specify a single pattern color with the CFILL= option, or with the CPATTERN= graphics option, or with a COLORS= list of one color. For the PIE and DONUT statements, CFILL= starts with the default solid color and uses the foreground color for outlines. In contrast, using the CPATTERN= graphics option or a COLORS= list of one color skips the solid pattern. Beginning with

P2N0, these options use each pie hatch pattern with the specified color, and use the fill color for the outline color.

- specify only the COLOR= option in one or more PATTERN statements. In this case, the procedure creates a solid pattern for each specified color. When it runs out of PATTERN statements, it returns to the default patterns, beginning with PSOLID, and rotates them each through the color list. Whenever you specify a PATTERN statement, the default outline color is SAME.

- To define specific patterns and colors for the slices, use PATTERN statements and specify both the VALUE= and COLOR= options. You might provide fewer PATTERN definitions than the chart requires. In this case the GCHART procedure uses the default pattern rotation for the slices that are drawn after all of the defined patterns are exhausted.

See "About Patterns" on page 998 for more information about how the GCHART procedure uses patterns and outlines. See the "PATTERN Statement" on page 455 for a description of default pie patterns.

Modifying the Statistic Heading and the Group Heading

By default, the procedure prints a heading at the top of each pie (or donut) chart that indicates the type of statistic charted and the name of the chart variable. For an example see Figure 36.112 on page 987. You can suppress this heading with the NOHEADING option.

When you use the GROUP= option, a heading is printed above each pie indicating the name of the group variable and its value for the particular pie (for example, *SITE=Paris*). You can suppress these headings with the NOGROUPHEADING option. You can also suppress the variable name *SITE=* so that only the value *Paris* remains. To do this, use a LABEL statement and assign a null value to the variable name, for example:

```
label site='00'x;
```

The AXIS statement cannot be used by the PIE, PIE3D, and DONUT statements. Instead, you should use the graphics options "FTEXT Graphics Option" on page 626 and "HTEXT Graphics Option" on page 650 to control the font and height of text on the chart. Increasing the value of the HTEXT= graphics option decreases the size of the pie if any slice labels are positioned outside.

Substituting BY Line Values in a Text String

The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

- The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.
- There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the "BY Statement" on page 424.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

STAR Statement

Creates star charts in which the length of the spines represents the value of the chart statistic for each category of data or midpoint.

Restriction:	Not supported by Java and ActiveX
Requirement:	At least one chart variable is required.
Supported Statements:	FOOTNOTE, PATTERN, TITLE
Supports:	Drill-down functionality (slices only)

Syntax

STAR *chart-variable(s)* < / *options* >;

Summary of Optional Arguments

Appearance options

ANGLE=*degrees*
: starts the first slice at the specified angle.

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate all graphs that are produced by the GCHART procedure.

ASCENDING
: arranges the bars in ascending order of the value of the chart statistic.

CFILL=*fill-color*
: specifies one color for all slices in the chart, regardless of whether the fill is solid or hatch.

COUTLINE=*star-outline-color* | SAME
: specifies the color for the circle that surrounds the star chart and for the slice outlines or spines.

DESCENDING
: arranges the bars in descending order of the value of the chart statistic.

FILL=SOLID | X
: specifies the fill pattern for all slices in the star chart.

LEGEND=LEGEND<1 …99>
: assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.

NOCONNECT
: draws only star spines without connecting lines.

NOLEGEND
: suppresses the legend automatically generated by the SUBGROUP= option.

STARMAX=*max-value*
: scales the chart so that the outside (or edge) of the circle represents the value that is specified by *max-value*.

STARMIN=*min-value*
: scales the chart so that the center of the circle represents the value that is specified by *min-value*.

WOUTLINE=*slice-outline-width*
: specifies the width of the outline in pixels.

Catalog entry description options

DESCRIPTION="*description*"
: specifies a description of the output.

NAME="*name*"
: specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Grouping options

ACROSS=*number-of-columns*
: draws *number-of-columns* stars across the procedure output area.

DOWN=*number-of-rows*
: draws *number-of-rows* stars vertically in the procedure output area.

GROUP=*variable*
: organizes the data according to values of *group-variable* and produces a separate star chart for each unique value of *group-variable*.

Labeling options

CTEXT=*text-color*
: specifies a color for all text on the axes and legend.

MATCHCOLOR
: uses the slice pattern color for all slice labels.

NOGROUPHEADING
: suppresses the headings normally printed above each star when you use the GROUP= option.

NOHEADING

suppresses the heading normally printed at the top of each page or display of star chart output.

PERCENT=ARROW | INSIDE | NONE | OUTSIDE
prints the percentage represented by each slice using the specified labeling method.

SLICE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice name (midpoint value) for each slice.

VALUE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice value (chart statistic) for each slice.

Midpoint options

DISCRETE
treats a numeric chart variable as a discrete variable rather than as a continuous variable.

LEVELS=*number-of-midpoints* | ALL
specifies number of midpoints to be graphed for a chart variable.

MIDPOINTS=OLD
generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

MIDPOINTS=*value-list*
specifies the midpoint values for the slices.

MISSING
accepts a missing value as a valid midpoint for the chart variable.

ODS options

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Statistic options

FREQ=*numeric-variable*
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

STATFMT=*format-specification*
overrides the GCHART default format of the displayed statistical value.

SUMVAR=*numeric-variable*
specifies a numeric variable for sum or mean calculations.

TYPE=*statistic*
specifies the chart statistic.

Required Argument

chart-variable(s)

specifies one or more variables that define the categories of data to chart. Each chart variable draws a separate chart. All variables must be in the input data set. Separate multiple chart variables with blanks.

See "Understanding Chart Variables" on page 992

Optional Arguments

Options in a STAR statement affect all of the graphs that are produced by that statement. You can specify as many options as you want and list them in any order. For details about specifying colors, see Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349.

ACROSS=*number-of-columns*

draws *number-of-columns* stars across the procedure output area. The ACROSS= option is ignored unless you also use the GROUP= option. If *number-of-columns* calls for more stars than fit horizontally in the graphics area of the output device, no stars are drawn and an error message is written to the SAS log.

If you also use the DOWN= option, the star charts are drawn in left-to-right and top-to-bottom order.

ANGLE=*degrees*

starts the first slice at the specified angle. A value of 0 for *degrees* corresponds to the three o'clock position. *Degrees* can be either positive or negative. Positive values move the starting position counterclockwise; negative values move the starting position clockwise.

If the star chart uses spines instead of slices, *degrees* specifies the angle of the position halfway between the first spine and the last spine.

By default, ANGLE=0, which places the first spine or the center of the first slice of the star at the 0-degree position. Successive star spines or slices are drawn counterclockwise from the starting position.

ANNOTATE=*Annotate-data-set*

specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

Note Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block, pie, donut, or star charts.

See Chapter 28, "Using Annotate Data Sets," on page 721

ASCENDING

arranges the bars in ascending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. ASCENDING reorders the bars from shortest to longest. In horizontal bar charts the ordering is top to bottom; in vertical bar charts the ordering is left to right.

Interactions If you also use the GROUP= option, the reordering is performed separately for each group, so the order of the midpoints might be different for each group.

The ASCENDING option overrides any midpoint order specified with the MIDPOINTS= option or specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

CFILL=*fill-color*

specifies one color for all slices in the chart, regardless of whether the fill is solid or hatch. For the PIE and DONUT statements, it is possible that no pattern is specified in the pattern statement or with the FILL= option. In this case the procedure starts with the default solid fill and then, beginning with P2N0, uses each default pie hatch pattern with the specified color. For the outline color, the procedure uses the default color, which is retrieved from the current style. However, if the NOGSTYLE option is specified, the procedure uses the first color in the device's color list. Use the COUTLINE= option to specify a different outline color. The CFILL= option overrides any other pattern color specification and controls the color of all slices.

Style reference Color attribute of the GraphData1 element

COUTLINE=*star-outline-color* | SAME

specifies the color for the circle that surrounds the star chart and for the slice outlines or spines. SAME specifies that the outline color of a slice is the same as the interior pattern color. Specifying COUTLINE=SAME affects only slice outlines and has no effect on the color of the circle.

The default circle and outline color are both specified in the current device. However, if the NOGSTYLE option is specified, then the default circle color is the first color in the device's color list (the foreground color). In addition, the default slice outline color is determined as follows:

- If you do not specify a PATTERN statement, the default outline color is the color defined in the current style.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

Alias CO=

Style reference Color attribute of the GraphOutlines element

Note Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

See “Selecting Patterns for the Star Charts” on page 1091

“About Patterns” on page 998

Example “Example 12: Charting a Discrete Numeric Variable in a Star Chart” on page 1120

CTEXT=*text-color*

specifies a color for all text on the axes and legend. This includes axis labels, tick mark values, legend labels, and legend value descriptions. The GCHART procedure looks for the text color in the following order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the STAR statement.
2. the color specified by the CTEXT= option in the STAR statement.
3. the color specified by the CTEXT= option in a GOPTIONS statement.
4. the color specified in the current style. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. The COLOR= color is still used for all other axis colors, such as tick marks.

Alias	CT=
Style reference	Color attributes for the GraphLabelText and the GraphValueText elements
Note	If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

DESCENDING

arranges the bars in descending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. DESCENDING reorders the bars from longest to shortest. In horizontal bar charts the ordering is top to bottom; in vertical bar charts the ordering is left to right. If you also use the GROUP= option, the reordering is performed separately for each group, so the order of the midpoints might be different for each group.

The DESCENDING option overrides any midpoint order that is specified either with the MIDPOINTS= option or the ORDER= option in an AXIS statement assigned to the midpoint axis.

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See “Chart Descriptions for Web Presentations” on page 251 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the SAS Explorer.

- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default STAR CHART OF *chart-variable*

DISCRETE

treats a numeric chart variable as a discrete variable rather than as a continuous variable. The GCHART procedure creates a separate midpoint and, hence, a separate star slice for each unique value of the chart variable. If the variable has a format that is associated with it, each format value is treated as a separate value.

The LEVELS=*number-of-midpoints* option is ignored when you use the DISCRETE option. The MIDPOINTS= option overrides the DISCRETE option.

Example "Example 12: Charting a Discrete Numeric Variable in a Star Chart" on page 1120

DOWN=*number-of-rows*

draws *number-of-rows* stars vertically in the procedure output area. The DOWN= option is ignored unless you also use the GROUP= option. If *number-of-rows* calls for more stars than fit vertically in the graphics area of the output device, no stars are drawn and an error message is written to the SAS log.

If you also use the ACROSS= option, the stars are drawn in left-to-right and top-to-bottom order.

FILL=SOLID | X

specifies the fill pattern for all slices in the star chart:

SOLID

rotates a solid fill through the list of colors available in the default style as many times as necessary. SOLID is the default.

Alias S

X

rotates a single hatch pattern through the list of colors defined in the current style. If the NOGSTYLE option is specified, it rotates the hatch pattern through the device color list as many times as necessary. If you do not specify the colors= goption, the fill skips the first color in the color list.

FILL= overrides any pattern that is specified in PATTERN statements.

By default, the outline color is the color defined by the current style. If the NOGSTYLE option is specified, the outline color is the first color in the device's color list. If PATTERN statements are used to specify colors, the slice outline color matches the slice fill color.

If any PATTERN statements have been defined, the colors in the PATTERN definitions are used, in order, before the default style color rotation.

Style reference Color attribute of the GraphData1 element

Example "Example 12: Charting a Discrete Numeric Variable in a Star Chart" on page 1120

FREQ=*numeric-variable*

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times that are specified by the value of *numeric-variable* for that observation. If the value of *numeric-variable* is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of *numeric-variable* are truncated to integers.

The FREQ= option is valid with all chart statistics.

Because you cannot use TYPE=PERCENT or TYPE=FREQ with the SUMVAR= option, you must use FREQ= to calculate percentages and frequencies based on a sum.

The statistics are not affected by applying a format to *numeric-variable*.

See "Calculating Weighted Statistics" on page 997

GROUP=*variable*

organizes the data according to values of *group-variable* and produces a separate star chart for each unique value of *group-variable*. *Group-variable* can be either character or numeric and is always treated as a discrete variable. Missing values for *group-variable* are treated as a valid group.

By default, the charts are produced in ascending order of group variable value and each is drawn on a separate page or display. Therefore, the effect of GROUP= is essentially the same as using a BY statement. The exception is that GROUP= causes the midpoints with the same value to use the same color and fill pattern. To place more than one star chart on a page or display, use the ACROSS= or DOWN= options, or both.

HTML=*variable*

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction Not supported by Java or ActiveX

See "Overview of Enhancing Web Presentations" on page 250

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Only star charts with slices support drill-down functionality

See "Overview of Enhancing Web Presentations" on page 250

LEGEND=LEGEND<1 …99>
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option. The LEGEND= option itself does not generate a legend.

The LEGEND= option is ignored if any of the following are true:

- The SUBGROUP= option is not used.
- The specified LEGEND definition is not in effect.
- The NOLEGEND option is used.
- The PATTERNID= option is set to any value other than SUBGROUP. That is, the value of PATTERNID= is BY or GROUP or MIDPOINT.

To create a legend based on the chart midpoints instead of the subgroups, use the chart variable as the subgroup variable:

```
block city / subgroup=city;
```

Restriction The Java and ActiveX devices do not support all LEGEND statement options.

See the "LEGEND Statement" on page 432 for more information

"SUBGROUP=*subgroup-variable*" on page 1011

Example "Example 2: Grouping and Subgrouping a Block Chart" on page 1095

LEVELS=*number-of-midpoints* | ALL
specifies number of midpoints to be graphed for a chart variable. After you specify the number of midpoints that you want, the range for each numeric midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985). If you specify LEVELS=ALL, then all unique numeric or character midpoint values are graphed. If your data contains a large number of unique midpoint values (more than 200), then you can use the XPIXELS and YPIXELS GOPTIONS. This enables the device driver to render a larger (and more readable) graph. The LEVELS=*number-of-midpoints* option is ignored if any of the following are true:

- The chart variable is character type.
- The DISCRETE option is used.
- The MIDPOINTS= option is used.

MATCHCOLOR
uses the slice pattern color for all slice labels. MATCHCOLOR overrides the color that is specified in the CTEXT= option. If the chart uses spines instead of slices, the spine color is used for the slice label and value text.

MIDPOINTS=*value-list*
specifies the midpoint values for the slices. The way you specify *value-list* depends on the type of variable:

- For numeric chart variables, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - *n* < *...n*>

- *n* TO *n* <BY *increment*>
- *n* < ...*n*> TO *n* <BY *increment*> <*n* < ...*n*>>

If a numeric variable has an associated format, the specified values must be the *unformatted* values.

If you omit the DISCRETE option, then numeric values are treated as continuous, which means that the following is true by default:

- The lowest midpoint consolidates all data points from negative infinity to the median of the first two midpoints.
- The highest midpoint consolidates all data points from the median of the last two midpoints up to infinity.
- All other values in *value-list* specify the median of a range of values, and the GCHART procedure calculates the midpoint values.

If you include the DISCRETE option, each value in *value-list* specifies a unique numeric value.

- For character chart variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:

'value-1' < ...*'value-n'*>

If a character variable has an associated format, the specified values must be the *formatted* values.

For a complete description of *value-list*, see the "*value-list*" on page 408 in the AXIS statement.

See "Understanding Midpoints" on page 993

MIDPOINTS=OLD
generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976). The MIDPOINTS=OLD option is ignored unless the chart variable is numeric

MISSING
accepts a missing value as a valid midpoint for the chart variable. By default, observations with a missing value are ignored. Missing values are always valid for the group variable.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.

- If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

- **For the graphics output file name:**

- The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
- All characters are represented in lowercase.
- If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
- If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

  Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GCHART

NOCONNECT
draws only star spines without connecting lines. By default, the spines are connected to form slices.

Example "Example 12: Charting a Discrete Numeric Variable in a Star Chart" on page 1120

NOGROUPHEADING
suppresses the headings normally printed above each star when you use the GROUP= option.

NOHEADING
suppresses the heading normally printed at the top of each page or display of star chart output.

Example "Example 12: Charting a Discrete Numeric Variable in a Star Chart" on page 1120

NOLEGEND
suppresses the legend automatically generated by the SUBGROUP= option. The NOLEGEND option is ignored if the SUBGROUP= option is not used.

PERCENT=ARROW | INSIDE | NONE | OUTSIDE
prints the percentage represented by each slice using the specified labeling method. For a description of the option values see "Selecting and Positioning

Spine and Slice Labels" on page 1090. By default, PERCENT=NONE (percentage is not displayed).

SLICE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice name (midpoint value) for each slice. For a description of the option values, see "Selecting and Positioning Spine and Slice Labels" on page 1090. By default, SLICE=OUTSIDE (the name is outside the slice).

STARMAX=*max-value*
scales the chart so that the outside (or edge) of the circle represents the value that is specified by *max-value*. By default, the value for STARMAX= is the maximum chart statistic value.

STARMIN=*min-value*
scales the chart so that the center of the circle represents the value that is specified by *min-value*. By default, STARMIN=0. If the chart statistic has negative values, by default the value for the STARMIN= option is the minimum chart statistic value.

STATFMT=*format-specification*
overrides the GCHART default format of the displayed statistical value.

Note: This option is valid starting with SAS 9.4M5.

The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use the STATFMT= option to change the default format that might contain decimal points, percentages, or commas. Here is an example:

```
proc gchart data=sashelp.electric;
   star year / discrete sumvar=Revenue
      type=sum statfmt=dollar8.1 percent=inside;
run;
quit;
```

In this example, the STATFMT= option in the STAR statement specifies format DOLLAR8.1 for the SUM values.

Alias SFMT=, SFORMAT=, STATFORMAT=

SUMVAR=*numeric-variable*
specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of the value of *numeric-variable* for each midpoint. The resulting statistics are represented by the size of the slice and displayed beside each slice.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

Example "Example 11: Specifying the Sum Statistic in a Star Chart" on page 1118

TYPE=*statistic*
specifies the chart statistic.

- If the SUMVAR= option is not used, *statistic* can be one of the following:

FREQ
frequency (the default)

PERCENT PCT
percentage

If the SUMVAR= option is used, *statistic* can be one of the following:

SUM
sum (the default)

MEAN
mean

Because you cannot use TYPE=FREQ or TYPE=PERCENT with the SUMVAR= option, you must use FREQ= to calculate percentages or frequencies based on a sum.

See "About Chart Statistics" on page 996

"Calculating Weighted Statistics" on page 997

URL=*character-variable*
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only.

Not supported by Java or ActiveX

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

VALUE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice value (chart statistic) for each slice.

Default VALUE=OUTSIDE (the value is outside of the slice)

See "Selecting and Positioning Spine and Slice Labels" on page 1090 for a description of the option values

WOUTLINE=*slice-outline-width*
specifies the width of the outline in pixels. The WOUTLINE= option affects the slice outlines.

Style reference LineThickness attribute of the GraphOutlines element

Details

Description

The STAR statement specifies the variable or variables that define the categories of data to chart. This statement automatically does the following:

- determines the midpoints.
- calculates the chart statistic for each midpoint (the default is FREQ).
- scales each spine or slice to represent the chart statistic. Slices or spines are drawn for all midpoints where the value of the chart statistic is greater than the value that is specified in the STARMIN= option.
- arranges the spines or slice counterclockwise around the star in ascending order of midpoint value, starting at the three o'clock position.
- prints the midpoint value and chart statistic beside each spine or slice.
- assigns patterns to the slices.

If all the data to be charted with the STAR statement are positive, the center of the star represents 0 and the outside circle represents the maximum value. If negative values are calculated for the chart statistic, the center represents the minimum value in the data. You can specify other values for the center and outside of the circle with the STARMIN= and STARMAX= options.

You can also use statement options to

- select or order the midpoints
- change the type of chart statistic
- modify the appearance of the chart. This includes the content and position of the spine or slice labels, and patterns that fill the slice.

You can specify additional variables by which to group or sum the data.

Star charts allow grouping, which creates two or more separate charts that are displayed in rows or columns on one graph.

In addition, you can use global statements to modify patterns as well as add titles, footnotes, and notes to the chart. You can also use an Annotate data set to enhance the chart.

Selecting and Positioning Spine and Slice Labels

By default, each spine or slice is labeled with its midpoint value and its chart statistic value, which are printed outside of the circle. You can control where and how these labels are displayed with the SLICE= and VALUE= options, respectively. In addition, each spine can display the percentage that its midpoint contributes to the total chart statistic (spine percent). Use the PERCENT= option to request spine percent.

The SLICE=, VALUE=, and PERCENT= options use the same values:

ARROW
: places the text outside of the star circle and connects the text to the circle with a line. The line points to the spine or the center of the slice. The arrow uses the color that is specified by the CTEXT= option in the STAR statement. If you omit the CTEXT= option, the arrow uses the color defined by the current style.

INSIDE
: places the text inside the star circle.

NONE
: suppresses the text.

OUTSIDE
: places the text outside the star circle.

Figure 36.122 on page 1074 illustrates these values.

The SLICE= and VALUE= options are dependent on each other. If you specify only VALUE= or only SLICE=, the other option automatically uses the same labeling method. The PERCENT= option is independent of these two.

Be careful about the combinations that you specify. For example, if you specify PERCENT=ARROW and VALUE=OUTSIDE, the line that connects the percentage information to each spine might overlay the statistic value.

Selecting Patterns for the Star Charts

Determining Pattern Selection

Star charts are always patterned by midpoint.

Default Patterns and Outlines

Each slice in a star chart is filled with a pattern. Because the system option GSTYLE is in effect by default, the procedure uses the current style's default patterns and outlines when producing output. By default, the procedure does the following:

- fills the slices with star patterns, beginning with the default fill, PSOLID, and rotates it through the list of colors available in the default style. When these colors are exhausted, the procedure rotates through a slightly modified version of the original list of colors. It continues in this fashion until all of the chart variables have been assigned a unique pattern.

  If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, the procedure uses all of the colors in the list to generate the patterns.
- outlines slices using the color defined by the style. To change the outline color, use the COUTLINE= option.

See “About Patterns” on page 998 for more information about how the GCHART procedure assigns default patterns and outlines.

Controlling Patterns

You can control slice patterns and their outlines in several ways.

- To select a different fill for the slices, such as empty or hatched, you can do the following:
  - request a single hatched fill pattern for all slices by specifying the FILL=X option in the STAR statement. The pattern that is specified by FILL=X rotates through the list of colors available in the current style. This happens as many times as needed to generate all the patterns required by the chart. If you specify a single color with either CFILL= or the graphics option, CPATTERN=, all slices use the same color as well as the same pattern.
  - specify a pattern with the VALUE= option in the PATTERN statement. Only star patterns are valid; all other pattern specifications are ignored. For a complete description of all star patterns, see the option description for “VALUE=*pie/star-pattern* ” on page 461 in the PATTERN statement

If no color options are specified, the procedure rotates each specified fill once through the list of colors available in the current style. Otherwise, the PATTERN statement generates one pattern definition for the specified pattern and color. When all of the specified patterns are exhausted, the procedure starts rotating through the default star patterns, beginning with PSOLID.

- To select colors for the slices, you can do the following:
  - specify a single pattern color with the CFILL= option, or with the CPATTERN= graphics option, or with a COLORS= list of one color. For the PIE and DONUT statements, CFILL= starts with the default solid color and uses the foreground color for outlines. In contrast, using the CPATTERN= graphics option or a COLORS= list of one color skips the solid pattern. Beginning with P2N0, these options use each pie hatch pattern with the specified color, and use the fill color for the outline color.
  - specify only the COLOR= option in one or more PATTERN statements. In this case, the procedure creates a solid pattern for each specified color. When it runs out of PATTERN statements, it returns to the default patterns, beginning with PSOLID, and rotates them each through the list of colors available in the current style. Whenever you specify a PATTERN statement, the default outline color is SAME.
- To define specific patterns and colors for the slices, use PATTERN statements and specify both the VALUE= and COLOR= options. If you provide fewer PATTERN definitions than the chart requires, the GCHART procedure uses the default pattern rotation for the slices that are drawn after all defined patterns are exhausted.

See “About Patterns” on page 998 for more information about how the GCHART procedure uses patterns and outlines. See the “PATTERN Statement” on page 455 for a description of default star patterns.

Modifying the Statistic Heading and the Group Heading

By default, the procedure prints a heading at the top of each chart indicating the type of statistic charted and the name of the chart variable. For an example see Figure 36.112 on page 987. You can suppress this heading with the NOHEADING option.

When you use the GROUP= option, a heading is printed above each star indicating the name of the group variable and its value for the particular star. An example is `SITE=Paris`. You can suppress these headings with the NOGROUPHEADING option. You can also suppress the variable name *SITE*= so that only the value *Paris* remains. To do this, use a LABEL statement and assign a null value to the variable name, as shown in this example:

```
label site="00"x;
```

The AXIS statement cannot be used by the STAR statement. Instead, you should use the “FTEXT Graphics Option” on page 626 and “HTEXT Graphics Option” on page 650 graphics options to control the font and height of text on the chart. Increasing the value of HTEXT= decreases the size of the star if any slice labels are positioned outside.

Examples: GCHART Procedure

Example 1: Specifying the Sum Statistic in a Block Chart

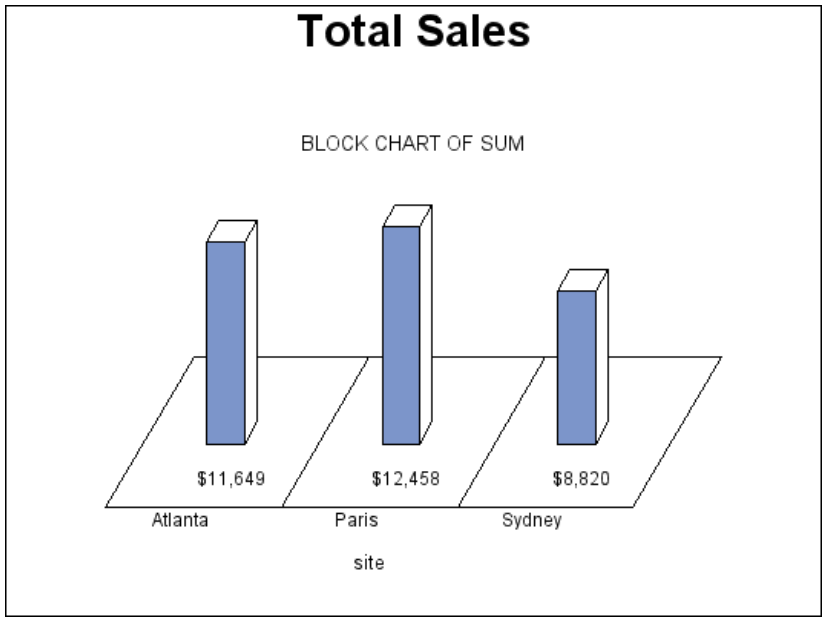
Features: BLOCK statement option SUMVAR=
FORMAT statement and the GOPTIONS statement option BORDER

Sample library member: GCHBKSUM

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces a block chart of total sales for three sites by charting the values of the character variable SITE and calculating the sum of the variable SALES for each site. It prints formatted values of the sales statistics below the blocks.

All the blocks use the same pattern because by default patterns change for subgroups and in this example subgroups are not specified.



Program

```
goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;

title "Total Sales";

proc gchart data=totals;
format sales dollar8.;
block site / sumvar=sales;
run;
quit;
```

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;
```

Define the title. This title appears on the chart.

```
title "Total Sales";
```

Produce the block chart. The BLOCK statement produces a block chart. SUMVAR= calculates the sum of SALES for each value of the chart variable SITE. With SUMVAR= the default statistic is SUM. The variable SALES is assigned a dollar format.

```
proc gchart data=totals;
format sales dollar8.;
block site / sumvar=sales;
```

```
run;
quit;
```

Example 2: Grouping and Subgrouping a Block Chart

Features: BLOCK statement options GROUP=, LEGEND=, MIDPOINTS=, NOHEADING, SUBGROUP=, and TYPE=

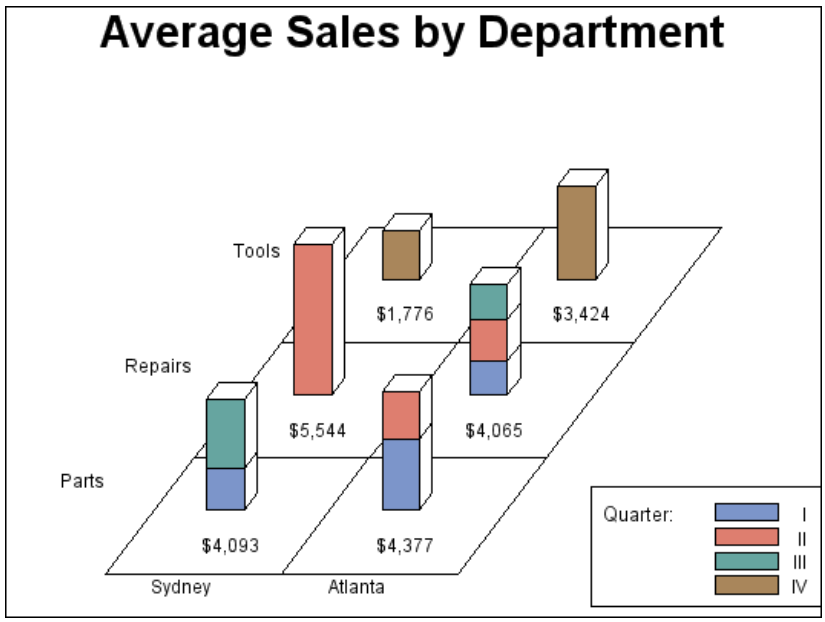
GOPTION statement option BORDER, LABEL statement, LEGEND statement, and default pattern rotation

Sample library member: GCHBKGRP

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example shows average quarterly sales for each department at two of the three manufacturing sites in the TOTALS data set; it excludes the Paris site from the chart. The program groups the chart data (sites) by department, and subgroups department sales data by quarter. Each site is a midpoint. Because the sites are grouped by department, each midpoint has a separate square for each department and the height of the block represents total sales for that department.

The blocks are subgrouped to show how quarterly sales contribute to total sales; each segment represents sales for a quarter. A legend explaining the subgroup patterns appears below the midpoint grid. The subgroups use four default patterns and colors that are retrieved from the current style. The patterns are created by rotating the default fill, solid, through the color list that is defined in the current style.



Program

```
goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 3043.97
Parts Sydney 3 5142.63
Parts Atlanta 1 5225.26
Parts Atlanta 2 3529.06
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Repairs Sydney 2 5543.97
Repairs Atlanta 1 3788.93
Repairs Atlanta 2 4492.89
Repairs Atlanta 3 3914.25
;

title "Average Sales by Department";

legend1 cborder=black
label=("Quarter:")
position=(bottom right outside)
mode=protect
across=1;

proc gchart data=totals;
format quarter roman.;
format sales dollar8.;
label site="00"x dept="00"x;

block site / sumvar=sales
type=mean
midpoints="Sydney" "Atlanta"
group=dept
subgroup=quarter
legend=legend1
noheading;
run;
quit;
```

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create data set TOTALS. TOTALS contains quarterly sales data for two of the three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
```

```
datalines;
Parts Sydney 1 3043.97
Parts Sydney 3 5142.63
Parts Atlanta 1 5225.26
Parts Atlanta 2 3529.06
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Repairs Sydney 2 5543.97
Repairs Atlanta 1 3788.93
Repairs Atlanta 2 4492.89
Repairs Atlanta 3 3914.25
;
```

Define the title. This title appears on the chart.

```
title "Average Sales by Department";
```

Define legend characteristics. LABEL= assigns new text to the legend label. CBORDER= draws a black frame around the legend.

```
legend1 cborder=black
label=("Quarter:")
position=(bottom right outside)
mode=protect
across=1;
```

Produce the block chart. The LABEL statement suppresses the midpoint and group labels by assigning a null hexadecimal string to each variable name.

```
proc gchart data=totals;
format quarter roman.;
format sales dollar8.;
label site="00"x dept="00"x;
```

The BLOCK statement produces the block chart. The TYPE= option specifies the chart statistic as the mean value of the summary variable SALES for each site. The MIDPOINTS= option selects the two sites and the order in which they appear. The GROUP= option creates a separate row of blocks for each different value of DEPT. The SUBGROUP= option divides each block into separate segments for the four quarters. The LEGEND= option assigns the LEGEND1 statement to the graph. NOHEADING suppresses the default heading that would otherwise appear above the chart.

```
block site / sumvar=sales
type=mean
midpoints="Sydney" "Atlanta"
group=dept
subgroup=quarter
legend=legend1
noheading;
run;
quit;
```

Example 3: Specifying the Sum Statistic in Bar Charts

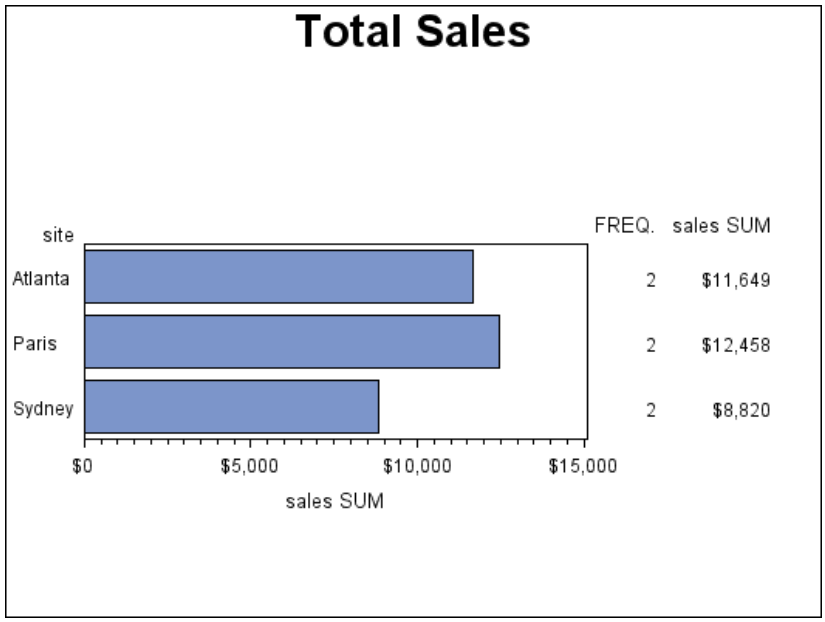
Features: HBAR statement option SUMVAR= and VBAR3D statement option SUMVAR=
FORMAT statement, GOPTIONS statement option BORDER, and RUN-group processing

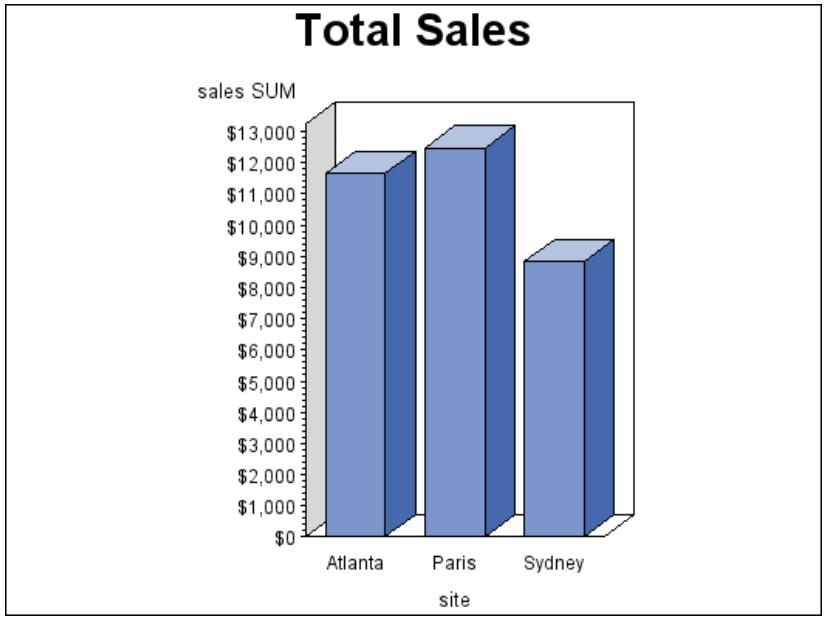
Sample library member: GCHBRSUM

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces two bar charts that show the total sales for three sites by charting the values of the character variable SITE and calculating the sum of the variable SALES for each site. In the first horizontal bar chart shown, the summary statistics are printed by default to the right of the bars and display the formatted values of SALES. The output also shows the frame that is drawn by default around the axis area.

The second bar chart is a three-dimensional vertical bar chart, shown in the following output. Vertical bar charts do not generate a table of statistics and by default do not print any chart statistics.





Program

```
goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;

title1 "Total Sales";

proc gchart data=totals;
format sales dollar8.;
hbar site / sumvar=sales;
run;

  vbar3d site / sumvar=sales;
run;
quit;
```

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;
```

Define the title. This title appears on the chart.

```
title1 "Total Sales";
```

Produce the horizontal bar chart. The HBAR statement produces a two-dimensional bar chart. SUMVAR= calculates the sum of SALES for each value of the chart variable SITE. The default statistic for SUMVAR= is SUM. The variable SALES is assigned a dollar format. The embedded RUN statement is required to end this first RUN-Group and honor the action statement and other SAS statements. It signals that the procedure is to remain active.

```
proc gchart data=totals;
format sales dollar8.;
hbar site / sumvar=sales;
run;
```

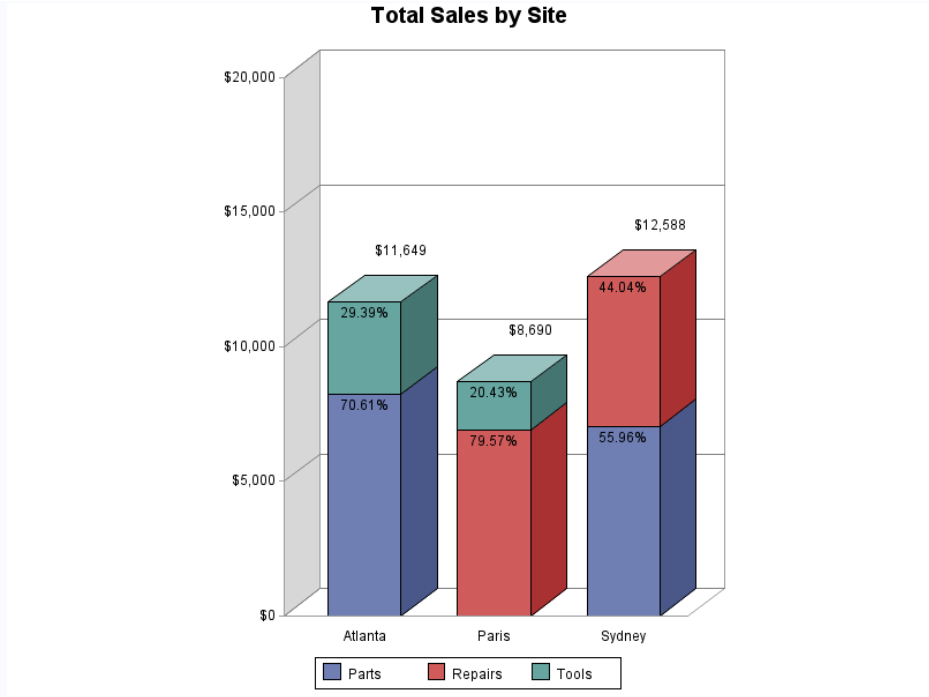
Produce the vertical bar chart. Because the procedure supports RUN-group processing, you do not have to repeat the PROC GCHART statement to generate the second chart. The VBAR3D statement produces a three-dimensional vertical bar chart.

```
  vbar3d site / sumvar=sales;
run;
quit;
```

Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart

Features:	VBAR statement options CFRAME=, INSIDE=SUBPCT, LEGEND=, MAXIS=, OUTSIDE=SUM, RAXIS=, SPACE=, SUBGROUP=, and WIDTH= AXIS statement, FORMAT statement, GOPTIONS statement option BORDER, and LEGEND statement
Sample library member:	GCHBRGRP
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example subgroups by department the three-dimensional vertical bar chart of total sales for each site that is shown in "Example 3: Specifying the Sum Statistic in Bar Charts" on page 1098. In addition to subdividing the bars to show the amount of sales for each department for each site, the chart displays statistics both inside and outside of the bars. OUTSIDE=SUM prints the total sales for the site above each bar. INSIDE=SUBPCT prints the percent each department contributed to the total sales for its site inside each subgroup segment.



Program

```
goptions reset=all cback=white;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Tools Paris 4 1775.74
Tools Atlanta 4 3424.19
Repairs Sydney 2 5543.97
Repairs Paris 3 6914.25
;

title1 "Total Sales by Site";

axis1 label=none offset=(10,8);

axis2 label=none order=(0 to 20000 by 5000) minor=none offset=(,0);

legend1 label=none shape=bar(.15in,.15in) cborder=black;

proc gchart data=totals;
format sales dollar8.;
```

```
vbar3d site / sumvar=sales subgroup=dept inside=subpct
     outside=sum
     width=9
     space=7
     maxis=axis1
     raxis=axis2
     cframe=white
     autoref cref=gray
     legend=legend1;
run;
quit;
```

Program Description

Set the graphics environment. The CBACK option in the GOPTIONS statement changes the graph background color to white.

```
goptions reset=all cback=white;
```

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Tools Paris 4 1775.74
Tools Atlanta 4 3424.19
Repairs Sydney 2 5543.97
Repairs Paris 3 6914.25
;
```

Define the title. This title appears on the chart.

```
title1 "Total Sales by Site";
```

Modify the midpoint axis. The LABEL= option suppresses the axis label. The OFFSET= option moves the first bar from the beginning of the axis line and moves the last bar from the end of the axis line.

```
axis1 label=none offset=(10,8);
```

Modify the response axis. The LABEL= option suppresses the axis label. The ORDER= option specifies the major tick values for the response axis. The MINOR= option suppresses the minor tick marks. The OFFSET= option moves the top tick mark to the end of the axis line.

```
axis2 label=none order=(0 to 20000 by 5000) minor=none offset=(,0);
```

Modify the legend. The LABEL= option suppresses the legend label. The SHAPE= option defines the size of the legend values. The CBORDER= option draws a black frame around the legend.

```
legend1 label=none shape=bar(.15in,.15in) cborder=black;
```

Produce the vertical bar chart. The SUBGROUP= option creates a separate bar segment for each department. The INSIDE= option prints the subgroup percent statistic inside each bar segment. The OUTSIDE= option prints the sum statistic

above each bar. The WIDTH= option makes the bars wide enough to display the statistics. The SPACE= option controls the space between the bars. The MAXIS= option assigns the AXIS1 statement to the midpoint axis. The RAXIS= option assigns the AXIS2 statement to the response axis. The CFRAME= option specifies the color for the three-dimensional planes. The AUTOREF option draws a gray reference line at each major tick mark on the response axis. The LEGEND= option assigns the LEGEND1 statement to the subgroup legend.

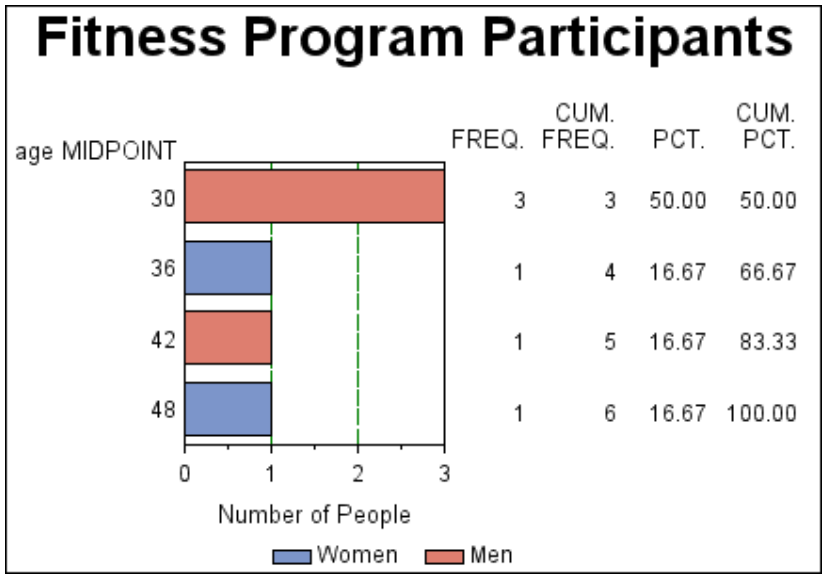
```
proc gchart data=totals;
format sales dollar8.;
vbar3d site / sumvar=sales subgroup=dept inside=subpct
     outside=sum
     width=9
     space=7
     maxis=axis1
     raxis=axis2
     cframe=white
     autoref cref=gray
     legend=legend1;
run;
quit;
```

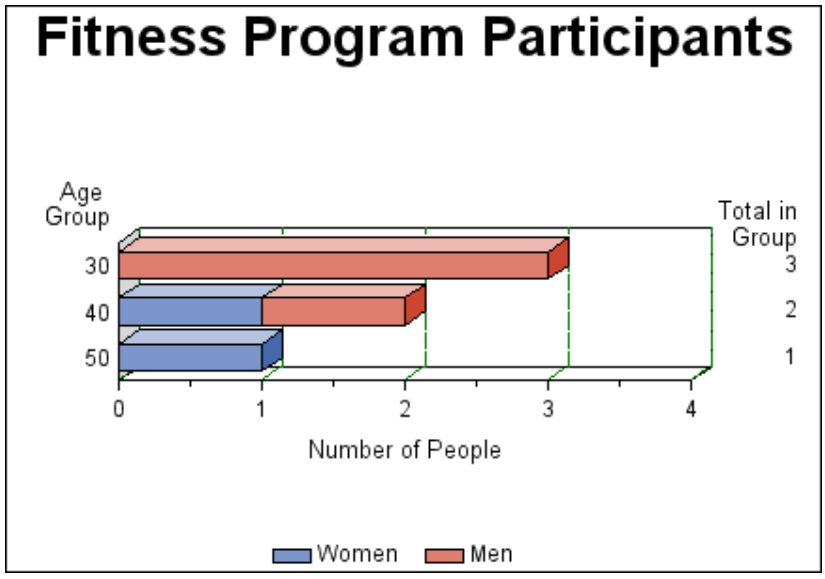
Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart

Features:	HBAR statement options AUTOREF, COUTLINE=, CLIPREF, and SUBGROUP=; and HBAR3D statement options FREQ, FREQLABEL=, and MIDPOINTS= GOPTIONS statement option BORDER, AXIS statement, LEGEND statement, and RUN-group processing
Sample library member:	GCHBRMID
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses the FITNESS data set to produce a horizontal bar chart that shows the number of people in each age group in a fitness program. It charts the numeric variable AGE with the frequency statistic. Because the values of AGE are continuous, the procedure automatically divides the ages into ranges and displays the midpoint of each age range. The frequency statistic calculates the number of observations in each range. The chart statistic defaults to FREQ because the SUMVAR= and TYPE= options are omitted. The table of statistics displays all the statistic values.

The second part of this example modifies the midpoint axis and the table of statistics, and uses RUN-group processing to produce the following chart. This part of the program specifies the midpoint value for each bar and requests only the FREQ statistic for the table.





Program

```
goptions reset=all border;

data fitness;
input age sex $ heart exer aero;
datalines;
28 M 86 2 36.6
41 M 76 3 26.7
30 M 78 2 33.8
29 M 54 3 44.8
48 F 66 2 28.9
36 F 66 2 33.2
;

title1 "Fitness Program Participants";
```

```
axis1 label=("Number of People")
minor=(number=1)
offset=(0,0);

legend1 label=none
value=("Women" "Men");

proc gchart data=fitness;
hbar age / subgroup=sex
legend=legend1
     autoref
     clipref
     raxis=axis1
     wref=1
     cref=green
     lref=4
;
run;

axis1 order=(0 to 4 by 1)
label=("Number of People")
minor=(number=1)
offset=(0,0);

axis2 label=("Age " j=r "Group");

hbar3d age / midpoints=(30 40 50)
     freq freqlabel="Total in Group"
     subgroup=sex
     autoref
     maxis=axis2
     raxis=axis1
     legend=legend1
     coutline=black
     clipref
     wref=1
     cref=green
     lref=4
 ;
run;
quit;
```

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create the data set FITNESS. FITNESS contains age and sex of participants, as well as the number of times they exercise each week and their resting heart rate and aerobic power.

```
data fitness;
input age sex $ heart exer aero;
datalines;
28 M 86 2 36.6
41 M 76 3 26.7
```

```
30 M 78 2 33.8
29 M 54 3 44.8
48 F 66 2 28.9
36 F 66 2 33.2
;
```

Define the title. This title appears on the chart.

```
title1 "Fitness Program Participants";
```

Modify the response axis. The OFFSET= option moves the first and last tick marks to the ends of the axis line. The ORDER= option places major tick marks on the response axis from 1 to 14.

```
axis1 label=("Number of People")
minor=(number=1)
offset=(0,0);
```

Modify the legend. The VALUE= option specifies the text that describes the values.

```
legend1 label=none
value=("Women" "Men");
```

Produce the first horizontal bar chart. Because neither the MIDPOINTS= option nor the DISCRETE option is used, the procedure automatically selects the midpoints. The SUBGROUP= option divides the bars according to the values of SEX and automatically generates a legend. The AUTOREF option adds reference lines to the chart at each major tick mark. The CLIPREF option positions the reference lines behind the bars. The WREF= option sets the reference line width to 1 pixel. The CREF= option sets the reference line color to green. The LREF= option sets the reference line type to 4 (dashed line). The embedded RUN statement is required to end this first RUN-Group and honor the action statement and other SAS statements. It signals that the procedure is to remain active.

```
proc gchart data=fitness;
hbar age / subgroup=sex
legend=legend1
     autoref
     clipref
     raxis=axis1
     wref=1
     cref=green
     lref=4
;
run;
```

Modify the response axis for the second chart. The ORDER= option places major tick marks on the response axis at intervals of 1.

```
axis1 order=(0 to 4 by 1)
label=("Number of People")
minor=(number=1)
offset=(0,0);
```

Modify the midpoint axis label for the second chart.

```
axis2 label=("Age " j=r "Group");
```

Produce the second horizontal bar chart with modified midpoints. The MIDPOINTS= option specifies the middle value of the range of values represented by each bar. The FREQ option requests that only the frequency statistic appears in the table. The FREQLABEL= option specifies the text for the column heading in the table of statistics.

```
hbar3d age / midpoints=(30 40 50)
     freq freqlabel="Total in Group"
     subgroup=sex
     autoref
     maxis=axis2
     raxis=axis1
     legend=legend1
     coutline=black
     clipref
     wref=1
     cref=green
     lref=4
 ;
run;
quit;
```

Example 6: Generating Error Bars in a Horizontal Bar Chart

Features: HBAR statement options CLM=, ERRORBAR=, FREQLABEL=, MEANLABEL=, NOFRAME, SUMVAR=, and TYPE=

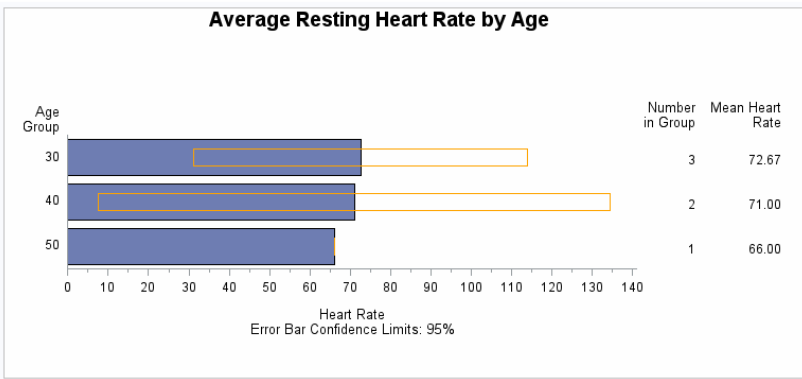
HBAR statement options:

GOPTIONS statement option BORDER and AXIS statement

Sample library member: GCHERRBR

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses the FITNESS data set to chart the mean heart rate for each age group with error bars showing the confidence limits for the average. The response axis label describes the confidence limit for the error bars. To make the error bars easier to read, the program suppresses the frame that the procedure draws around the axis area. Descriptive column head labels in the table of statistics replace the statistic names that appear by default.



Program

```
goptions reset=all border;

data fitness;
input age sex $ heart exer aero;
datalines;
28 M 86 2 36.6
41 M 76 3 26.7
30 M 78 2 33.8
29 M 54 3 44.8
48 F 66 2 28.9
36 F 66 2 33.2
;

title1 "Average Resting Heart Rate by Age";

axis1 label=("Heart Rate" j=c "Error Bar Confidence Limits: 95%")
minor=(number=1);
axis2 label=("Age" j=r "Group");

proc gchart data=fitness;
hbar age / type=mean
sumvar=heart
freqlabel="Number in Group"
meanlabel="Mean Heart Rate"
errorbar=bars
cerror=orange
clm=95
midpoints=(30 40 50)
raxis=axis1
maxis=axis2
noframe;
run;
quit;
```

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create the data set FITNESS. FITNESS contains age and sex of participants, as well as the number of times they exercise each week and their resting heart rate and aerobic power.

```
data fitness;
input age sex $ heart exer aero;
datalines;
28 M 86 2 36.6
41 M 76 3 26.7
30 M 78 2 33.8
29 M 54 3 44.8
48 F 66 2 28.9
36 F 66 2 33.2
```

```
;
```

Define the title. This title appears on the chart.

```
title1 "Average Resting Heart Rate by Age";
```

Modify the axis labels. AXIS1 is assigned to the response axis and AXIS2 is assigned to the midpoint axis.

```
axis1 label=("Heart Rate" j=c "Error Bar Confidence Limits: 95%")
minor=(number=1);
axis2 label=("Age" j=r "Group");
```

Produce the horizontal bar chart. The SUMVAR= option calculates the mean of the variable HEART for all the observations in each midpoint group. The TYPE= option specifies the mean statistic for the summary variable, HEART. The FREQLABEL= and MEANLABEL= options specify new column labels for the frequency and mean statistics. The ERRORBAR= option draws the error bars as empty bars and CLM= specifies the confidence level. The NOFRAME option suppresses the axis area frame.

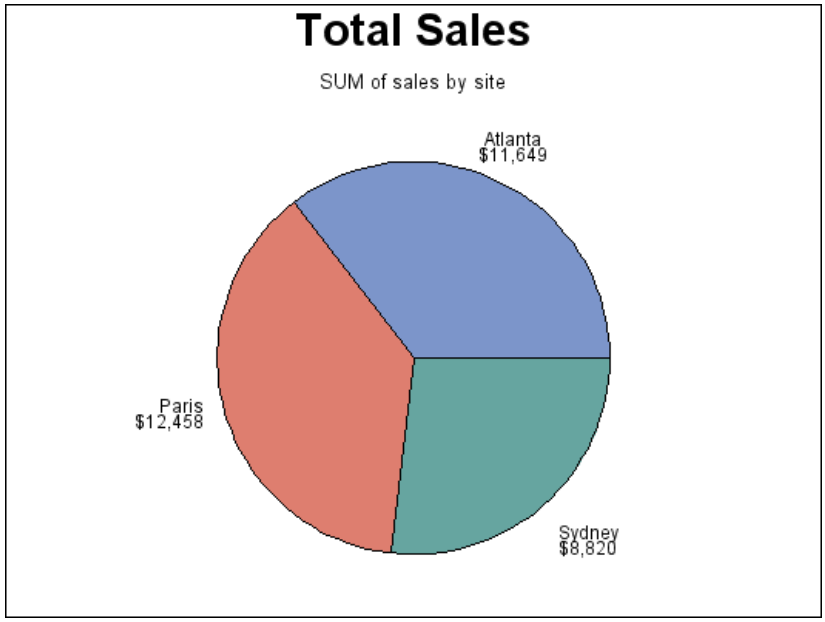
```
proc gchart data=fitness;
hbar age / type=mean
sumvar=heart
freqlabel="Number in Group"
meanlabel="Mean Heart Rate"
errorbar=bars
cerror=orange
clm=95
midpoints=(30 40 50)
raxis=axis1
maxis=axis2
noframe;
run;
quit;
```

Example 7: Specifying the Sum Statistic for a Pie Chart

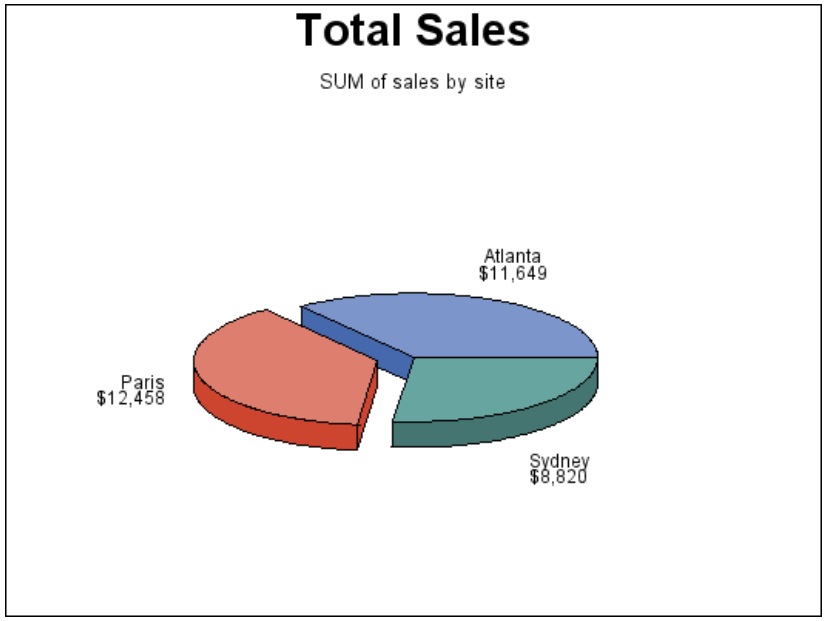
Features:	PIE statement option SUMVAR=, and PIE3D statement options EXPLODE= and SUMVAR= GOPTIONS statement option BORDER, FORMAT statement, and RUN-group processing
Sample library member:	GCHPISUM
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces two pie charts that show total sales for three sites by charting the values of the character variable SITE and calculating the sum of the variable SALES for each site. It represents the statistics as slices of the pie. By default, the midpoint value and the summary statistic are printed beside each slice.

The pie slices use the default pattern fill, which is solid. Each slice displays a different color because, by default, pie charts are patterned by midpoint.



The second pie chart is a three-dimensional pie chart with an exploded slice, as shown in the following output.



Program

```
goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
```

```
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;

title "Total Sales";

proc gchart data=totals;
format sales dollar8.;
pie site / sumvar=sales;
run;

pie3d site / sumvar=sales
explode="Paris";
run;
quit;
```

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;
```

Define the title. This title appears on the chart.

```
title "Total Sales";
```

Produce the first pie chart. The pie statement produces a two dimensional pie chart. The SUMVAR= option calculates the sum of SALES for each value of the chart variable SITE. The default statistic for the SUMVAR= option is SUM. The variable SALES is assigned a dollar format. The embedded RUN statement is required to end this first RUN-Group and honor the action statement and other SAS statements. It signals that the procedure is to remain active.

```
proc gchart data=totals;
format sales dollar8.;
pie site / sumvar=sales;
run;
```

Produce the second pie chart. The PIE3D statement produces a three-dimensional pie chart. The EXPLODE= option separates the slice for PARIS from the rest of the pie.

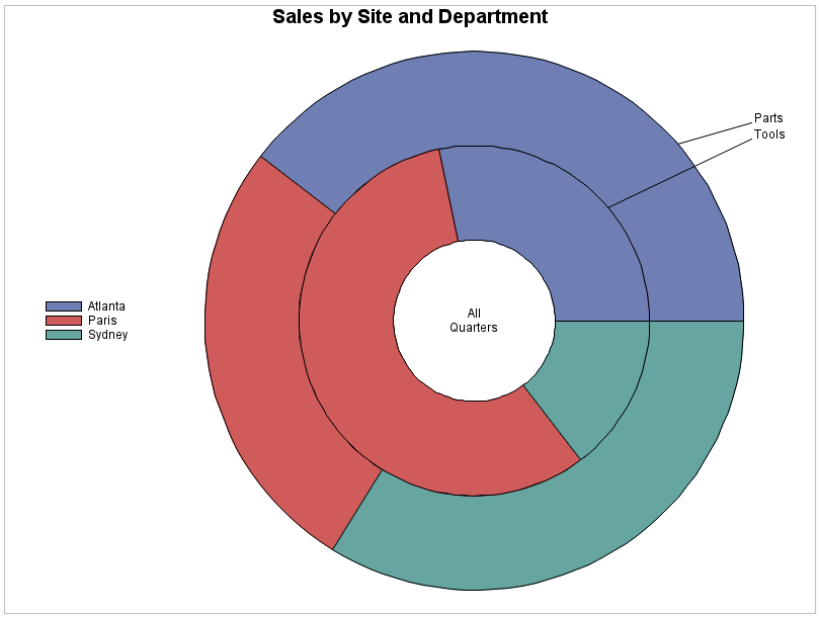
```
pie3d site / sumvar=sales
explode="Paris";
run;
quit;
```

Example 8: Subgrouping a Donut or Pie Chart

Features:	DONUT statement options DONUTPCT=, LABEL=, LEGEND=, NOHEADING, and SUBGROUP= GOPTIONS statement option BORDER and LEGEND statement
Sample library member:	GCHSBGRP
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces a donut chart that is similar to the pie chart in “Example 7: Specifying the Sum Statistic for a Pie Chart” on page 1109. The similarity is that each slice represents total sales for a site and each slice is a different color. However, in this donut chart the sites are subgrouped by department, so that each department is represented as a concentric ring with slices. Subgrouping suppresses the chart statistic and the midpoint labels. Instead, it automatically labels the rings with the subgroup values and generates a legend that shows how the patterns are associated with the midpoint values. Subgrouping a pie chart produces the same results but without the hole in the center.

To make the donut chart as large as possible, the program suppresses the default heading and moves the legend into the space at the left of the chart.



Program

```
goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;

title "Sales by Site and Department";

legend1 label=none
position=(middle left)
offset=(5,)
across=1;

proc gchart data=totals;
format sales dollar8.;
donut site / sumvar=sales
subgroup=dept
donutpct=30
label=("All" justify=center "Quarters")
noheading
legend=legend1;
run;
quit;
```

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;
```

Define the title. This title appears on the chart.

```
title "Sales by Site and Department";
```

Modify the subgroup legend. The LABEL= option suppresses the legend label. The POSITION=, the OFFSET=, and the ACROSS= options arrange the legend entries in a column to the left of the pie chart.

```
legend1 label=none
position=(middle left)
offset=(5,)
across=1;
```

Produce the donut chart. The SUBGROUP= option divides the donut into rings. Each ring represents a value of the subgroup variable, DEPT. The DONUTPCT= option controls the size of the donut hole, which contains the text specified by the LABEL= option. The NOHEADING option suppresses the default heading that contains the name of the chart variable and the type of statistic. The LEGEND= option assigns the LEGEND1 statement to the chart.

```
proc gchart data=totals;
format sales dollar8.;
donut site / sumvar=sales
subgroup=dept
donutpct=30
label=("All" justify=center "Quarters")
noheading
legend=legend1;
run;
quit;
```

Example 9: Ordering and Labeling Slices in a Pie Chart

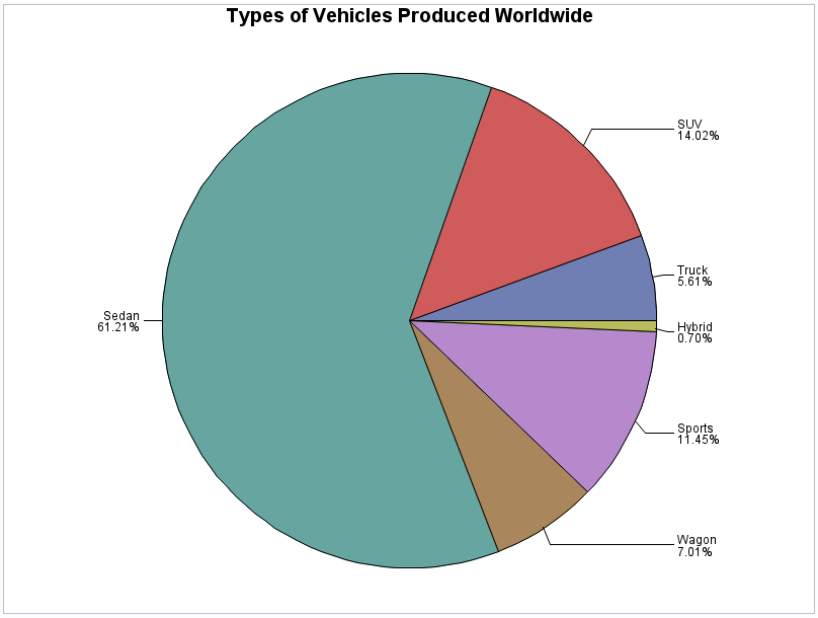
Features: PIE statement options MIDPOINTS=, PERCENT=ARROW, SLICE=ARROW, and VALUE=NONE

GOPTIONS Statement option BORDER

Sample library member: GCHLABEL

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces a pie chart of the types of vehicles produced worldwide. The labeled slices represent the percent of total production for each source. Instead of the sum statistic, each slice displays the percent each midpoint contributes to the whole pie. Arrows connect the midpoint labels to the slices. The slices are arranged by the MIDPOINTS= option so that similar types of vehicles are shown next to each other in the pie chart.



Program

```
goptions reset=all border;

title "Types of Vehicles Produced Worldwide";

proc gchart data=sashelp.cars;
pie type / other=0
```

```
midpoints="Truck" "SUV" "Sedan" "Wagon" "Sports" "Hybrid"
value=none
percent=arrow
slice=arrow
noheading;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title. This title appears on the chart.

```
title "Types of Vehicles Produced Worldwide";
```

Produce the pie chart. This graph uses the data set entitled CARS found in the SASHELP library. OTHER=0 specifies that all midpoints, no matter how small, display a slice. The MIDPOINTS= option assigns the order of the slices. Each slice displays the percent contribution to total production and the slice name. VALUE=NONE suppresses the chart statistic. Both the SLICE= and PERCENT= options specify the ARROW labeling method to point to the slice, but only one arrow line is displayed.

```
proc gchart data=sashelp.cars;
pie type / other=0
midpoints="Truck" "SUV" "Sedan" "Wagon" "Sports" "Hybrid"
value=none
percent=arrow
slice=arrow
noheading;
run;
quit;
```

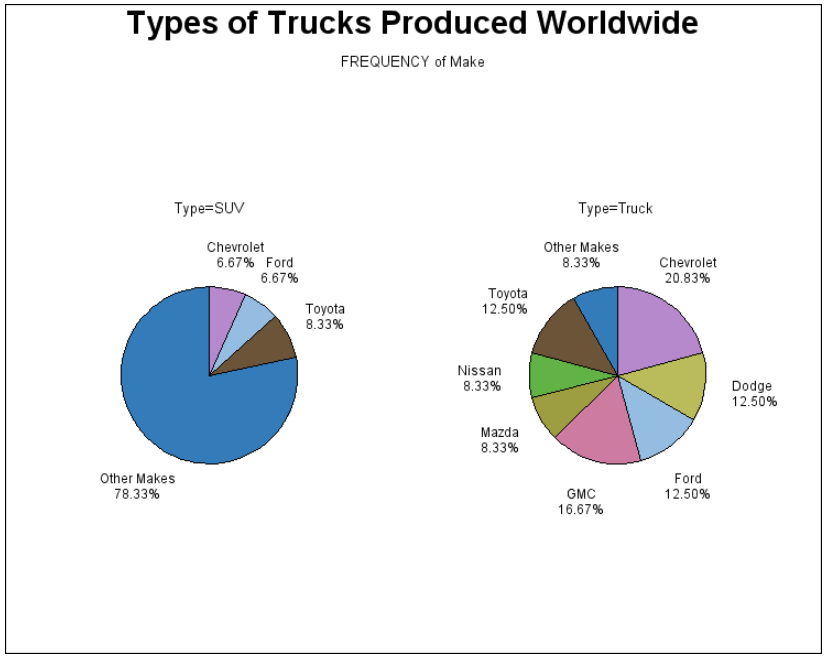
Example 10: Grouping and Arranging Pie Charts

Features:	PIE statement options ACROSS=, CLOCKWISE, GROUP=, OTHER=, PERCENT=OUTSIDE, and SLICE=OUTSIDE GOPTIONS statement option BORDER
Sample library member:	GCHPIGRP
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces two pie charts that show the production of trucks worldwide. Both charts are displayed on one page and are arranged two across. The program

uses the CLOCKWISE option to arrange the slices, which begin at the 12 o'clock position and proceed clockwise in alphabetic order of the midpoint.

The chart statistic is suppressed and the midpoint label and the percent of the chart statistic are displayed outside of the slice.



Program

```
goptions reset=all border;

title "Types of Trucks Produced Worldwide";

proc gchart data=sashelp.cars(where=(type="SUV" or type="Truck"));
  pie make / group=type
             across=2
             other=5 otherlabel="Other Makes"
             clockwise value=none
             slice=outside percent=outside;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title. This title appears on the chart.

```
title "Types of Trucks Produced Worldwide";
```

Produce the pie charts. This graph uses the data set entitled CARS found in the SASHELP library. The GROUP= option creates a separate pie for each model. In

combination with the GROUP= option, the ACROSS= option draws two charts across one page. The OTHER= option collects all the midpoints with statistic values less than or equal to 5% of the total into one slice. CLOCKWISE begins drawing the slices at the 12 o'clock position in alphabetic order of the midpoint. The PERCENT=OUTSIDE and SLICE=OUTSIDE display the labels outside the slices.

```
proc gchart data=sashelp.cars(where=(type="SUV" or type="Truck"));
  pie make / group=type
             across=2
             other=5 otherlabel="Other Makes"
             clockwise value=none
             slice=outside percent=outside;
run;
quit;
```

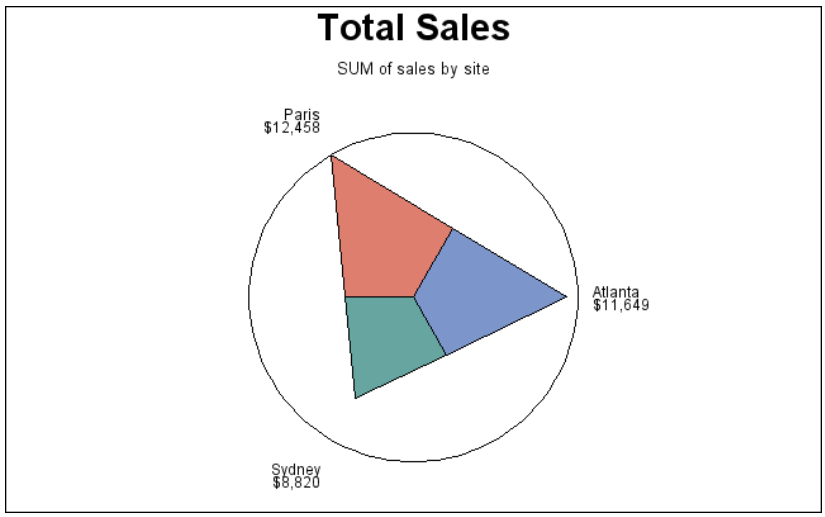
Example 11: Specifying the Sum Statistic in a Star Chart

Features: STAR statement option SUMVAR=
FORMAT statement

Sample library member: GCHSTSUM

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces a star chart of total sales for three sites by charting the values of the character variable SITE and calculating the sum of the variable SALES for each site. It represents the statistics as slices of the star. The center of the circle represents 0 and the edge of the circle represents the largest value, in this case Paris sales. By default, the spines are joined and filled with a solid pattern to form slices. In addition, the midpoint value and the formatted values of the sales statistics are printed beside each slice.



Program

```
goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;

title "Total Sales";

proc gchart data=totals;
format sales dollar8.;
star site / sumvar=sales;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;
```

Define the title. This title appears on the chart.

```
title "Total Sales";
```

Produce the star chart. The SUMVAR= option calculates the sum of SALES for each value of the chart variable SITE. Because the TYPE= option is omitted, the default statistic is sum. The FORMAT statement assigns a format to the variable SALES.

```
proc gchart data=totals;
format sales dollar8.;
star site / sumvar=sales;
run;
```

```
quit;
```

Example 12: Charting a Discrete Numeric Variable in a Star Chart

Features: STAR statement options DISCRETE, FILL=, NOCONNECT, NOHEADING, and SUMVAR=

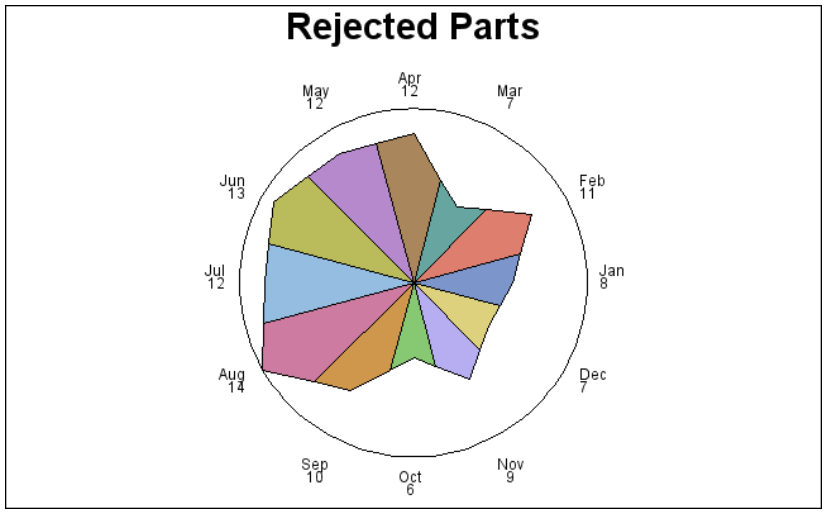
GOPTIONS statement option BORDER

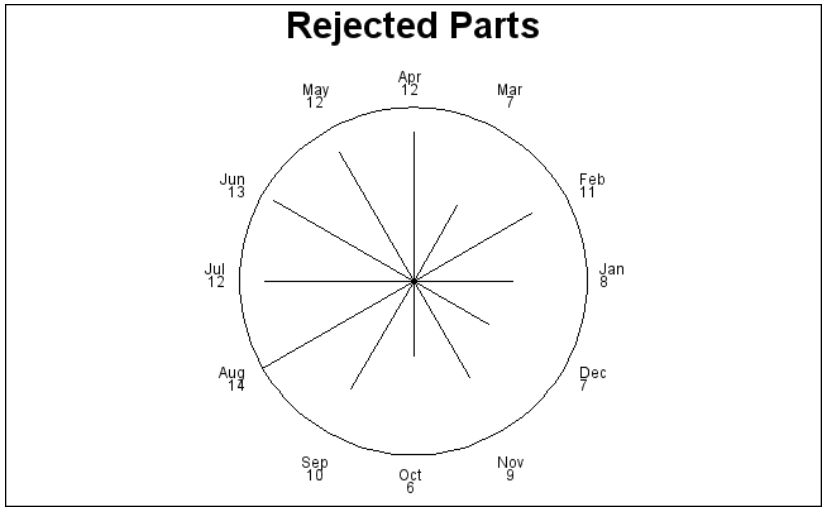
Sample library member: GCHDSCRT

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces two star charts that show the total number of parts that were rejected each month for a year. The STAR statement uses the DISCRETE option so that each unique value of the numeric variable DATE is a separate midpoint and has a separate spine. Each slice displays the formatted midpoint value and the chart statistic. Specifying FILL=S rotates the solid pattern through all the colors in the style's list of colors as many times as necessary to provide patterns for all the slices.

The second chart uses the NOCONNECT option so that the chart uses spines instead of slices.





Program

```
goptions reset=all border;

data rejects;
informat date date9.;
input site $ date badparts;
datalines;
Sydney 01JAN1997 8
Sydney 01FEB1997 11
Sydney 28JUN1997 13
Sydney 31OCT1997 6
Paris 11APR1997 12
Paris 04MAY1997 12
Paris 30AUG1997 14
Paris 01DEC1997 7
Atlanta 15MAR1997 7
Atlanta 18JUL1997 12
Atlanta 03SEP1997 10
Atlanta 12NOV1997 9
;

title "Rejected Parts";

proc gchart data=rejects;
format date worddate3.;
star date / discrete
sumvar=badparts
noheading
fill=s;
run;

star date / discrete
sumvar=badparts
noheading
noconnect;
run;
quit;
```

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create the data set REJECTS. REJECTS contains data on the number of defective parts produced at each of three sites for 12 months. BADPARTS is the number of parts that were rejected at each site for each month.

```
data rejects;
informat date date9.;
input site $ date badparts;
datalines;
Sydney 01JAN1997 8
Sydney 01FEB1997 11
Sydney 28JUN1997 13
Sydney 31OCT1997 6
Paris 11APR1997 12
Paris 04MAY1997 12
Paris 30AUG1997 14
Paris 01DEC1997 7
Atlanta 15MAR1997 7
Atlanta 18JUL1997 12
Atlanta 03SEP1997 10
Atlanta 12NOV1997 9
;
```

Define the title. This title appears on the chart.

```
title "Rejected Parts";
```

Produce the first star chart. The DISCRETE option must be specified because the numeric chart variable, DATE is assigned the WORDDATE3. Using FILL=S fills all the slices with solid patterns. The embedded RUN statement is required to end this first RUN-Group and honor the action statement and other SAS statements. It signals the procedure to remain active.

```
proc gchart data=rejects;
format date worddate3.;
star date / discrete
sumvar=badparts
noheading
fill=s;
run;
```

Produce the second star chart with slices and a solid fill. The NOHEADING option suppresses the default heading for the star chart. The NOCONNECT option suppresses the lines that by default join the ends of the spines.

```
star date / discrete
sumvar=badparts
noheading
noconnect;
run;
quit;
```

Example 13: Creating a Detail Pie Chart

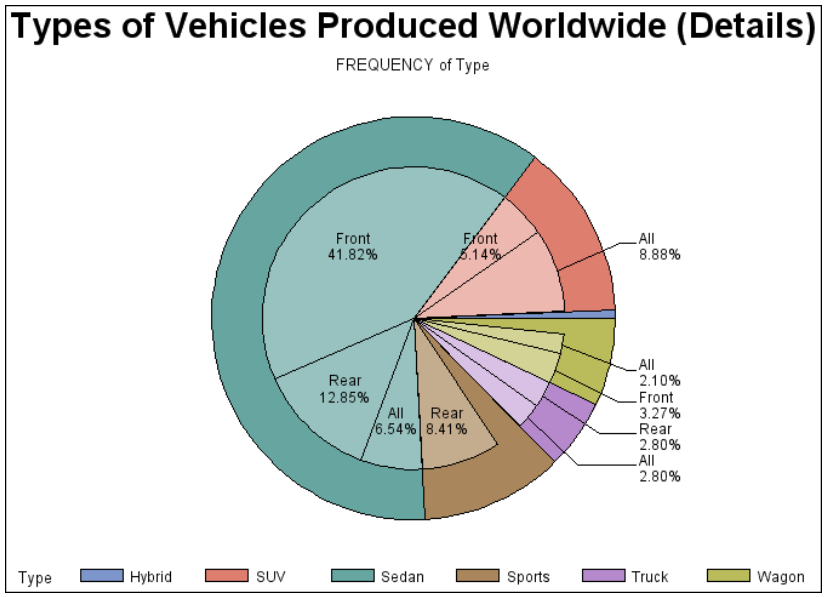
Features: PIE statement options DETAIL=, DETAIL_PERCENT=, DETAIL_SLICE=, DETAIL_THRESHOLD=, DETAIL_VALUE=, and LEGEND

GOPTIONS statement option BORDER

Sample library member: GCHDTPIE

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example produces a normal pie chart with a detail pie overlay. The pie chart shows the percentage of vehicle types produced worldwide. The detail pie overlay shows the percentage of DRIVETRAINS for each vehicle TYPE.



Program

```
goptions reset=all border;

title "Types of Vehicles Produced Worldwide (Details)";

proc gchart data=sashelp.cars;
pie type / detail=drivetrain
detail_percent=best
detail_value=none
detail_slice=best
detail_threshold=2
legend
```

```
;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title. This title appears on the chart.

```
title "Types of Vehicles Produced Worldwide (Details)";
```

Produce the detail pie chart. This graph uses the data set entitled CARS found in the SASHELP library. The DETAIL= option produces an inner pie overlay showing the percentage that each DRIVETRAIN contributes toward each type of vehicle. The DETAIL_PERCENT= option and the DETAIL_SLICE= option control the positioning of the detail slice labels. The DETAIL_VALUE= option turns off the display of the number of DRIVETRAINS for each detail slice. The DETAIL_THRESHOLD= option shows all detail slices that contribute more than 2% of the entire pie. The LEGEND option displays a legend for the slice names and their midpoint values, instead of printing them beside the slices.

```
proc gchart data=sashelp.cars;
pie type / detail=drivetrain
detail_percent=best
detail_value=none
detail_slice=best
detail_threshold=2
legend
;
run;
quit;
```

References

Nelder, J. A. 1976. “A Simple Algorithm for Scaling Graphs.” *Journal for the Royal Statistical Society, Series C (Applied Statistics)* 25 (1): 94–96.

Terrell, G. R., and D. W. Scott. 1985. “Oversmoothed Nonparametric Density Estimates.” *Journal of the American Statistical Association* 80 (389): 209–214.

Chapter 37

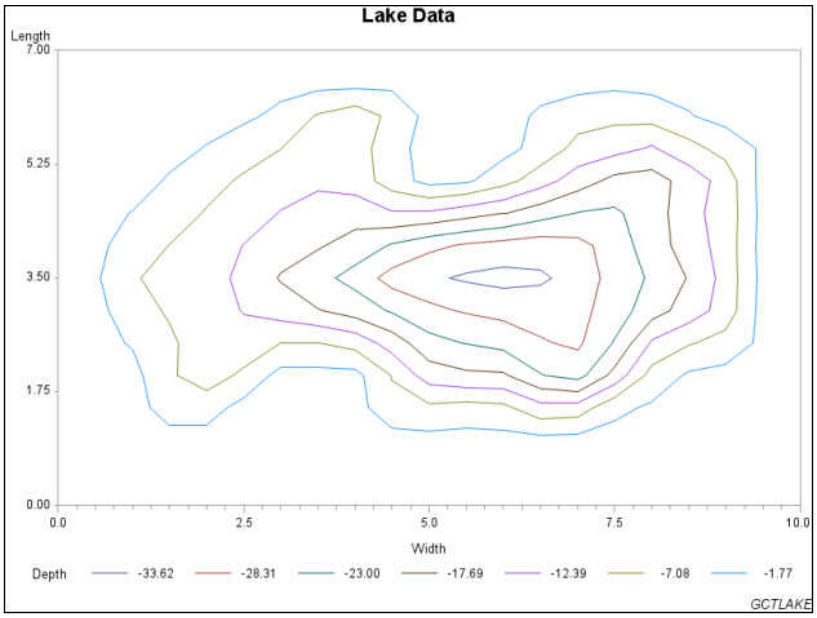
GCONTOUR Procedure

Overview: GCONTOUR Procedure

What Does the GCONTOUR Procedure Do?

The GCONTOUR procedure enables you to generate two-dimensional plots representing three-dimensional relationships. For example, the following Figure 37.123 on page 1126 displays various depths of a lake. The dimensions of the lake are plotted on the *x* and *y* axes. The *z* variable is plotted as the third dimension, and is displayed as uniquely colored contour lines.

Figure 37.1 Simple Contour Plot



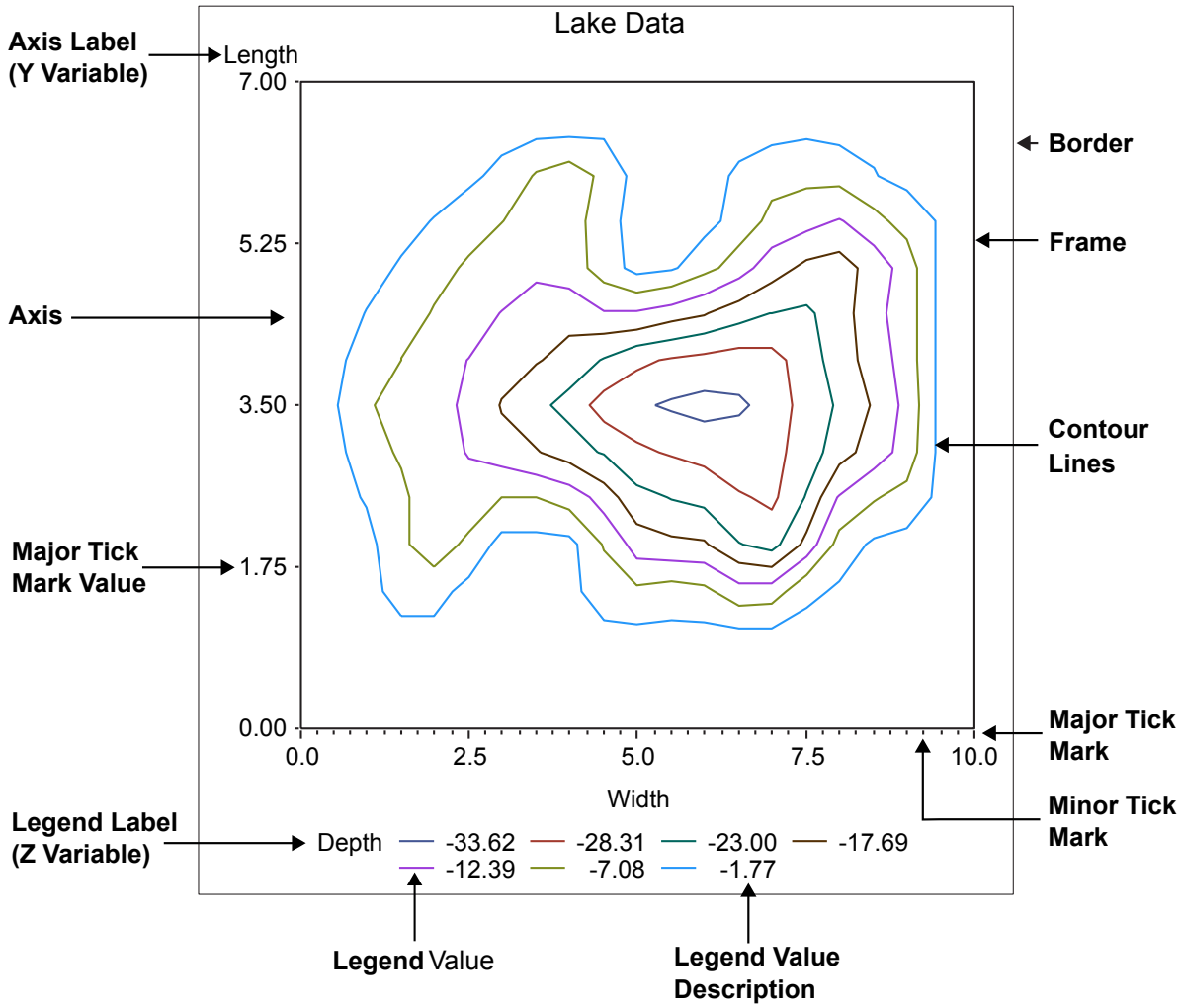
With PROC GCONTOUR, you can do the following actions:

- use AXIS statements to customize the axes
- use line styles and patterns to emphasize the contour levels
- use reference lines to see how *(x,y)* combinations align to *z* values
- use SYMBOL statements to customize labels or highlight data trends

Concepts: GCONTOUR Procedure

CONTOUR Plot Terms

Figure 37.2 GCONTOUR Procedure Terms



Input Data

The GCONTOUR procedure requires three numeric variables to produce a plot. The input data set forms a rectangular grid from the values of *x* and *y*. The *z* variable is plotted on the grid as the third dimension. Only one value of *z* is required for each *(x,y)* grid location. If multiple observations have the same *z* value for any *(x,y)* combination, only the last observation is plotted.

Data Ranges

PROC GCONTOUR produces a rectangular grid with axes scaled to include the minimum data values and maximum data values of *x* and *y*. Each axis is labeled with the variable name or label. The contour lines represent the levels of magnitude

by grouping the common values of the *z* variable. The level of each contour line is displayed in the legend. The legend label is the *z* variable's name or label.

Missing Values

PROC GCONTOUR requires data values for at least 50% of the *z* variable, for each unique combination of *(x,y)*. The INCOMPLETE option can be used to override this requirement. The G3GRID procedure can also be used to create data for missing values. (See Chapter 48, "G3GRID Procedure," on page 1555).

Interpolating Data

The G3GRID procedure enables you to produce a data set with nonmissing values for *z* for every unique *(x,y)* combination. The output data set from the G3GRID procedure can be used as the input data set for the GCONTOUR procedure. The G3GRID procedure also enables you to smooth data for use with GCONTOUR. For details see Chapter 48, "G3GRID Procedure," on page 1555. For an interpolation example see "Example 1: Using the Default Interpolation Method" on page 1567.

Syntax: GCONTOUR Procedure

Requirement: At least one PLOT statement is required.

Supported Statements: AXIS , FOOTNOTE, GOPTIONS, LEGEND, PATTERN, SYMBOL, TITLE

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, "Plotting a Cloud Analytic Services (CAS) In-Memory Table," on page 153.

The procedure can include the SAS/GRAPH statement BY on page 424, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See "SAS/GRAPH Statements" on page 397 and *SAS DATA Step Statements: Reference* for more information.

```
PROC GCONTOUR <DATA=input-data-set>
<ANNOTATE=Annotate-data-set>
<GOUT=<libref.>output-catalog>
<INCOMPLETE>;
   PLOT y*x=z </options>;
```

PROC GCONTOUR Statement

Identifies the data set that contains the plot variables. Can also specify an annotate data set, an output catalog, and the incomplete option.

Requirement: An input data set is required.

Syntax

PROC GCONTOUR <DATA= *input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<INCOMPLETE>;

Optional Arguments

PROC GCONTOUR statement options affect all graphs produced by the procedure.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs produced by the GCONTOUR procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

Restriction Partially supported by Java and ActiveX

See Chapter 28, “Using Annotate Data Sets,” on page 721

DATA=*input-data-set*
specifies the SAS data set that contains the variables to plot. The procedure uses the most recently created SAS data set if none is specified.

See “The SAS Data Set: Your Key to the SAS System” in *Step-by-Step Programming with Base SAS*

“About Data Set Options” in *SAS Data Set Options: Reference*

“Input Data” on page 1127

GOUT=<*libref.*>*output-catalog*
specifies the SAS catalog in which to save the graphics output that is produced by the GCONTOUR procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary library called WORK and creates the output catalog if it does not exist.

INCOMPLETE
allows the plotting of data when values are missing for more than half of the *z* variable in the input data set.

Restriction Not supported by Java and ActiveX

PLOT Statement

Creates contour plots using the values of three numeric variables from the input data set as the source of the contour coordinates.

Requirement: A plot request is required.

Supported Statements: AXIS, FOOTNOTE, GOPTIONS, LEGEND, NOTE, PATTERN, SYMBOL, TITLE

Syntax

PLOT *y*x=z* <*/options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE= *Annotate-data-set*
: specifies an Annotate data set to enhance the charts produced by the PLOT statement.

CAXIS=*axis-color*
: specifies a color for axis lines, axis tick marks, and the frame around the plot.

CFRAME=*background-color*
: fills the axis area with the specified color.

COUTLINE=*outline-color*
: specifies a color for outlining filled areas.

CTEXT=*text-color*
: specifies a color for the axis labels, axis tick mark values, legend labels, and legend value descriptions.

GRID
: draws reference lines at all major tick marks on both axes.

LEGEND=LEGEND<1...99>
: assigns legend characteristics from the corresponding legend definition to the plot's legend.

NOAXIS
: specifies that a plot have no axis values, axis labels, or axis tick marks.

NOFRAME
: suppresses the frame that is drawn around the plot area.

NOLEGEND
: suppresses the legend that describes the plot by displaying the *z* variable name or label, the legend values, and legend value descriptions.

Catalog entry description options

DESCRIPTION="*description*"
: specifies a description of the output.

NAME="*name*"
: specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Contour options

CLEVELS=*color(s)*
: specifies a color or list of colors for the contour levels.

JOIN
: combines adjacent grid cells with the same pattern to form a single pattern area.

LEVELS=*value-list*
: specifies up to 100 values for the *z* variable.

LJOIN
: displays filled contour areas with contour lines.

LLEVELS=*line-type-list*
: lists line types for plot contour lines.

NLEVELS=*number-of-levels*
: specifies the number of contour levels to plot.

PATTERN
: specifies that the plot contour levels are represented by rectangles filled with patterns.

SMOOTH
: produces smooth gradient areas between levels.

Horizontal axis options

AUTOHREF
: displays reference lines originating at the major tick marks on the horizontal axis.

CHREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color list*)
: specifies a color or colors for reference lines drawn with the HREF= option and the AUTOHREF option.

HAXIS=AXIS<1...99>
: assigns axis characteristics from the corresponding axis definition to the horizontal *x* axis.

HMINOR=*number-of-minor-tick marks*
: specifies the number of minor tick marks to draw between each major tick mark on the horizontal *x* axis.

HREF=*value* | (*value*) | (*value-list*)
: displays up to 100 reference lines originating on the horizontal *x* axis at specified values within the *x* axis range.

HREVERSE
: specifies that the order of the values on the horizontal *x* axis be reversed.

LAUTOHREF=*reference-line-type*
: specifies a line type for reference lines specified by the AUTOHREF option.

LHREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type list*)
: specifies line types for reference lines originating on the horizontal axis.

WAUTOHREF=*reference-line-width*
: specifies a line width for reference lines specified by the AUTOHREF option.

WHREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
: specifies a line width for reference lines specified by the HREF= option.

XTICKNUM=*number-of-major-tick-marks*
: specifies the number of major tick marks on the horizontal *x* axis.

Labeling option

AUTOLABEL | AUTOLABEL=*(autolabel_suboptions)*
: labels the contour lines.

Vertical axis options

AUTOVREF
: displays reference lines originating at the major tick marks on the vertical axis.

CAUTOHREF=*reference-line-color*
: specifies a color for all the reference lines displayed by the AUTOHREF option.

CAUTOVREF=*reference-line-color*
: specifies a color for all the reference lines displayed by the AUTOVREF option.

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color list*)
: specifies a color or colors for reference lines drawn with the VREF= option and the AUTOVREF option.

LAUTOVREF=*reference-line-type*
: specifies a line type for reference lines specified by the AUTOVREF option.

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
: specifies line types for reference lines originating on the vertical axis.

VAXIS=AXIS<1...99>
: assigns axis characteristics from the corresponding axis definition to the vertical *y* axis.

VMINOR=*number-of-minor-tick marks*
: specifies the number of minor tick marks located between each major tick mark on the vertical *y* axis.

VREF=*value* | (*value*) | (*value-list*)
: displays up to 100 reference lines originating on the vertical *y* axis at specified values within the *y* axis range.

VREVERSE
: specifies that the order of the values on the vertical axis be reversed.

WAUTOVREF=*reference-line-width*
: specifies a line width for reference lines specified by the AUTOVREF option.

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
: specifies a line width for reference lines specified by the VREF= option.

YTICKNUM= *number-of-major-tick-marks*
: specifies the number of major tick marks on the vertical *y* axis.

Required Argument

y*x=z

specifies three numeric variables from the input data set:

y
: is the variable that is plotted on the vertical axis.

x
: is the variable that is plotted on the horizontal axis.

z
: is the variable that is plotted as contour lines.

Optional Arguments

Options in a PLOT statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order. If you use a BY statement on the procedure, the options in each PLOT statement affect all graphs produced by that BY statement.

ANNOTATE= ***Annotate-data-set***

specifies an Annotate data set to enhance the charts produced by the PLOT statement.

Alias ANNO=

Restriction Partially supported by Java and ActiveX

See Chapter 33, “GANNO Procedure,” on page 887

“Annotate Function Dictionary” on page 759

“Annotate Variable Dictionary” on page 803

“Annotate Macro Dictionary” on page 853

AUTOHREF

displays reference lines originating at the major tick marks on the horizontal axis.

Restriction Not supported by Java

AUTOLABEL | AUTOLABEL=*(autolabel_suboptions)*

labels the contour lines. Autolabel suboptions enable you to control the appearance of these labels.

The label for each contour line is the value of the *z* variable for that contour level. The labels are displayed in BEST format. The FORMAT statement enables you to change the display format.

You can also use the SYMBOL statement to control the appearance and text of contour labels and lines.

When AUTOLABEL is used with the LLEVELS= option, LLEVELS is ignored.

When AUTOLABEL is used with the CLEVELS= option, AUTOLABEL is ignored.

Restriction Not supported by Java and ActiveX

See “AUTOLABEL= Suboptions” on page 1143

Example “Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1152

AUTOVREF

displays reference lines originating at the major tick marks on the vertical axis.

Restriction Not supported by Java

CAUTOHREF=*reference-line-color*

specifies a color for all the reference lines displayed by the AUTOHREF option. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Restriction Not supported by Java

CAUTOVREF=*reference-line-color*
specifies a color for all the reference lines displayed by the AUTOVREF option. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified

Restriction Not supported by Java

CAXIS=*axis-color*
specifies a color for axis lines, axis tick marks, and the frame around the plot. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Restriction Partially supported by Java

CFRAME=*background-color*
fills the axis area with the specified color. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Alias CFR=

CHREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color list*)
specifies a color or colors for reference lines drawn with the HREF= option and the AUTOHREF option. Colors are specified as follows:

- specifying a single color without parentheses applies that color to all reference lines drawn with the HREF= option and the AUTOHREF option
- specifying a single color within parentheses applies the color to the first reference line drawn with the HREF= option only
- specifying a list of colors within parentheses applies the colors sequentially to the reference lines drawn with the HREF= option only

The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Alias CH=

Restriction Not supported by Java

CLEVELS=*color(s)*
specifies a color or list of colors for the contour levels. GCONTOUR substitutes user-defined colors in the ODS style. If more colors are needed, GCONTOUR uses the next color in the ODS style until all lines have an associated color. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Restriction Not supported by Java and partially supported by ActiveX

COUTLINE=*outline-color*
specifies a color for outlining filled areas. This option is ignored unless the PATTERN option is also used. COUTLINE=SAME creates a plot with outlines that are the same color as the adjacent fill color.

Restriction Not supported by Java and ActiveX

Note The outline color is the only distinction between empty patterns. Use of this option makes the patterns look the same when VALUE=EMPTY in PATTERN definitions.

Example "Example 4: Using Patterns and Joins" on page 1157

CTEXT=*text-color*

specifies a color for the axis labels, axis tick mark values, legend labels, and legend value descriptions. GCONTOUR uses the first color that it finds in the following list:

- colors specified for labels and values on assigned AXIS and LEGEND statements.
- the color specified by the CTEXT= option in the PLOT statement.
- the color specified by the CTEXT= option in a GOPTIONS statement.
- the color specified in the current style, or if the NOGSTYLE system option is specified, the default color is the first color in the color list for each device.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification. In this case CTEXT= is used for axis labels and values. Meanwhile, the COLOR= color is still used for all other axis colors, such as tick marks.

Note If you use a BY statement in the procedure, the color of the BY variables' labels is controlled by the CBY= option in the GOPTIONS statement.

Example "Example 4: Using Patterns and Joins" on page 1157

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color list*)

specifies a color or colors for reference lines drawn with the VREF= option and the AUTOVREF option. Colors are specified as follows:

- specifying a single color without parentheses applies that color to all reference lines drawn with the VREF= option and the AUTOVREF option
- specifying a single color within parentheses applies the color to the first reference line drawn with the VREF= option only
- specifying a list of colors within parentheses applies the colors sequentially to the reference lines drawn with the VREF= option only

Alias CV=

Restriction Not supported by Java

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open
- the description and the properties for the output in the Results window
- the description and properties for the catalog entry in the SAS Explorer
- the **Description** field of the PROC GREPLAY window

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default PLOT OF *y***x*=*z*, where *y***x*=*z* is the request specified in the PLOT statement.

GRID

draws reference lines at all major tick marks on both axes. This option produces the same result as would be achieved with specifying both the AUTOHREF and AUTOVREF options.

Restriction Not supported by Java

HAXIS=AXIS<1...99>

assigns axis characteristics from the corresponding axis definition to the horizontal *x* axis. If the AXIS statement specifies the REFLABEL= option, labels are applied in sequence to all reference lines generated with the HREF= option.

Restriction Partially supported by Java and ActiveX

See "AXIS Statement" on page 399

Example "Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend" on page 1152

HMINOR=*number-of-minor-tick marks*

specifies the number of minor tick marks to draw between each major tick mark on the horizontal *x* axis.

Alias HM=

Interaction The HMINOR= option overrides the MINOR= option in an AXIS definition assigned to the horizontal axis

Example "Example 3: Specifying Contour Levels" on page 1155

HREF=*value* | (*value*) | (*value-list*)

displays up to 100 reference lines originating on the horizontal *x* axis at specified values within the *x* axis range. Any values specified beyond the axis range are not drawn, and a warning is issued to the log. To specify labels for this option, use the HAXIS= option. The *value-list* can be an explicit list of values, a starting value and an ending value with an interval increment, or a combination of both forms:

- *n* <... *n* >.
- *n* TO *n* <BY *increment* >.
- *n* <... *n* > TO *n* <BY *increment* > *n* <... *n* >.

Restriction Not supported by Java

HREVERSE
specifies that the order of the values on the horizontal *x* axis be reversed.

Restriction Not supported by Java

JOIN
combines adjacent grid cells with the same pattern to form a single pattern area.

Restriction This option is ignored unless the PATTERN option is also used.

Note Java and ActiveX support the JOIN option without the pattern option.

LAUTOHREF=*reference-line-type*
specifies a line type for reference lines specified by the AUTOHREF option. The *reference-line-type* value is any integer from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines.

Default 1 (solid)

Restriction Not supported by Java

See "Specifying Line Types" on page 502 for available line types

LAUTOVREF=*reference-line-type*
specifies a line type for reference lines specified by the AUTOVREF option. The *reference-line-type* value is any integer from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines.

Restriction Not supported by Java

See "Specifying Line Types" on page 502 for available line types

LEGEND=LEGEND<1...99>
assigns legend characteristics from the corresponding legend definition to the plot's legend. To suppress the legend, use the NOLEGEND option. The LEGEND= option is ignored if the specified LEGEND definition is not currently in effect.

Restriction Partially supported by Java (always displayed on the right side of plot) and ActiveX

Interaction If you use the SHAPE= option in a LEGEND statement, the value LINE is valid. If you use the PATTERN option, SHAPE=BAR is also valid.

See "LEGEND Statement" on page 432

Example "Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend" on page 1152

LEVELS=*value-list*

specifies up to 100 values for the *z* variable. Because GCONTOUR uses the *z* variable to calculate plot contour levels, you can use the LEVELS= option to change the number of contour levels plotted.

The *value-list* can be an explicit list of values, a starting value and an ending value with an interval increment, or a combination of both forms:

- *n* <... *n* >.
- *n* TO *n* <BY *increment* >.
- *n* <... *n* > TO *n* <BY *increment* > *n* <... *n* >.

If a variable has an associated format, the specified values must be the unformatted values.

The contour lines on the plot represent the intersection of a plane formed by the *(x,y)* variables, and the surface that is formed by the values of the *z* variable.

LHREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type list*)

specifies line types for reference lines originating on the horizontal axis. The *reference-line-type* value is any integer from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. When using this option, the following is true:

- specifying a single line type without parentheses applies that line type to all reference lines drawn with the HREF= option and the AUTOHREF option
- specifying a single line type within parentheses applies the line type to the first reference line drawn with the HREF= option only
- specifying a list of line types within parentheses applies the line types sequentially to the reference lines drawn with the HREF= option only
- the LAUTOHREF= option overrides the LHREF= option for lines drawn with the AUTOHREF option

Alias LH=

Restriction Not supported by Java and partially supported by ActiveX

See "Specifying Line Types" on page 502 for available line types

LJOIN

displays filled contour areas with contour lines.

Restriction Supported by Java and ActiveX only

LLEVELS=*line-type-list*

lists line types for plot contour lines. Each line type represents one contour level. If fewer line types are specified than the number of levels in the plot, GCONTOUR provides additional line types. Valid values for *line-type-list* are integers from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines.

Default 1 (solid)

Restriction Not supported by Java and partially supported by ActiveX

See "Specifying Line Types" on page 502 for the line types represented by each number.

Example "Example 3: Specifying Contour Levels" on page 1155

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines originating on the vertical axis. Valid values for *line-type-list* are integers from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. When using this option, the following is true:

- specifying a single line type without parentheses applies that line type to all reference lines drawn with the VREF= option and the AUTOVREF option
- specifying a single line type within parentheses applies the line type to the first reference line drawn with the VREF= option only
- specifying a list of line types within parentheses applies the line types sequentially to the reference lines drawn with the VREF= option only
- the LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option

Alias LV=

Restriction Partially supported by Java and ActiveX

See "Specifying Line Types" on page 502 for the line types represented by each number

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.

- If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

  Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GCONTOUR

Example "Example 3: Specifying Contour Levels" on page 1155

NLEVELS=*number-of-levels*

specifies the number of contour levels to plot.

Valid values are integers from 1 to 100.

There are 2 algorithms used to compute the contour levels, where L represents an array of levels.

If the value of NLEVELS= is less than 7, then the algorithm is as follows:

```
D = (Zmax - Zmin) / NLEVELS
d= 0.5 * D
L[0] = Zmin + d, L[i] = L[i-1] + D
```

In this algorithm, each level is the midpoint of a number of ranges equal to the value of the NLEVELS= option. These ranges exactly cover the range of the *z* variable.

If the value of NLEVELS= is greater than or equal to 7, then the algorithm is as follows:

```
e= 0.05 * (100 - NLEVELS) / 93
d = (Zmax - Zmin) * e
D = ((Zmax - Zmin - 2*d) / (NLEVELS - 1)
L[0] = Zmin + d, L[i] = L[i-1] + D
```

Default 20 for Java, 6 for ActiveX, 7 for all other devices

Restriction Partially supported by Java and ActiveX

Example "Example 3: Specifying Contour Levels" on page 1155

NOAXIS

specifies that a plot have no axis values, axis labels, or axis tick marks. The frame is displayed around the plot unless you use the NOFRAME option.

Alias NOAXES

Restriction Partially supported by Java

NOFRAME

suppresses the frame that is drawn around the plot area.

Restriction Not supported by Java

NOLEGEND
suppresses the legend that describes the plot by displaying the *z* variable name or label, the legend values, and legend value descriptions.

Default LEGEND

PATTERN
specifies that the plot contour levels are represented by rectangles filled with patterns. The pattern for each rectangle is determined by calculating the mean of the values of the *z* variable for the four corners of the rectangle. The pattern is then assigned for the level closest to the mean.

To explicitly define patterns, use PATTERN definitions for map or plot patterns.

See "PATTERN Statement" on page 455

Example "Example 4: Using Patterns and Joins" on page 1157

SMOOTH
produces smooth gradient areas between levels.

Restriction Supported by Java and ActiveX only

VAXIS=AXIS<1...99>
assigns axis characteristics from the corresponding axis definition to the vertical *y* axis. If the AXIS statement specifies the REFLABEL= option, labels are applied in sequence to all reference lines generated with the VREF= option.

Restriction Partially supported by Java and ActiveX

See "AXIS Statement" on page 399

Example "Example 3: Specifying Contour Levels" on page 1155

VMINOR=*number-of-minor-tick marks*
specifies the number of minor tick marks located between each major tick mark on the vertical *y* axis. Value labels are not displayed for minor tick marks.

Alias VM=

Interaction The VMINOR= option overrides the MINOR= option in an AXIS definition that is assigned to the vertical axis.

Example "Example 3: Specifying Contour Levels" on page 1155

VREF=*value* | (*value*) | (*value-list*)
displays up to 100 reference lines originating on the vertical *y* axis at specified values within the *y* axis range. Any values specified beyond the axis range are not drawn, and a warning is issued to the log. To specify labels for these reference lines, use the VAXIS= option. The *value-list* can be an explicit list of values, a starting value and an ending value with an interval increment, or a combination of both forms:

- *n* <... *n* >.
- *n* TO *n* <BY *increment* >.
- *n* <... *n* > TO *n* <BY *increment* > *n* <... *n* >.

Restriction Not supported by Java

VREVERSE
specifies that the order of the values on the vertical axis be reversed.

Restriction Not supported by Java

WAUTOHREF=*reference-line-width*
specifies a line width for reference lines specified by the AUTOHREF option. The *reference-line-width* can be any number greater than zero, and does not need to be an integer. If you specify a value that is less than zero, an error message is written to the SAS log.

Default Current style setting, 1 if NOGSTYLE

Restriction Not supported by Java and ActiveX

WAUTOVREF=*reference-line-width*
specifies a line width for reference lines specified by the AUTOVREF option. The *reference-line-width* can be any number greater than zero, and does not need to be an integer. If you specify a value that is less than zero, an error message is written to the SAS log.

Default Current style setting, 1 if NOGSTYLE

Restriction Not supported by Java and ActiveX

WHREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies a line width for reference lines specified by the HREF= option. The *reference-line-width* can be any integer that is greater than zero. If you specify a value that is less than zero or is not an integer value, an error message is written to the SAS log. Specifying a line width without parentheses applies that type to all reference lines drawn with the WAUTOHREF and HREF= options. Note that the WAUTOHREF= option overrides WHREF=*reference-line-width* for reference lines drawn with the WAUTOHREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the HREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the HREF= option. The syntax of the line-width list requires parentheses and the line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.). To specify colors for these reference lines, use the CHREF= option.

Default Current style setting, 1 if NOGSTYLE

Restriction Not supported by Java and ActiveX

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies a line width for reference lines specified by the VREF= option. The *reference-line-width* can be any integer that is greater than zero. If you specify a value that is less than zero or is not an integer value, an error message is written to the SAS log. Specifying a line width without parentheses applies that type to all reference lines drawn with the WAUTOVREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=*reference-line-width* for reference lines drawn with the WAUTOVREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the

VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the HVEF= option. The syntax of the line-width list requires parentheses and the line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.). To specify colors for these reference lines, use the CVREF= option.

Default Current style setting, 1 if NOGSTYLE

Restriction Not supported by Java and ActiveX

XTICKNUM=*number-of-major-tick-marks*
specifies the number of major tick marks on the horizontal *x* axis. The value of *number-of-major-tick–marks* must be greater than or equal to 2. The MAJOR= and ORDER= options in an AXIS definition that is assigned to the horizontal *x* axis overrides the XTICKNUM= option.

YTICKNUM= *number-of-major-tick-marks*
specifies the number of major tick marks on the vertical *y* axis. The value of *number-of-major-tick-marks* must be greater than or equal to 2. The MAJOR= and ORDER= options in an AXIS definition that is assigned to the vertical *y* axis overrides the YTICKNUM= option.

AUTOLABEL= Suboptions

The AUTOLABEL= option accepts the following autolabel suboptions:

CHECK=*checking-factor* | NONE
specifies a collision checking factor that controls collisions between contour label text and other contour lines or other labels. Values can be integers from 0 to 100, inclusive, where 0 provides minimal collision checking and 100 provides maximal collision checking. Fractional values are permitted.

CHECK=NONE suppresses contour label collision checking and can lessen the time needed to compute the contour plot.

Default 75

MAXHIDE=*amount* <*units*>
specifies the maximum amount of the contour line that can be hidden by contour labels. The value of *amount* must be greater than zero.

Valid *units* are CELLS (horizontal character cell positions), CM (centimeters), IN (inches), or PCT (percentage of the width of the graphics output area). If you omit *units*, a unit specification is searched for in this order:

1. the GUNIT= option in a GOPTIONS statement
2. the default unit, CELLS

For *units* that you specify as PCT or CELLS, the MAXHIDE= suboption calculates the amount of contour line that can be hidden based on the width of the graphics output area. For example, suppose you specify MAXHIDE=50 PCT and the graphics output area is 9 inches wide. Then the maximum amount of the contour line that can be hidden by labels is 4.5 inches.

This option maintains data integrity. It provides a check for overly small increments in the STEP= option in the SYMBOL statement. Moreover, the MAXHIDE= option can prevent small contours from being significantly hidden even when the value of the STEP= option is sufficiently large.

Default MAXHIDE=100

REVEAL
specifies that the contour lines are visible through the label text as dashed lines. Line style 33 is used. This option provides a simple way to see all portions of labeled contours, and can be used to inspect the label positions with respect to the contour lines. The REVEAL option is primarily used for debugging. Occasionally, single-character contour labels can be placed off center from the clipped portion of the contour line when the contour line is irregular or jagged.

TOLANGLE=*angle*
specifies the maximum angle (the tolerance angle) between any two adjacent characters of a contour label. The value of *angle* must be between 1 and 85 degrees. To force contour labels to fall on very smooth sections, specify a small tolerance angle.

Default 30

Details

Description

The PLOT statement specifies the three variables to plot. It can also control the contour levels, label the plot lines, and modify the axes as well as the general appearance of the graph. Only one plot request can be specified in a PLOT statement.

To specify multiple plots for a single PROC GCONTOUR statement, use multiple PLOT statements. If a global statement is specified more than once, the last occurrence is applied to all PLOT statements in that PROC step.

The PLOT statement does the following actions:

- plots the contour lines as levels using the values of the *z* variable
- scales the axes to include the minimum data values and the maximum values of *x* and *y*
- labels the *x* and *y* axes
- generates a labeled legend representing the values of the *z* variable's contour levels

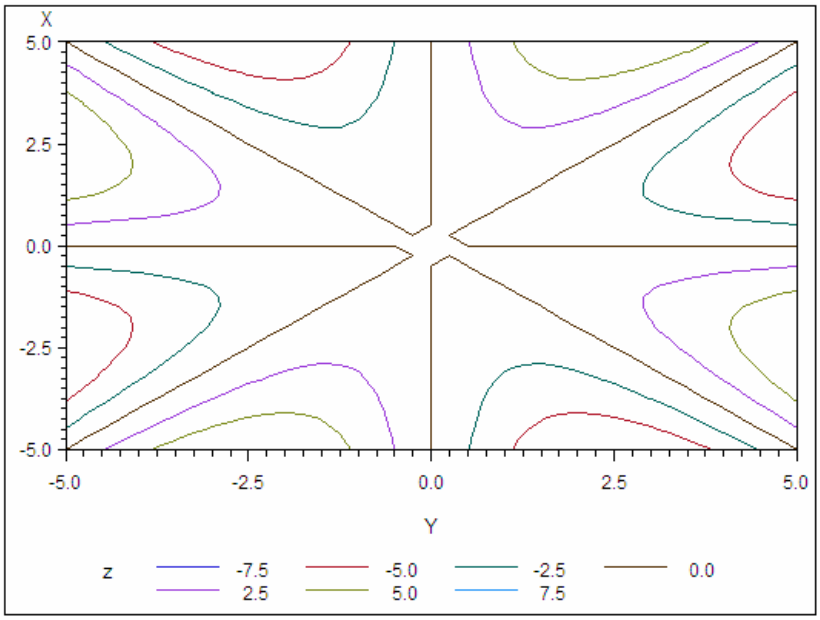
Global statements enable you to modify the axes, the legend, the contour lines and contour line labels, the fill patterns and pattern colors for contour areas. You can also add titles, footnotes, and notes to the plot. You can use an Annotate data set and set GOPTIONS to enhance the appearance of the plot. Moreover, you can filter your data with a WHERE clause. You can format your data values and add labels to the variables. You can generate multiple plots with a BY statement or multiple plot statements.

Selecting Contour Levels

The LEVEL= option enables you to customize your plot, by specifying values for the contour levels.

The LEVELS= option represents the *z* variable values as a third dimension, using uniquely colored contour lines.

Figure 37.3 *Selecting Contour Levels*

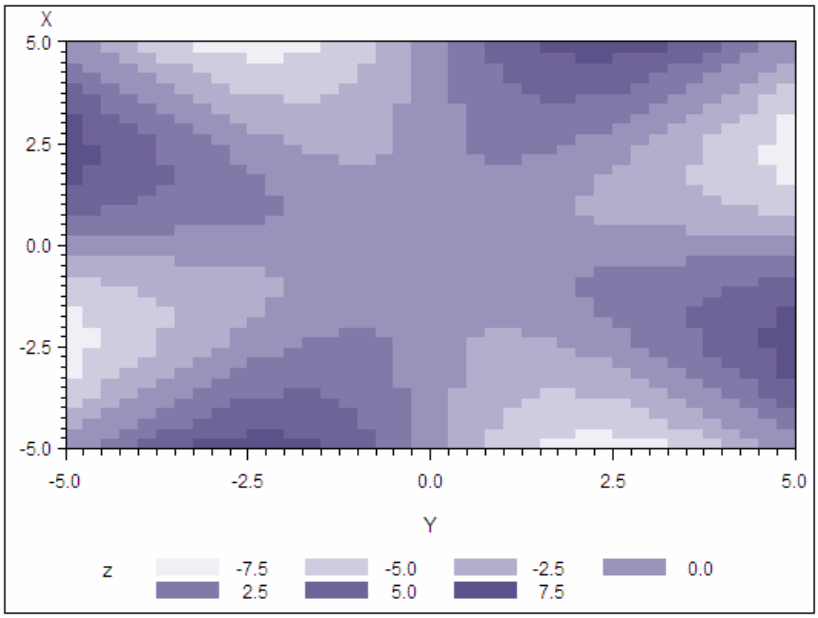


Using the PATTERN option with the LEVELS= option generates a plot with contour levels that are displayed as solid filled rectangles. The rectangles are formed by points in the *(x, y)* grid. The contour pattern of a rectangle, or grid cell, is determined by average value of the *z* variable for the four corners of the rectangle. The grid cell is assigned the pattern for the level closest to the calculated mean. For example, suppose you specified contour levels of 0, 5, and 10, and the plot contains a grid cell with a mean of 100. The grid cell is assigned the pattern for the nearest level: 10. A grid cell with a mean of 7.6 is also assigned the pattern for the 10 level. The same data used with the following PLOT statement in the GCONTOUR procedure produces a similar contour plot:

```
plot y*x=z / levels=-7.5 to 7.5 by 2.5/
      pattern;
run;
```

The following contour plot with the PATTERN option uses the same data and contour levels as “Example 4: Using Patterns and Joins” on page 1157. Contour plots using the same contour levels can present your data differently, if one plot uses a pattern and the other does not. However, the contour pattern boundaries in the following contour plot do not correspond to the contour lines shown in the example.

Figure 37.4 Contour Plot with the PATTERN Option

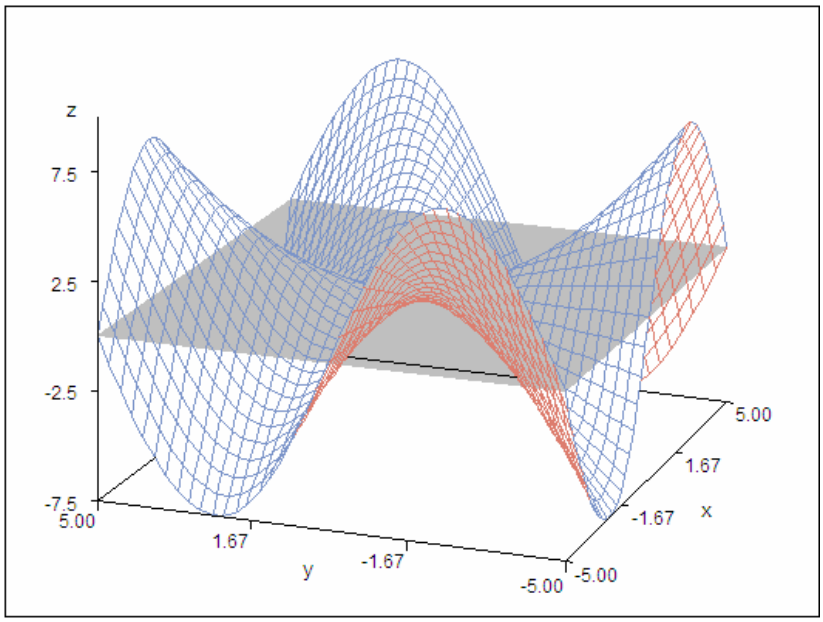


Using the data to create a surface plot with the G3D procedure, the contour lines in a GCONTOUR procedure plot represent the intersection of the plane and the surface.

For example, suppose that you use the G3D procedure, and your data produces a surface plot like the one shown below.

The contour lines represent the intersection of the surface lines with the plane parallel to the plane formed by the variables *x* and *y*. The intersections are located at the *z* values of –7.5, –5.0, –2.5, and so on.

Figure 37.5 G3D Surface Plot



Specifying Axis Order

You can use AXIS statements to modify the text and appearance of plot axes. You can then assign the axes to the contour plot with the PLOT statement's HAXIS= and VAXIS= options. If the AXIS statement uses an ORDER= option, there are special considerations for using that AXIS definition with the GCONTOUR procedure.

A list of variable values that are specified with the AXIS statement's ORDER= option must contain numbers listed in ascending or descending order. These numbers are treated as a continuous data range for an axis. Thus, for a contour line or pattern to span the entire specified range, it is not necessary for the maximum and minimum values of the list to match exactly with the maximum and minimum data values of the corresponding *x* or *y* variable. For example, suppose that you assign this AXIS definition to the horizontal (*x*) axis:

```
axis1 order=-2.5 to 2.5 by .5
```

Suppose also that the horizontal axis variable has these values: –5, –4, –3, –2, –1, 0, 1, 2, 3, 4, 5. Depending on the data, contours could extend through the full range of the ORDER= list rather than from –2 to 2. –2 to 2 are the actual values of the variable assigned to the horizontal (*x*) axis. In this case, values are interpolated for the *x* variable at any point where the *y* variable intersects the minimum axis value (–2.5) or the maximum axis value (2.5). Data values that are outside of the axis range (in this case, –5, –4, –3, 3, 4, and 5) are clipped from the plot.

When ORDER= lists cause data clipping, internal plotting grids are modified according to these rules:

- If an ORDER= list causes data clipping on a single axis, linear interpolation generates the *z* values of the starting or ending column, or both columns of the plotting grid. For example, in the previous example, the value of *z* is interpolated for –2.5 and 2.5 on the horizontal (*x*) axis.
- If ORDER= lists cause data clipping on both axes, the response variable values of the new corners are derived by fitting the new *x*, *y* location on a plane. This plane is formed by three of the original four points of the corresponding grid square.
- When assigning the following AXIS definition to a plot of the same data, the contour levels on the plot do not extend beyond the range of the data:

  ```
  axis1 order=-10 to 10 by 1;
  ```

Figure 37.6 *The ORDER= Option, Values Match the Range of the Data Values*

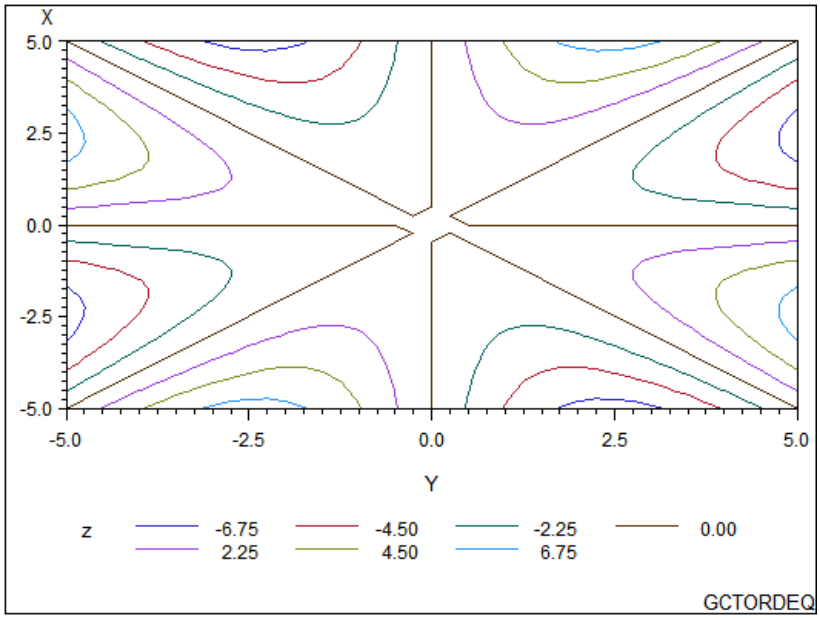


Figure 37.7 *The ORDER= Option, Values Clip the Range of Data Values*

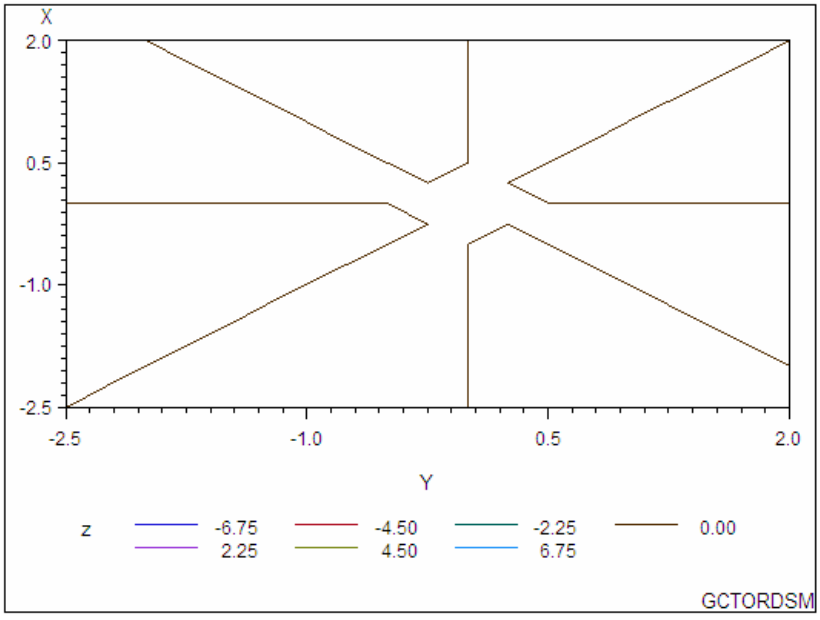
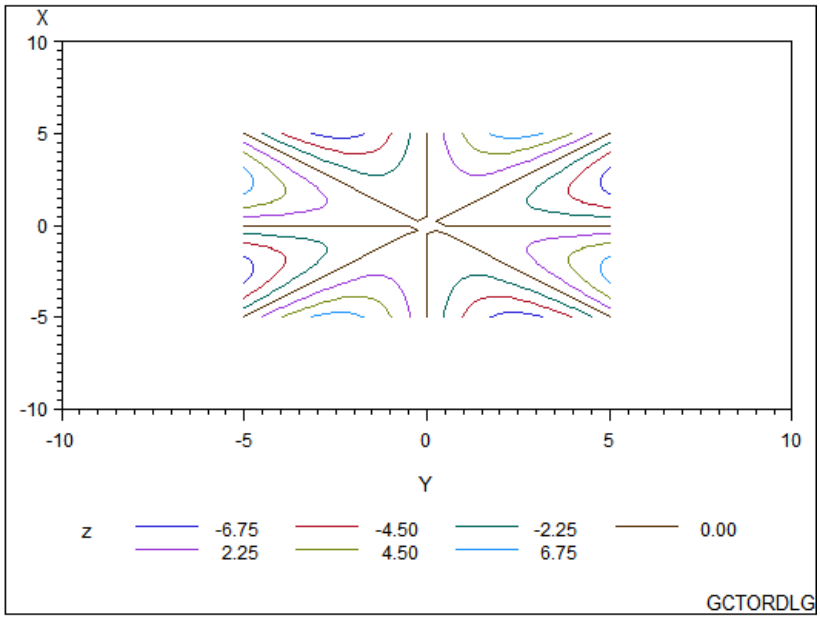


Figure 37.8 *The ORDER= Option, Values Extend beyond the Range of the Data Values*



Modifying Contour Lines and Labels with the SYMBOL Statement

Controlling Line and Label Attributes

When you use the AUTOLABEL option, the LLEVELS= and CLEVELS= options are ignored, and contour-line and label attributes are controlled by the SYMBOL statement. Defaults are used if not enough SYMBOL statements are specified to match the number of contour levels.

If a SYMBOL statement does not include a color option, the SYMBOL statement can be applied to more than one contour level. In this case, the SYMBOL statement is used once with every color in the color list and generates more than one SYMBOL definition. See the "SYMBOL Statement" on page 470 for details.

Table 37.85 on page 1149 describes how SYMBOL statement options affect contour plot lines and labels.

Table 37.1 *The Effect of SYMBOL Statement Options on Contour Lines and Labels*

SYMBOL Statement Option	Contour Line or Label Element Affected
LINE=*line-type*	Contour line style
WIDTH=*n*	Contour line thickness
CI=*line-color* or COLOR=*color*	Contour line color
FONT=*font*	Contour label font

SYMBOL Statement Option	Contour Line or Label Element Affected
HEIGHT=*height*	Contour label height
CV=*color* or COLOR=*color*	Contour label color
STEP=*distance*<*units*>	Minimum distance between labels on the same contour line
VALUE='*text*'	Contour label text
VALUE=NONE	Suppresses the contour label text

The SYMBOL statement option INTERPOL= is not supported by the GCONTOUR procedure.

The STEP= option specifies the minimum distance between contour labels. The lower the value, the more labels the procedure uses. A STEP= value of less than 10% is ignored by the GCONTOUR procedure and a value of 10% is substituted.

For more information, see the "SYMBOL Statement" on page 470.

Specifying Text for Contour Labels

To override the default labels that are displayed by the AUTOLABEL option, you can specify label text for one or more contour lines. To do so, use both the FONT= and VALUE= options in the SYMBOL statement that is assigned to the contour level. Default labels are used for contour levels that you do not label.

For example, this SYMBOL1 statement displays the text string `DEEP` in the Arial font on the contour line that it modifies:

```
symbol1 font="Arial Rounded MT Bold" value="DEEP";
```

You must specify both FONT= and VALUE= options or the text is not used. For an example, see "Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend" on page 1152.

Examples: GCONTOUR Procedure

Example 1: Simple Contour Plot

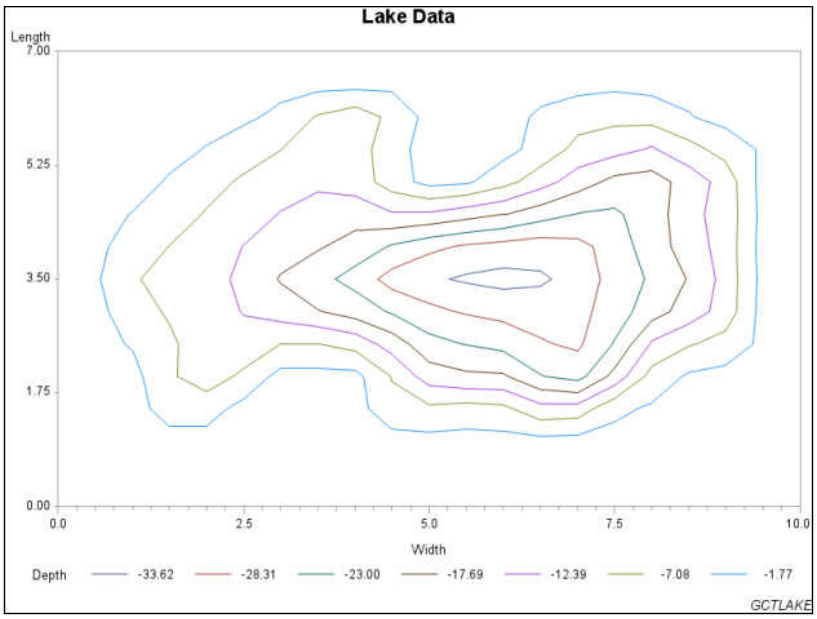
Features: PLOT statement

Data set: SASHELP.LAKE

Sample library member: GCTLAKE

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This simple contour plot displays the various depths of a lake. The dimensions of the lake are plotted on the *x* and *y* axes. The *z* variable is plotted as the third dimension, as levels represented by contour lines. The contour line levels are displayed and labeled in the legend.



Program

```
goptions reset=all border;

title "Lake Data";
footnote j=r "GCTLAKE";

proc gcontour data=sashelp.lake;
    plot length*width=depth;
run;
quit;
```

Program Description

Set the graphics environment. Draw a BORDER around the graphics output area.

```
goptions reset=all border;
```

Define titles and footnotes. Add TITLE content. Add FOOTNOTE content and placement.

```
title "Lake Data";
footnote j=r "GCTLAKE";
```

Generate contour plot. Generate a simple contour plot using SASHELP.LAKE. Use one PLOT statement to define the grid lines and the contour lines.

```
proc gcontour data=sashelp.lake;
    plot length*width=depth;
run;
quit;
```

Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend

Features:

PLOT statement options
- AUTOLABEL=
- HAXIS=
- LEGEND=
- NAME=

AXIS statement options
- MINOR=
- ORDER=

LEGEND statement options
- ACROSS=
- LABEL=
- POSITION=

SYMBOL statement options
- COLOR=
- FONT=
- HEIGHT=
- VALUE=

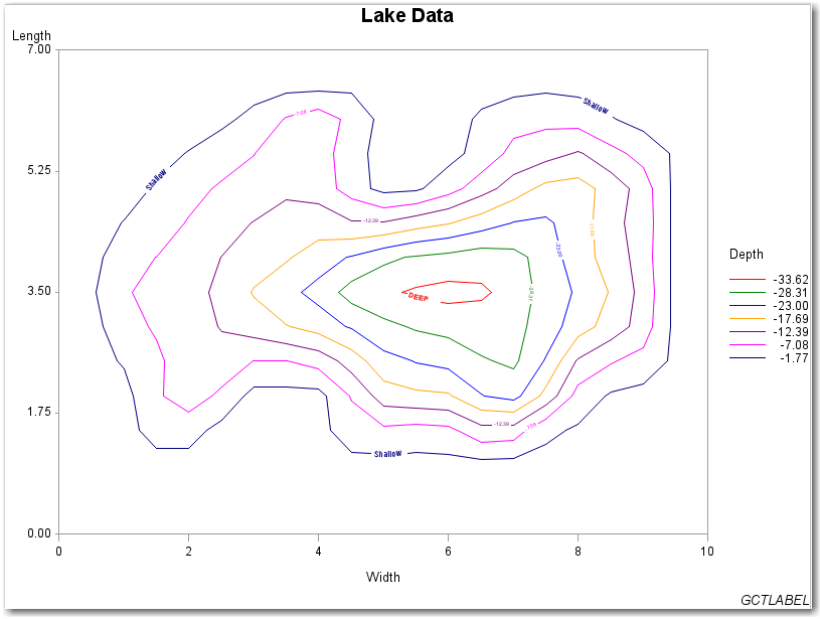
Data set: SASHELP.LAKE

Sample library member: GCTLABEL

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example modifies “Example 1: Simple Contour Plot” on page 1150 to label contour levels with the AUTOLABEL option. The SYMBOL statement used with the AUTOLABEL option enables you to customize the attributes of the contour lines and labels.

In this example, SYMBOL1 and SYMBOL7 assign text labels, text fonts, text height, and text and line color for the first and seventh contour level lines. SYMBOL2–SYMBOL6 define the text height, and the text and line color for contour level lines 2–6. Moreover, the LEGEND statement attributes are modified to reposition the legend closer to the data.



Program

```
goptions reset=all border;

title1 "Lake Data";
footnote1 j=r "GCTLABEL";

symbol1 value="DEEP"
        color=red
        font="Arial Rounded MT Bold"
        height=.6;
symbol2 color=green
        height=.45;
symbol3 color=blue
        height=.45;
symbol4 color=orange
        height=.45;
symbol5 color=purple
        height=.45;
symbol6 color=magenta
        height=.45;
symbol7 value="Shallow"
        color=navy
        font="Arial Rounded MT Bold"
        height=.7;

legend1 position=(right middle)
        label=(position=top)
        across=1;

axis1 order=(0 to 10 by 2) minor=none;

proc gcontour data=sashelp.lake;
   plot length*width=depth/
       autolabel=(check=none)
      haxis=axis1
```

```
        legend=legend1
        name="GCTLABEL";
run;
quit;
```

Program Description

Set the graphics environment. Draw a BORDER around the graphics output area.

```
goptions reset=all border;
```

Define title and footnote. Add TITLE content . Add FOOTNOTE content and placement.

```
title1 "Lake Data";
footnote1 j=r "GCTLABEL";
```

Define symbol statements. SYMBOL statements define the characteristics of the lines and labels for the contour lines. Each SYMBOL statement is associated with an individual contour level starting with the first level displayed in the LEGEND.

```
symbol1 value="DEEP"
        color=red
        font="Arial Rounded MT Bold"
        height=.6;
symbol2 color=green
        height=.45;
symbol3 color=blue
        height=.45;
symbol4 color=orange
        height=.45;
symbol5 color=purple
        height=.45;
symbol6 color=magenta
        height=.45;
symbol7 value="Shallow"
        color=navy
        font="Arial Rounded MT Bold"
        height=.7;
```

Define legend characteristics. The LEGEND statement controls the location and appearance of the legend. The POSITION= option specifies the position of the legend relative to the plot; RIGHT specifies the horizontal position; MIDDLE specifies the vertical position. The LABEL= option modifies the legend label; POSITION=TOP places the legend label relative to the legend entries; the ACROSS= option defines the number of columns in the legend.

```
legend1 position=(right middle)
        label=(position=top)
        across=1;
```

Define axis characteristics. The ORDER= option specifies the order in which the data values appear on the axis; the ORDER= values are displayed as the major tick marks values; MINOR=NONE suppresses all minor tick marks.

```
axis1 order=(0 to 10 by 2) minor=none;
```

Generate the contour plot. The AUTOLABEL= option labels the contour lines; (CHECK=NONE) suppresses contour label collision checking, and might lessen the

time needed to compute the plot. HAXIS=AXIS1 assigns the AXIS1 definition to the horizontal axis. LEGEND=LEGEND1 assigns the LEGEND1 definition to the LEGEND. The NAME= option specifies the name of the catalog entry for the graph.

```
proc gcontour data=sashelp.lake;
   plot length*width=depth/
       autolabel=(check=none)
      haxis=axis1
      legend=legend1
      name="GCTLABEL";
run;
quit;
```

Example 3: Specifying Contour Levels

Features: PLOT statement options
- HMINOR=
- LLEVELS=
- NAME=
- NLEVELS=
- VAXIS=
- VMINOR=

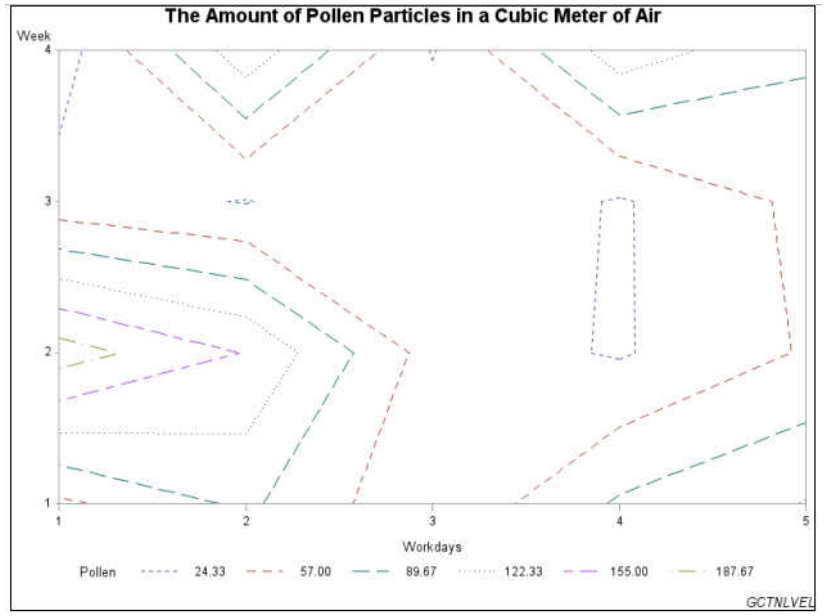
AXIS statement option
- ORDER=

Data set: POLLEN

Sample library member: GCTNLVEL

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This contour plot shows the amount of pollen in the air for five work days (*x*–axis) in a four-week series (*y*-axis). The PLOT statement uses the NLEVELS= option to specify the number of contour levels to plot for the *z* variable. The NLEVELS= option enables you to specify up to 100 levels in your plot.



Program

```
goptions reset=all border;

data pollen;
input Week Workdays Pollen @@;
datalines;
1 1 50 1 2 96 1 3 28 1 4 94 1 5 124 2 1 204 2 2 153 2 3 43 2 4 21 2 5
60
3 1 37 3 2 23 3 3 57 3 4 21 3 5 65 4 1 8 4 2 144 4 3 22 4 4 141 4 5 95
;
run;

title1 "The Amount of Pollen Particles in a Cubic Meter of Air";
footnote1 j=r "GCTNLVEL";

axis1 order=(1 to 4 by 1);

proc gcontour data=pollen;
    plot week*workdays=pollen/
       hminor=0
       llevels= 2 20 21 33 25 41
       name="GCTNLVEL"
       nlevels=6
       vaxis=axis1
       vminor=0;
 run;
 quit;
```

Program Description

Set the graphics environment. Draw a border around the graphics output area.

```
goptions reset=all border;
```

Create data set. Create the data set.

```
data pollen;
input Week Workdays Pollen @@;
datalines;
1 1 50 1 2 96 1 3 28 1 4 94 1 5 124 2 1 204 2 2 153 2 3 43 2 4 21 2 5
60
3 1 37 3 2 23 3 3 57 3 4 21 3 5 65 4 1 8 4 2 144 4 3 22 4 4 141 4 5 95
;
run;
```

Add a title and a footnote. Add TITLE content. Add FOOTNOTE content and placement

```
title1 "The Amount of Pollen Particles in a Cubic Meter of Air";
footnote1 j=r "GCTNLVEL";
```

Define an axis statement for the vertical axis. Define an AXIS statement to order and increment the axis values.

```
axis1 order=(1 to 4 by 1);
```

Generate the contour plot. HMINOR=0 sets the number of minor tick marks on the horizontal axis to 0. The LLEVELS= option lists a line type for each contour line. The number of line types listed correspond to the number of contour levels specified in the NLEVELS= option. NLEVELS=6 specifies the number of levels to compute for *z*. The NAME= option specifies the name of the catalog entry for the plot. The VAXIS= option assigns the AXIS1 statement to the vertical axis. VMINOR=0 sets the number of minor tick marks on the vertical axis to 0.

```
proc gcontour data=pollen;
    plot week*workdays=pollen/
       hminor=0
       llevels= 2 20 21 33 25 41
       name="GCTNLVEL"
       nlevels=6
       vaxis=axis1
       vminor=0;
 run;
 quit;
```

Example 4: Using Patterns and Joins

Features:

PLOT statement options
- COUTLINE=
- CTEXT=
- HAXIS=
- JOIN
- LEGEND=
- PATTERN
- VAXIS=

AXIS statement options
- COLOR=
- LABEL=
- VALUE=

WIDTH=

LEGEND statement

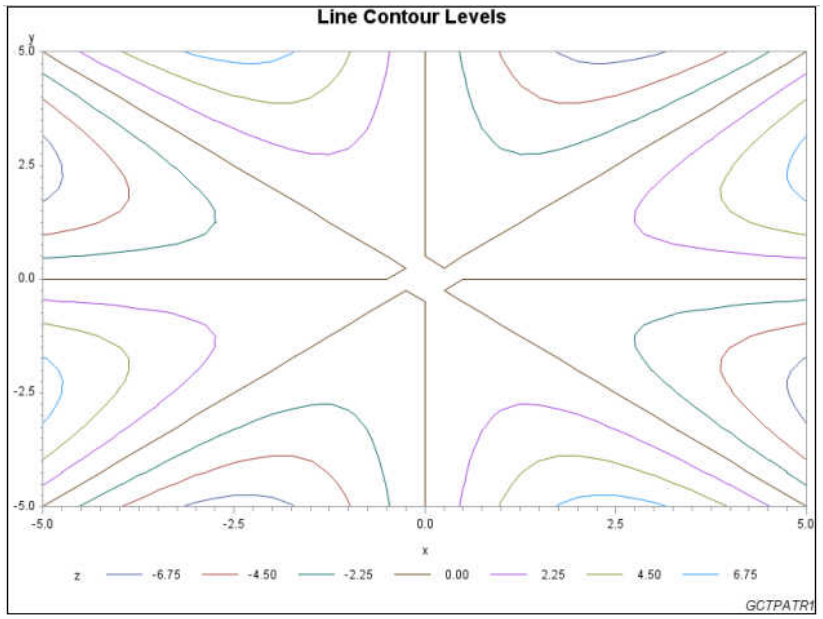
Data set: SWIRL

Sample library member: GCTPATJ

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

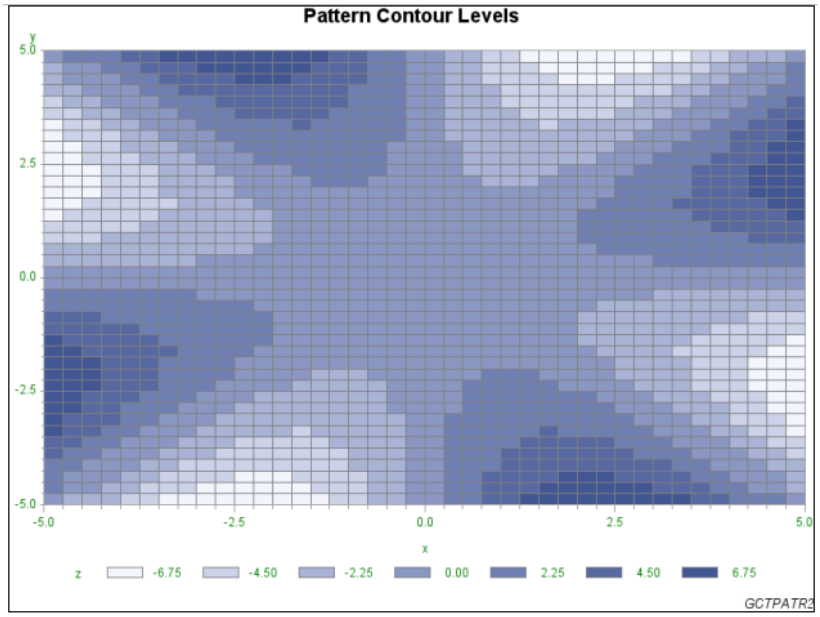
This example demonstrates the differences between using lines and patterns to represent contour levels. The first PLOT statement generates a plot with lines representing contour levels.

Output 37.1 *Line Contour Levels*



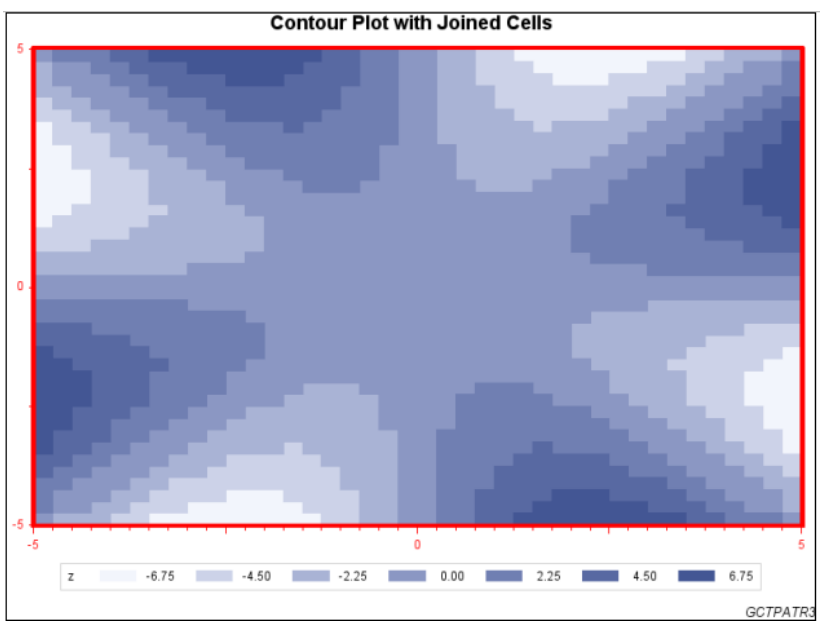
The second PLOT statement specifies the PATTERN option to fill and color contour levels. Additional PLOT statement options outline filled areas in gray and specify green text for all text on the axes and in the legend.

Output 37.2 *Pattern Contour Levels*



The third PLOT statement uses the JOIN option to combine adjacent grid cells with the same pattern to form a single pattern area. Additional options enhance the plot by modifying the axes and framing the legend.

Output 37.3 *Contour Plot with Joined Cells*



Program

```
goptions reset=all border;

data swirl;
    do x= -5 to 5 by 0.25;
```

```
        do y= -5 to 5 by 0.25;
           if x+y=0 then z=0;
              else z=(x*y)*((x*x-y*y)/(x*x+y*y));
           output;
        end;
    end;
run;

title1 "Line Contour Levels";
footnote1 j=r "GCTPATR1";

proc gcontour data=swirl;
   plot y*x=z;
run;
quit;

title1 "Pattern Contour Levels";
footnote j=r "GCTPATR2";

proc gcontour data=swirl;
   plot y*x=z /
      ctext=green
      coutline=gray
      pattern;
run;
quit;

title "Contour Plot with Joined Cells";
footnote j=r "GCTPATR3";

axis1 label=none
    value=("-5" '' "0" '' "5")
    color=red
     width=3;
  axis2 label=none
     value=("-5" '' "0" '' "5")
     color=red
     width=3;


legend1 frame;

proc gcontour data=swirl;
    plot y*x=z /
        haxis=axis1
       join
       legend=legend1
       pattern
       vaxis=axis2;
run;
quit;
```

Program Description

Set the graphics environment. Draw a border around the graphics output area.

```
goptions reset=all border;
```

Create the data set. The data set SWIRL is generated data that produces a symmetric contour pattern, which is useful for illustrating the pattern option.

```
data swirl;
    do x= -5 to 5 by 0.25;
       do y= -5 to 5 by 0.25;
          if x+y=0 then z=0;
             else z=(x*y)*((x*x-y*y)/(x*x+y*y));
          output;
       end;
    end;
run;
```

Define the title and the footnote. Add TITLE content. Add FOOTNOTE content, and placement.

```
title1 "Line Contour Levels";
footnote1 j=r "GCTPATR1";
```

Generate the first contour plot. Generate a simple contour plot.

```
proc gcontour data=swirl;
   plot y*x=z;
run;
quit;
```

Define the title and footnote for the second plot. Add TITLE content for the second plot. Add FOOTNOTE content and placement for the second plot.

```
title1 "Pattern Contour Levels";
footnote j=r "GCTPATR2";
```

Generate the second contour plot. CTEXT=green specifies green for all text on the axes and legend. COUTLINE=gray specifies gray outlining of filled areas. The PATTERN option specifies the fill pattern and colors for the contour levels.

```
proc gcontour data=swirl;
   plot y*x=z /
      ctext=green
      coutline=gray
      pattern;
run;
quit;
```

Define the title and footnote for the third plot. Add TITLE content for the third plot. Add FOOTNOTE content and placement for the third plot.

```
title "Contour Plot with Joined Cells";
footnote j=r "GCTPATR3";
```

Define the axis characteristics. Blanks are used to suppress tick mark labels at positions -2.5 and 2.5.

```
axis1 label=none
    value=("-5" '' "0" '' "5")
    color=red
     width=3;
  axis2 label=none
     value=("-5" '' "0" '' "5")
     color=red
     width=3;
```

Define the legend characteristics. Add a frame around the legend.

```
legend1 frame;
```

Generate the third contour plot. The HAXIS=AXIS1 option assigns an axis definition to the horizontal axis. The JOIN= option combines adjacent grid cells with the same pattern to form a single pattern area. LEGEND=LEGEND1 assigns the legend definition. The PATTERN option specifies the fill pattern and colors for the contour levels. VAXIS=AXIS2 assigns an axis definition to the vertical axis.

```
proc gcontour data=swirl;
    plot y*x=z /
        haxis=axis1
       join
       legend=legend1
       pattern
       vaxis=axis2;
run;
quit;
```

References

Snyder, W.V. 1978. “Contour Plotting [J6] .” *ACM Transactions on Mathematical Software* 4: 290–294.

Chapter 38

GDEVICE Procedure

Overview: GDEVICE Procedure

Overview of the GDEVICE Procedure

The GDEVICE procedure is a tool for examining and changing the parameters of the graphics device driver catalog entries used with SAS/GRAPH software. With the GDEVICE procedure, you can use either the GDEVICE windows or GDEVICE procedure statements to do the following:

- list the device entries stored in any DEVICES catalog
- view the parameters for any device entry
- create and modify new device entries
- copy, modify, rename, or delete existing device entries

See “Using Graphics Devices” on page 89 for a discussion of device drivers and device entries, as well as directions for selecting device drivers, and changing the settings of device parameters.

For a complete list of device entries supplied by SAS that are supported by your operating environment, see the SASHELP.DEVICES catalog.

Concepts: GDEVICE Procedure

Device Catalogs

The DEVICES Catalogs

Device entries are stored in SAS catalogs that are named *libref*.DEVICES. Device entries for your operating environment that SAS supplies with SAS/GRAPH software are stored in the SASHELP.DEVICES.

Custom device entries are typically stored in a catalog named GDEVICE*n*.DEVICES (where *n* can be any number from 0 to 9). However, device entries that have been created or modified by a system administrator specifically for your site also might be stored in SASHELP.DEVICES. (On multi-user systems, the administrator is usually the person who has Write access to the SASHELP.DEVICES catalog.)

The Current Catalog

When the GDEVICE procedure determines which catalog it should use, it searches for the catalog in the following order:

1. the catalog name specified in the CATALOG= option in the PROC GDEVICE statement.
2. the catalog associated with the GDEVICE0 libref, if the libref has been assigned.
3. the catalog SAS supplies, SASHELP.DEVICES. (SASHELP.DEVICES is usually write-protected and is opened in Browse mode.)

The first catalog SAS encounters is the current catalog.

To specify the current catalog by:

- use the CATALOG= option in the PROC GDEVICE statement (this is required to open a driver entry in Update mode)
- assign the GDEVICE0 libref to the appropriate catalog

Search Order of Device Catalogs

SAS/GRAPH searches only librefs starting with GDEVICE0 through GDEVICE9. The libraries must contain a catalog named DEVICES for SAS/GRAPH to search for the device entries for any driver. If you have personal device catalogs in more than one SAS library, you must assign librefs in the sequence GDEVICE0, GDEVICE1, GDEVICE2, and so on.

If the libref GDEVICE0 has been assigned to a SAS library, SAS/GRAPH looks in that library for a catalog named DEVICES. If the GDEVICE0.DEVICES catalog exists, it is checked for the specified device entry. If the device entry is not there, SAS/GRAPH looks next for a library with the libref GDEVICE1 and for a catalog named DEVICES in that library. The search is repeated for the sequence of librefs through GDEVICE9.

The search terminates if either of the following are true:

- Any of the GDEVICE libraries do not contain a DEVICES catalog.
- The librefs do not follow the numeric sequence of GEVICE0, GDEVICE1, GDEVICE2, and so on, in that order.

If SAS/GRAPH terminates the search for either reason, or it does not find the specified device entry, SAS/GRAPH searches the SASHELP.DEVICES catalog. If the specified device entry is not found in the SASHELP.DEVICES catalog, an error message is written to the log.

Note: As stated above, the search for entries terminates if there is a break in the sequence. For example, the catalog GDEVICE1.DEVICES is not checked if the libref GDEVICE0 is undefined or if GDEVICE0 does not contain a catalog named DEVICES.

Syntax: GDEVICE Procedure

Requirement: Statements other than the PROC GDEVICE statement can be used only in a non-windowing or batch environment. In these environments, at least one statement is required to give GDEVICE an action to perform. In a windowing environment, the PROC GDEVICE statement is required. In program mode, at least one statement is required.

Note: You must have Write access to the device catalog in order to modify, add, or delete entries.

PROC GDEVICE <CATALOG=<*libref*.>*SAS-catalog*> <BROWSE> <NOFS>;

ADD *new-device-entry device-parameters* <*device-row-column-parameters*>;

COPY *device-entry copy-options*;
DELETE *device-entry*;
FS;
LIST *device-entry* | _ALL_ | _NEXT_ | _PREV_ | DUMP;
MODIFY *device-entry device-parameters*;
RENAME *device-entry-name* NEWNAME=*new-device-entry-name*;
QUIT ;

PROC GDEVICE Statement

Starts the procedure and determines whether it is running in windowing mode or program mode. Can identify a device catalog and specify how that catalog is opened.

Syntax

PROC GDEVICE <CATALOG=<*libref.*>*SAS-catalog*> <BROWSE> <NOFS>;

Optional Arguments

Options used in the PROC GDEVICE statement affect how you use the procedure. They specify how to open the catalog.

BROWSE
opens a catalog in Browse mode. You cannot modify a catalog when you open it with the BROWSE option. If you are running in program mode when you use BROWSE, you can use only the FS, LIST, QUIT, END, or STOP statements.

CATALOG=<*libref.*>*SAS-catalog*
specifies the catalog containing device information. If you do not specify a catalog, the procedure opens the first catalog found in the search order of catalogs in Browse mode. For search order of source catalogs, see “Search Order of Device Catalogs” on page 1165 .

To edit the device entries in a catalog, you must specify the CATALOG= option.

Aliases CAT=

C=

NOFS
specifies that you are using program mode. In windowing environments, the GDEVICE windows are the default, and you must specify NOFS to start GDEVICE in program mode.

ADD Statement

Adds a new device entry to the catalog selected by the CATALOG= option in the PROC GDEVICE statement. The device entry is initialized with NULL values for most parameters.

Restriction: Not valid in Browse mode.

Requirement: You must have Write access to the device catalog in order to add entries, and use CATALOG= in the PROC GDEVICE statement.

Tip: The best way to add a new driver is to copy an existing driver and modify the parameters. The ADD statement initializes all the parameter values to NULL, and you have to set values for all of the parameters to something other than NULL.

Syntax

ADD *new-device-entry required-device-parameters* <*optional-device-parameters*>;

Required Arguments

new-device-entry
specifies the one-level name of the new device entry. *New-device-entry* must be a valid name for a SAS catalog entry for your operating environment and cannot already exist in the current catalog.

device-parameters
specifies the parameters for the new device. All of the required parameters for the ADD statement correspond to device parameters of the same name. Refer to "Graphics Options and Device Parameters Dictionary" on page 577 for a description of each parameter. You must specify all of the following parameters:

COLORS=(*device-color-list*)
specifies one or more colors for the device.

Each color must be a valid color name. See "Color-Naming Schemes" on page 368. When specifying multiple colors, separate each color name with a space.

Note If COLORS= is not specified, the device is created, but the GDEVICE procedure issues an error message.

Example colors=(black cx95c051 cxA359B2 cxD65259 cx69D6D2 cxFFB74F cx929cff)

MODULE=*driver-module*
specifies the name of the driver module.

XMAX=*width* <IN | CM>
specifies the maximum width of the device output.

YMAX=*height* <IN | CM>
specifies the maximum height of the device output.

XPIXELS=*width-in-pixels*
specifies the device output width in pixels.

YPIXELS=*height-in-pixels*
specifies the device output height in pixels.

device-row-column-parameters
specifies device row and column parameters LCOLS= and LROWS= for landscape, PCOLS= and PROWS for portrait, or all four for both.

LCOLS=*landscape-columns*
specifies the number of landscape columns.

Requirement You must specify LROWS= with LCOLS=.

LROWS=*landscape-rows*
specifies the number of landscape rows.

Requirement You must specify LCOLS= with LROWS=.

PCOLS=*portrait-columns*
specifies the number of portrait columns.

Requirement You must specify PROWS= with PCOLS=.

PROWS=*portrait-rows*
specifies the number of portrait rows.

Requirement You must specify PCOLS= with PROWS=.

Optional Arguments

All optional parameters for the ADD statement correspond to device parameters of the same name that are documented in "Graphics Options and Device Parameters Dictionary" on page 577.

Note: Each entry in the dictionary states whether the entry (that is, the option or parameter) can be used by the GOPTIONS statement, the GDEVICE procedure, or any of the GDEVICE windows. Any entry that says it can be used by the GDEVICE procedure or a GDEVICE window is a device parameter. You can specify any parameter that can be specified with the GDEVICE procedure.

COPY Statement

Copies a device entry and places the copy in the current catalog. The original device entry can be either in the current catalog or in a different catalog.

Restriction: Not valid in Browse mode.

Requirement: You must have Write access to the catalog where the device entry is being copied.

Tip: If you copy and modify a device entry that is a Universal Printer shortcut device, then you also need to create a Universal Printer that has the same name.See "Device Categories and Modifying Default Output Attributes" on page 98 and "Managing

Universal Printing Using the SAS Windowing Environment" in *SAS 9.4 Universal Printing* for more information.

See: "Creating or Modifying Device Entries" on page 1181
"Example: Creating a Custom Device Entry with Program Statements" on page 1183

Syntax

COPY *device-entry copy-options*;

Required Arguments

device-entry
specifies the one-level name of the device entry to copy.

Restriction The entry must exist in the current catalog (the default) or the catalog specified by FROM= argument.

copy-options
specifies the copy options, which must be one or both of the following:

FROM=<*libref*>.*SAS catalog*
names the catalog from which to copy *device-entry*.

NEWNAME=*new-device-entry*
specifies a name for the copy of the device entry that is placed in the current catalog.

New-device-entry must be a valid name for a SAS catalog entry and cannot already exist in the current catalog. If you copy device entries across catalogs and you do not specify a new name, the GDEVICE procedure uses the original name for the new device entry.

DELETE Statement

Deletes the device entry from the current catalog.

Restriction: Not valid in Browse mode.

Requirement: You must have Write access to the current catalog to delete a device entry. You must specify the CATALOG= option in the PROC GDEVICE statement to have Write access to the current catalog.

Note: A device entry cannot be restored once it has been deleted. Depending on the environment in which you are using the GDEVICE procedure, you might be asked to verify that you really want to delete the entry.

Syntax

DELETE *device-entry*;

Required Argument

device-entry
specifies the one-level name of device entry to delete.

Restriction The entry must exist in the current catalog.

FS Statement

Switches from program mode to the GDEVICE windows.

Requirement: You must be running SAS in a windowing environment.

Syntax

FS;

LIST Statement

Lists all of the parameters of the specified device entry in the Output window.

Default: _ALL_

See: "Running the GDEVICE Procedure in Program Mode" on page 1174

Syntax

LIST *device-entry* | _ALL_ | _NEXT_ | _PREV_ | DUMP;

Required Argument

Specify one of the following:

device-entry
specifies the one-level name of the device entry whose contents you want to list.

Restriction The entry must exist in the current catalog.

ALL
lists only the name, description, and creation date of all device entries in the current catalog. If no entries exist in the catalog, the GDEVICE procedure issues a message.

NEXT
lists the contents of the next device entry. The GDEVICE procedure lists the first entry in the catalog if no entries have been previously listed.

PREV
lists the contents of the previous device entry. If you have not previously listed the contents of a device entry, the GDEVICE procedure issues this message: `No objects preceding current object.`

DUMP
lists detailed information about all device entries in the current catalog. Depending on the number of device entries in the catalog, the DUMP option can create a large amount of output.

MODIFY Statement

Changes the values in a device entry.

Restriction: Not valid in Browse mode.

Requirement: You must have Write access to the current catalog to modify a device entry. You must specify the CATALOG= option in the PROC GDEVICE statement to have Write access to the current catalog.

See: "Creating or Modifying Device Entries" on page 1181
"Example: Creating a Custom Device Entry with Program Statements" on page 1183

Syntax

MODIFY *device-entry device-parameters*;

Required Arguments

device-entry
specifies the one-level name of the device entry that you want to modify.

Restriction The entry must exist in the current catalog.

device-parameters
specifies the parameters that you want to modify. These can be any of the parameters that you can specify with the ADD statement, whether listed as required or optional for the ADD statement. See "ADD Statement" on page 1167 for more information. Refer to "Graphics Options and Device Parameters Dictionary" on page 577 for a description of each parameter.

Details

To modify a device entry, create your own catalog and then copy the device entries that you need into it. You can then change your personal copies of the device entries without affecting the original drivers in SASHELP.DEVICES. (To copy device entries, use either the COPY statement, the COPY command that is available after

you select **Import Device Entry** from the DIRECTORY` window's File menu, or the CATALOG procedure, which is part of Base SAS.)

CAUTION
Be careful when modifying device entries in program mode. In program mode, you cannot cancel any modifications that you have just made. To change a value that you have modified, you must use another MODIFY statement to replace the original value or reset it to its default. (In the GDEVICE windows, you can enter the CANCEL command in the command line to cancel changes that you have made to the fields.)

QUIT Statement

Saves all modifications made to device entries during the procedure. Exits the GDEVICE procedure.

Aliases: END
STOP

Syntax

QUIT;

RENAME Statement

Changes the name of the device entry to the name specified in the statement.

Restriction: Not valid in Browse mode.

Requirement: You must have Write access to the current catalog to rename a device entry. You must specify the CATALOG= option in the PROC GDEVICE statement to have Write access to the current catalog.

Syntax

RENAME *device-entry-name* NEWNAME=*new-device-entry-name*;

Required Arguments

device-entry-name
specifies the one-level name of the device entry that you want to rename.

Restriction The entry must exist in the current catalog.

NEWNAME=*new-device-entry-name*
specifies the new entry name. *New-device-entry-name* must be a valid name for a SAS catalog entry and cannot already exist in the current catalog. If the name already exists, the GDEVICE procedure issues an error message.

Usage: GDEVICE Procedure

Ways to Use the GDEVICE Procedure

Procedure Windows versus Program Code

There are two ways to use the GDEVICE procedure:

- browse or edit the fields in the GDEVICE procedure windows (windowing mode). See "Running the GDEVICE Procedure in a Windowing Environment" on page 1173.
- submit GDEVICE procedure statements in a SAS program (program mode). See "Running the GDEVICE Procedure in Program Mode" on page 1174.

If you run SAS in a windowing environment, you can use either the GDEVICE procedure windows or the GDEVICE procedure statements. In a windowing environment, the GDEVICE procedure automatically opens the GDEVICE procedure windows.

If you run SAS in line mode or batch mode, you can use only GDEVICE procedure statements. In a non-windowing environment, the GDEVICE procedure automatically uses line mode.

Both methods provide identical functionality and enable you to display or modify device parameters or create new device entries.

Running the GDEVICE Procedure in a Windowing Environment

In a windowing environment, open the GDEVICE windows by submitting the PROC GDEVICE statement without the NOFS option:

```
proc gdevice;
```

This opens the DIRECTORY window in Browse mode. This window lists all of the device entries in the current catalog. (See "The Current Catalog" on page 1164.)

To open the DIRECTORY window in Edit mode, or to specify a different catalog, include the CATALOG= option in the PROC GDEVICE statement.

From the DIRECTORY window, you can select the device entry that you want to work with and open other GDEVICE windows in which you can view or modify device parameters. For more information, see “Using the GDEVICE Windows” on page 1174.

In a windowing environment, you can switch between the GDEVICE windows and program statements while you are running the procedure. See the “FS Statement” on page 1170 and the “NOFS” on page 1166 option in “PROC GDEVICE Statement” on page 1166.

To exit the GDEVICE windows, submit the End command or close the window.

Running the GDEVICE Procedure in Program Mode

If you are in a non-windowing or batch environment, the GDEVICE procedure automatically starts in program mode. If you are in a windowing environment, specify the NOFS option to start the GDEVICE procedure in program mode:

```
proc gdevice nofs;
```

By default, the GDEVICE procedure accesses the current catalog in Browse mode and prompts you in the LOG to enter additional program statements. (See “The Current Catalog” on page 1164.) To specify the current catalog, include the CATALOG= option in the PROC GDEVICE statement.

Once you start the GDEVICE procedure, you can enter and run additional statements without resubmitting the PROC GDEVICE statement. You can exit the GDEVICE procedure in these three ways:

- submit the END, QUIT, or STOP statement
- submit another PROC statement or DATA step
- exit your SAS session

PROC GDEVICE procedure output is displayed in the Output window.

Using the GDEVICE Windows

About the GDEVICE Windows

In the SAS windowing environment, you can use GDEVICE windows instead of program mode to view, modify, copy, create, or delete device entries.

Note: The GDEVICE windows are not available in SAS Studio.

You can perform tasks in the GDEVICE windows by entering values in the fields, by using the menus, and by issuing commands from the command line.

These are the thirteen GDEVICE windows in order of appearance:

- DIRECTORY Window
- Detail Window
- Parameters Window
- Gcolors Window
- Chartype Window
- Colormap Window
- Metagraphics Window
- Gprolog Window
- Gepilog Window
- Gstart Window
- Gend Window
- Host File Options Window
- Host Commands Window

The fields in these windows represent device entry parameters. The GDEVICE windows group the device parameters by topic. When you open a device entry in Edit mode, you can modify the fields directly. For a description of each field, see the corresponding parameter in "Graphics Options and Device Parameters Dictionary" on page 577.

Note: Each entry in the dictionary states whether the entry (that is, the option or parameter) can be used by the GOPTIONS statement, the GDEVICE procedure, or any of the GDEVICE windows. Any entry that says it can be used by the GDEVICE procedure or a GDEVICE window is a device parameter.

Note: The parameters are sometimes an abbreviation of the field names. For example, in the Detail window, the "Driver query" field corresponds to the DRVQRY parameter, and the "Queued messages" field corresponds to the QMSG parameter.

GDEVICE Window Commands

You can navigate and manipulate the GDEVICE windows by entering commands on the command line, or selecting items from the menus. For a complete description of all the GDEVICE window commands, open the Help for the GDEVICE windows. You can open the Help by entering `Help` on the command line or by selecting **Help** ⇨ **Using This Window**.

Note: In a Windows environment, the GDEVICE commands are presented on pop-up menus. Click the secondary mouse button on a GDEVICE window to access a pop-up menu.

DIRECTORY Window

This window is displayed when you start the GDEVICE procedure in window mode. It lists all the device entries in the default catalog or the catalog that you specified in the PROC GDEVICE statement. You can use it to do the following:

- copy, rename, or delete device entries in the catalog
- select a device entry whose parameters you want to browse or edit

You can enter these commands in the DIRECTORY window selection field:

B | S
: open the Detail window and browse (B) or, if you are in Edit mode, edit (S) the selected device entry.

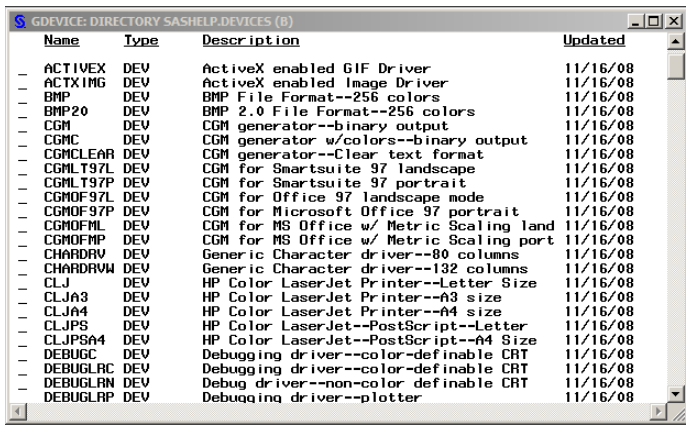
D
: delete the selected device entry. You cannot restore a device entry once it has been deleted.

E
: open the Detail window and edit the selected device entry.

R
: rename the device entry or the description, or both.

You cannot edit the TYPE and UPDATED fields in the DIRECTORY Window.

Figure 38.1 *The DIRECTORY Window*



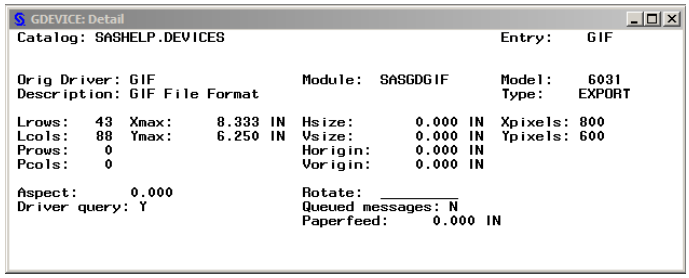
Name	Type	Description	Updated
ACTIVEX	DEV	ActiveX enabled GIF Driver	11/16/08
ACTXIMG	DEV	ActiveX enabled Image Driver	11/16/08
BMP	DEV	BMP File Format--256 colors	11/16/08
BMP20	DEV	BMP 2.0 File Format--256 colors	11/16/08
CGM	DEV	CGM generator--binary output	11/16/08
CGMC	DEV	CGM generator w/colors--binary output	11/16/08
CGMCLEAR	DEV	CGM generator--Clear text format	11/16/08
CGMLT97L	DEV	CGM for Smartsuite 97 landscape	11/16/08
CGMLT97P	DEV	CGM for Smartsuite 97 portrait	11/16/08
CGMOF97L	DEV	CGM for Office 97 landscape mode	11/16/08
CGMOF97P	DEV	CGM for Microsoft Office 97 portrait	11/16/08
CGMOFML	DEV	CGM for MS Office w/ Metric Scaling land	11/16/08
CGMOFMP	DEV	CGM for MS Office w/ Metric Scaling port	11/16/08
CHARDRV	DEV	Generic Character driver--80 columns	11/16/08
CHARDRVW	DEV	Generic Character driver--132 columns	11/16/08
CLJ	DEV	HP Color LaserJet Printer--Letter Size	11/16/08
CLJA3	DEV	HP Color LaserJet Printer--A3 size	11/16/08
CLJA4	DEV	HP Color LaserJet Printer--A4 size	11/16/08
CLJPS	DEV	HP Color LaserJet--PostScript--Letter	11/16/08
CLJPSA4	DEV	HP Color LaserJet--PostScript--A4 Size	11/16/08
DEBUGC	DEV	Debugging driver--color-definable CRT	11/16/08
DEBUGLRC	DEV	Debugging driver--color-definable CRT	11/16/08
DEBUGLRN	DEV	Debug driver--non-color definable CRT	11/16/08
DEBUGLRP	DEV	Debugging driver--plotter	11/16/08

After the DIRECTORY window appears, you can navigate through the GDEVICE windows by selecting the **View** ⇨ **Next Screen**.

Detail Window

This window contains device parameters that control basic characteristics of the device (for example, the size of the graphics output area).

Figure 38.2 *The Detail Window*



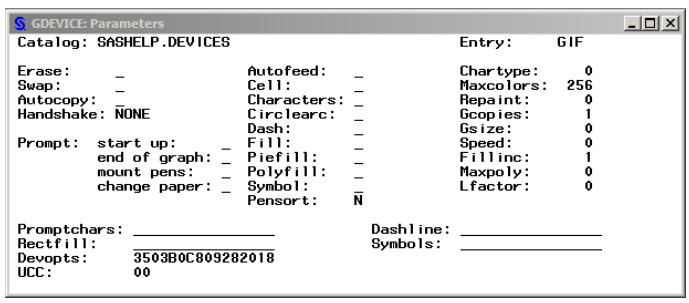
Parameters Window

This window includes additional device parameters that affect how graphs are drawn. For example, you can define the following parameters:

- whether certain graphics primitives are drawn by your hardware or by SAS/GRAPH
- whether to feed paper to printers or plotters automatically
- whether to have SAS/GRAPH prompt you with messages under certain conditions

Note: If the device does not support a hardware characteristic, the catalog entry cannot enable the support.

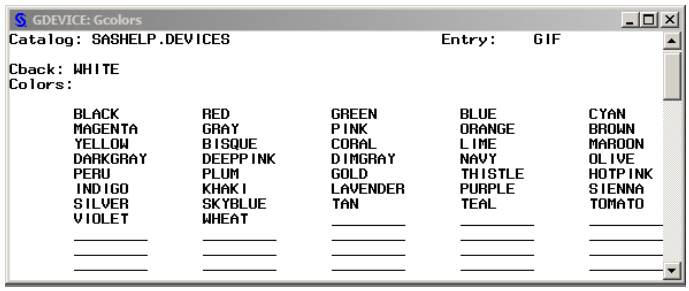
Figure 38.3 *The Parameters Window*



Gcolors Window

This window lists the colors that the device driver uses by default when the NOGSTYLE option is in effect. When you do not explicitly specify the color of a graphics feature in your program or in a GOPTIONS statement, SAS/GRAPH uses this list to determine what color to use.

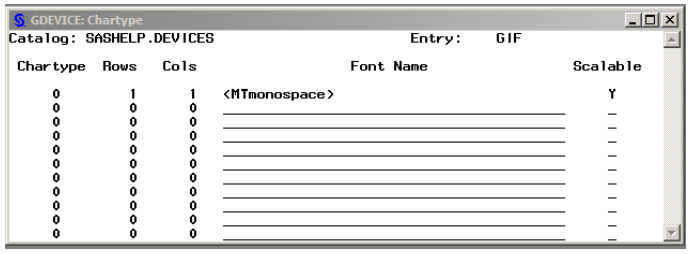
Figure 38.4 *The Gcolors Window (partial view)*



Chartype Window

This window lists the device-resident fonts that the device can use, along with information about the size of the characters. The Chartype value is the value to reference a font in another window. For example, you can enter a Chartype number in the Parameters window's Chartype field.

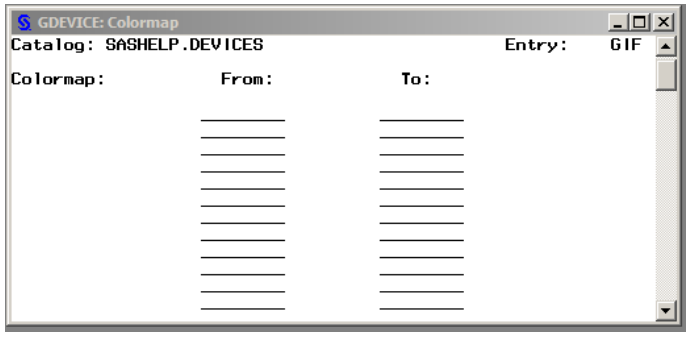
Figure 38.5 *The Chartype Window (partial view)*



Colormap Window

This window enables you to specify a color map for the device. The FROM field specifies the name to assign to the color designated by the *color* value. The TO field specifies a predefined SAS/GRAPH color name. Once you have defined the color mapping, the new color name is available for use in any color option. For example, map the color name DAFFODIL to the SAS color value PAOY. Specify COLOR=DAFFODIL anywhere the COLOR= option is supported. The driver substitutes the color value PAOY. Contact SAS Technical Support for assistance in determining predefined SAS color names.

Figure 38.6 *The Colormap Window (partial view)*



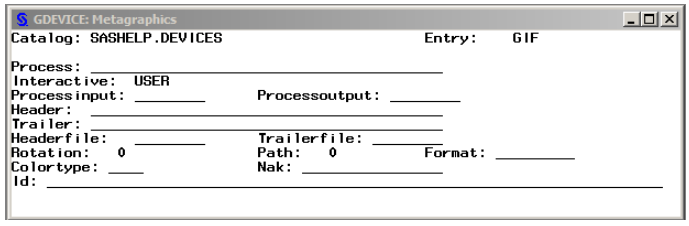
Metagraphics Window

This window is used by all drivers that support multiple color spaces (for example, RGB or CMYK). It is also used if the device entry is a Metagraphics (user-written) driver. Metagraphics drivers are created when a device entry that was provided by SAS cannot be adapted to support your graphics device. For information about Metagraphics drivers, contact SAS Technical Support.

CAUTION

Do not alter the fields in the Metagraphics window unless you are changing the color scheme (colortype) or building a Metagraphics driver.

Figure 38.7 *The Metagraphics Window*



Gprolog Window

This window enables you to specify one or more hexadecimal strings sent to the device before graphics commands are sent. Additional commands can be sent with the PREGPROLOG= and POSTGPROLOG= graphics options. See “Graphics Options and Device Parameters Dictionary” on page 577 for descriptions of these options.

Gepilog Window

This window enables you to specify one or more hexadecimal strings that are sent to the device after graphics commands are sent. Additional commands can be sent with the PREGEPILOG= and POSTGEPILOG= graphics options. See “Graphics Options and Device Parameters Dictionary” on page 577 for descriptions of these options.

Gstart Window

This window enables you to specify one or more hexadecimal strings that are placed at the beginning of each record of graphics data.

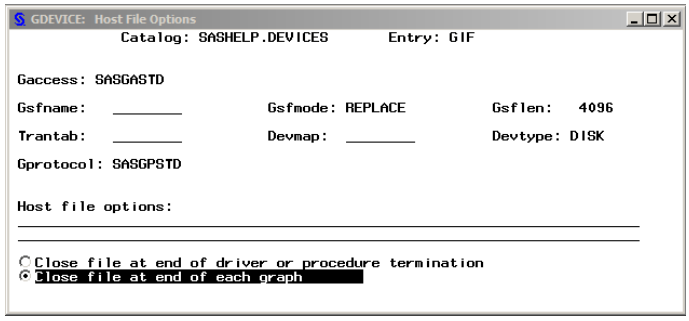
Gend Window

This window enables you to specify one or more hexadecimal strings that are placed at the end of each record of graphics data.

Host File Options Window

This window controls the output destination and formatting of the data stream produced by the driver. (Most of these values can also be specified with the GOPTIONS statement or with the FILENAME statement. See “Specifying the Graphics Output File Type for Your Graph” on page 124.)

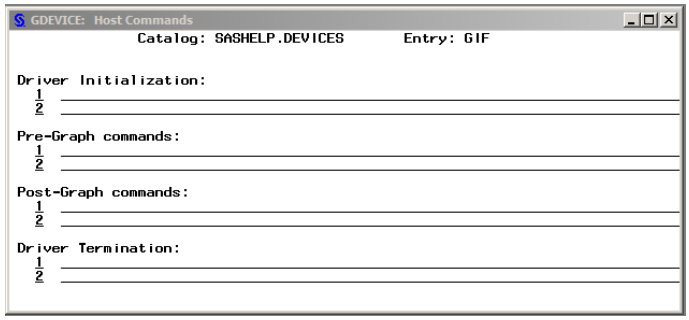
Figure 38.8 *The Host File Options Window*



Host Commands Window

This window stores the host commands issued at driver initialization, before and after each graph is produced, and at driver termination. These commands are typically used to send graphics output to a hardcopy device such as a printer or a plotter.

Figure 38.9 *The Host Commands Window*



Creating or Modifying Device Entries

Write Access, Personal Catalogs, and Device Entry Names

In order to add, modify, or delete device entries, you must have Write access to the catalog. On multi-user systems, your site administrator is usually the only person who has Write access to the SASHELP.DEVICES catalog and can make any changes. Therefore, when creating new entries or modifying existing ones, individual users usually work in a personal catalog. Be sure the catalog in which you store new or modified device entries is named DEVICES.

To use a device entry stored in a personal catalog, you must assign the GDEVICE*n* libref to the library that contains the device catalog. See “Device Catalogs” on page 1164.

It is a good idea to give a new or modified device entry a name that is different from the original. Then, if you want to use the original device, SAS/GRAPH can find that device when it searches the device catalogs. Remember that SAS/GRAPH searches the GDEVICE*n* libraries before it searches SASHELP.DEVICES and uses the first device that it finds whose name matches the one you have specified. (See “Search Order of Device Catalogs” on page 1165.)

For example, suppose there is a customized copy of PSCOLOR in your GDEVICE0.DEVICES catalog as well as the original in SASHELP.DEVICES. If you

specify DEV=PSCOLOR and the libref GDEVICE0 is assigned, SAS/GRAPH searches GDEVICE0.DEVICES first and uses the copy of PSCOLOR. Unless you cancel the GDEVICE0 libref, SAS/GRAPH will never find the original device entry in SASHELP.DEVICES. (To include SASHELP.DEVICES in the search path, you would need to cancel the GDEVICE0 libref.)

Creating a New Device Entry

Typically, you create a new device entry by copying an existing device and modifying its parameters to meet your needs. You can copy and modify a device entry in two ways:

- Use the DIR command to open the DIRECTORY window, and then use the COPY command to make a copy of an existing device entry. Edit the new entry and modify its parameters. The existing device entry can be from any catalog. (See help for information about using GDEVICE windows and commands. You can open the Help by entering `Help` on the command line or by selecting **Help** ⇨ **Using This Window**.)
- In program mode, use the COPY statement to make a copy of the device entry. Use the MODIFY statement to change the parameters. See "Example: Creating a Custom Device Entry with Program Statements" on page 1183.

  Note: If you copy and modify a device entry that is a Universal Printer shortcut device, then you also need to create a Universal Printer that has the same name. See "Device Categories and Modifying Default Output Attributes" on page 98 and "Managing Universal Printing Using the SAS Windowing Environment" in *SAS 9.4 Universal Printing* for more information.

If you want to start with a blank device entry and fill in values for the parameters, use the EDIT command from the DIRECTORY window or use the ADD statement with program mode PROC GDEVICE.

With either method, you provide values for the parameters listed in "Required Arguments" on page 1167. If you copy and modify an existing entry, all the required parameters have values. If you create a new entry with GDEVICE windows, you are prompted to fill in the appropriate fields.

Note: When you change a field in a device entry that was provided by SAS (either the original device entry in SASHELP.DEVICES or a copy), SAS/GRAPH asks whether you really want to change the entry.

Modifying an Existing Device Entry

Typically, you modify an existing device entry when you want to change the device parameters permanently in order to customize a device entry. The process is similar to creating a new catalog entry. Copy the device entry that you want to modify into your personal catalog. Change the parameters in the new device entry. See "Example: Creating a Custom Device Entry with Program Statements" on page 1183 for an example of creating a custom device entry.

Changing Device Parameters Temporarily

You can change some device parameters temporarily by overriding their settings with graphics options in a GOPTIONS statement. In this case, the settings remain in effect until you change them or end your SAS session. For details, see "Overriding Style Attributes with SAS/GRAPH Statement Options" on page 329 and "Precedence of Appearance Option Specifications" on page 330. See also "Style Attributes versus Device Entry Parameters" on page 323.

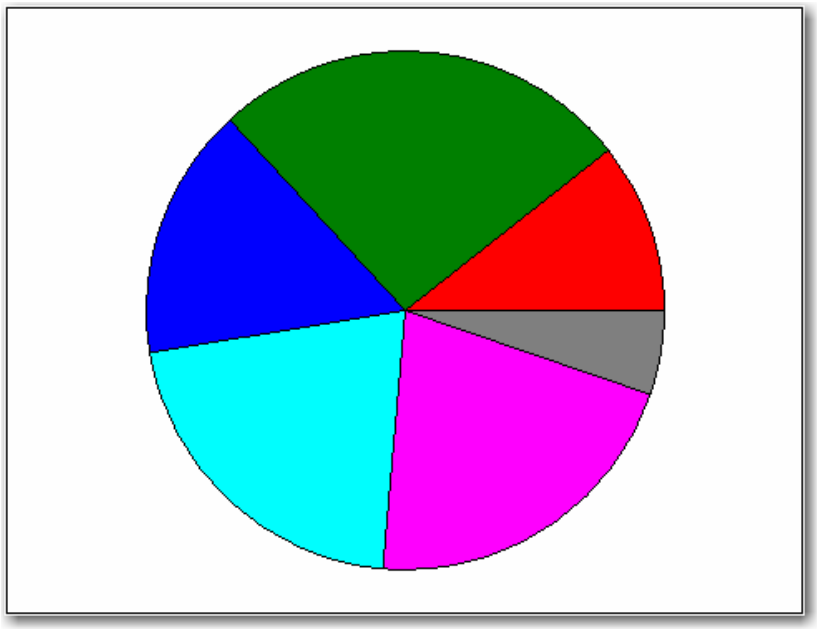
Example: Creating a Custom Device Entry with Program Statements GDEVICE Procedure

Example 1: Creating a Custom Device Entry with Program Statements

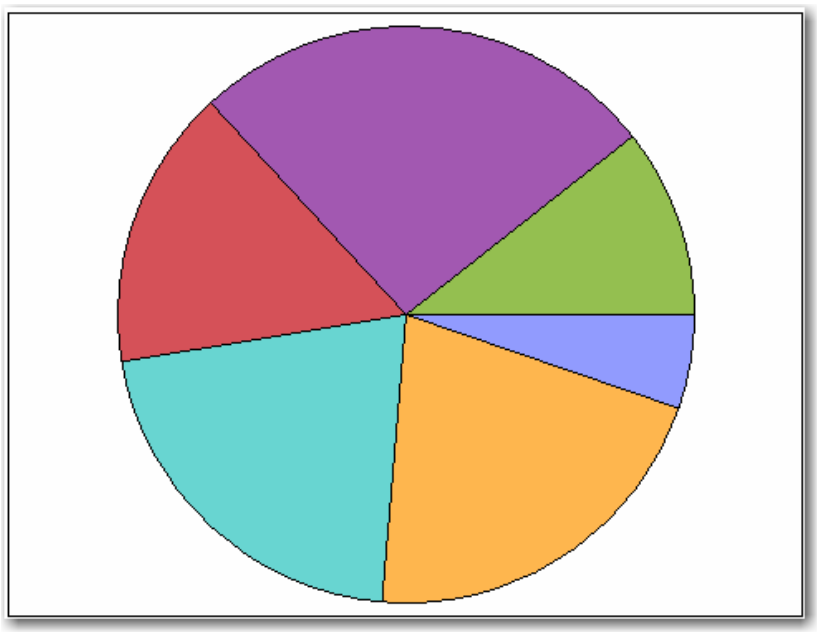
Features:	COPY statement options FROM= NEWNAME= MODIFY statement options COLORS= DESCRIPTION= LIBNAME statement
Sample library member:	GDVCSTOM
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses the GDEVICE procedure statements COPY and MODIFY statements to create a customized version of the JPEG device. It copies the original entry into a personal devices catalog and changes the device parameters. After running this example, all output generated with the JPEG device will use the customized color list created by this example.

Output 38.1 *Pie Chart Created with Default JPEG Device Entry*



Output 38.2 *Pie Chart Created with Customized JPEG Device Entry*



Program

```
goptions reset=all border;

options nogstyle;

goptions device=jpeg;
proc gchart data=sashelp.class;
  pie age/discrete noheading value=none;
```

```
 run;
quit;

libname gdevice0 ".";

proc gdevice nofs catalog=gdevice0.devices;

   copy jpeg from=sashelp.devices newname=myjpeg;

   modify myjpeg
        description="JPEG with new color list"
        colors=(black cx95c051 cxA359B2 cxD65259 cx69D6D2 cxFFB74F
cx929cff);

quit;

goptions target=myjpeg;
proc gchart data=sashelp.class;
  pie age/discrete noheading value=none;
 run;
quit;

options gstyle;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Turn styles off so that the device color list is used.

```
options nogstyle;
```

Generate the pie chart using the JPEG device.

```
goptions device=jpeg;
proc gchart data=sashelp.class;
  pie age/discrete noheading value=none;
 run;
quit;
```

Assign the libref GDEVICE0. The LIBNAME statement assigns the gdevice0 libref to the aggregate file storage location that will contain the DEVICES catalog and the modified JPEG device entry.

```
libname gdevice0 ".";
```

Start the GDEVICE procedure. The NOFS option causes GDEVICE to use program mode. The CATALOG= option assigns GDEVICE0.DEVICES as the current device catalog. If the DEVICES catalog does not already exist in the library, it is automatically created.

```
proc gdevice nofs catalog=gdevice0.devices;
```

Copy the original device entry from SASHELP.DEVICES to the current catalog. The NEWNAME= option specifies a name for the copy of JPEG that is placed in GDEVICE0.DEVICES. The name of a catalog entry cannot exceed eight characters.

```
   copy jpeg from=sashelp.devices newname=myjpeg;
```

Modify the new entry. The DESCRIPTION= option specifies a new device description that appears in the catalog listing. The COLORS= option defines a new color list.

```
   modify myjpeg
        description="JPEG with new color list"
        colors=(black cx95c051 cxA359B2 cxD65259 cx69D6D2 cxFFB74F
cx929cff);
```

Exit the procedure.

```
quit;
```

Test the new device entry. The TARGET= option specifies the new device. The GDEVICE0 libref is already defined, so SAS/GRAPH searches GDEVICE0 for the specified device entry. The GHART procedure generates a pie chart with the new color list.

```
goptions target=myjpeg;
proc gchart data=sashelp.class;
  pie age/discrete noheading value=none;
 run;
quit;
```

Turn styles back on.

```
options gstyle;
```

Chapter 39

GFONT Procedure

Overview: GFONT Procedure

About the GFONT Procedure

The GFONT procedure displays fonts and creates SAS/GRAPH fonts for use in SAS/GRAPH programs. These fonts can contain standard Roman alphabet characters, foreign language characters, symbols, logos, or figures.

Displaying Fonts

You can use the GFONT procedure output when you want to do the following tasks:

- review the characters that are available in SAS/GRAPH fonts
- examine the default device-resident font for your device
- see the character codes associated with font characters
- view the hexadecimal values associated with font characters
- modify the color and height of font characters
- draw reference lines around font characters

See “Example 1: Displaying Fonts with Character Codes” on page 1214.

Creating Fonts

The GFONT procedure enables you to create and store any series of figures or alphabet fonts that you can digitize or draw using X and Y coordinates. Font characters or figures can be displayed with any SAS/GRAPH statement or option that allows for a font specification and a text string. See “Storing or Displaying User-Created Fonts: GFONT0 Libref” on page 1190 and “Font Data Sets and the GFONT Procedure” on page 1201 for more information.

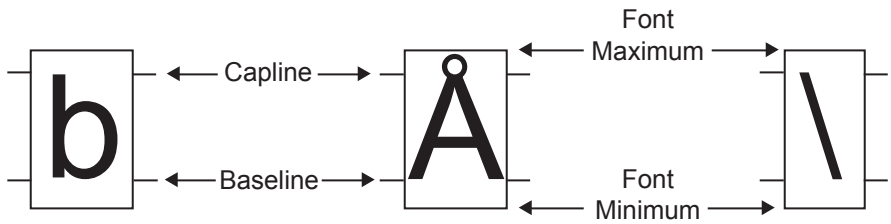
Concepts: GFONT Procedure

Font Terminology and Characteristics

Some specialized terms are associated with font characteristics:

- The *capline* is the highest point of a normal uppercase letter.
- The *baseline* is the line upon which the characters rest.
- The *font maximum* is the highest vertical coordinate.
- The *font minimum* is the lowest vertical coordinate.

Figure 39.1 *Font Characteristics Terminology*



Specialized terms are also associated with font types:

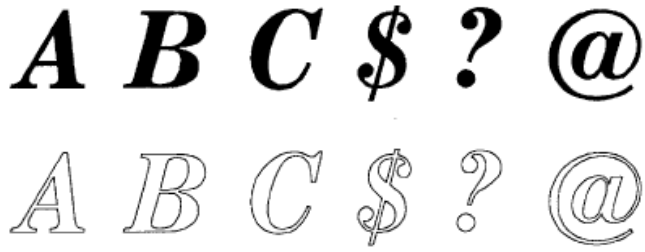
- A *uniform font* is a font in which all of the characters occupy exactly the same amount of space. Each character in a uniform font is placed in the center of its space, and a fixed amount of space is added between characters.
- A *proportional font* is a font in which each character occupies a space that is relative to its width.
- A *stroked font* is drawn with discrete line segments or circular arcs. This is a stroked font with several characters from the Simplex font.

Figure 39.2 *Characters from a Stroked Font*

- A *polygon font* is drawn with one or more line segments or circular arcs.
- A *filled font* is a polygon font in which the areas between the lines are solid.
- An *outline font* is a polygon font in which the areas between the lines are empty.

Here are examples of a filled font and an outline font.

Figure 39.3 *Filled and Outline Characters from Polygon Fonts*



In the GFONT procedure, the term *line segment* means a continuous line that can change direction. All font characters are drawn with line segments. The letter C is drawn with one line segment, whereas the letter A can be drawn with two.

Polygon characters can be drawn with one or more line segments. In a polygon font the following is true:

- A character can consist of a single polygon. The letter C above is a single polygon with one line segment
- A character can consist of multiple polygons. The question mark consists of two polygons, each drawn with a separate line segment

- A character can include holes. The letter A is a polygon with a hole in it. It is drawn with one line segment that is broken to form the outer boundary of the figure and the boundary of the hole.

Storing or Displaying User-Created Fonts: GFONT0 Libref

The GFONT procedure stores user-created SAS/GRAPH fonts in the location that is associated with the libref GFONT0. Before you create or display a user-created SAS/GRAPH font, submit a LIBNAME statement to associate the libref GFONT0 with a location where the font is stored, as follows:

```
LIBNAME gfont0 "SAS-data-library";
```

The GFONT0 library is the first place that SAS/GRAPH software searches for fonts. Always assign GFONT0 to the library that contains your personal SAS/GRAPH fonts. If you have personal SAS/GRAPH fonts in more than one SAS library, assign them librefs in the sequence GFONT0, GFONT1, GFONT2, and so on. The search for entries terminates if there is a break in the numbering sequence. If the libref GFONT0 is not defined, by default SAS/GRAPH software begins searching for fonts in SASHELP.FONTS.

If a user-created SAS/GRAPH font has the same name as a font supplied by SAS, and if the libref GFONT0 has been defined, then the user-created SAS/GRAPH font is used, because GFONT0 is first in the search order.

To cancel or redefine the libref GFONT*n*, submit the following statement:

```
LIBNAME GFONTn;
```

Specifying Units for Creating Fonts

When the syntax of an option includes *units*, specify one of the units:

CELLS	character cells
CM	centimeters
IN	inches
PCT	percentage of the graphics output area
PT	points

If you omit *units*, a unit specification is searched for in the following order:

1. the value of GUNIT= in a GOPTIONS statement
2. the value of units in the ODS STYLE
3. the default unit CELLS

Syntax: GFONT Procedure

Requirement: One font name is required. To display a font, include the NOBUILD option. To create a font, include the DATA= option.

Supported Statements: FOOTNOTE, NOTE, TITLE

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, "Plotting a Cloud Analytic Services (CAS) In-Memory Table," on page 153.

Before you run the GFONT procedure, use the DEVICE= option in a GOPTIONS statement or in an OPTIONS statement to specify an output device.

PROC GFONT NAME=*SAS/GRAPH-font* | *device-resident-font* | *system-font*
NOBUILD <*options-for-displaying-fonts*>;

PROC GFONT NAME=*SAS/GRAPH-font*
DATA=*font-data-set* <*options-for-creating-fonts*>;

PROC GFONT Statement

The PROC GFONT statement can either create SAS/GRAPH fonts or display existing SAS/GRAPH fonts. The GFONT procedure names the font to be created or displayed. If the GFONT procedure creates a font, then an input data set name is required. You can modify the design and appearance of the fonts that you create or display, and specify a destination catalog for graphics output.

Requirement: When you create or display a user-created font, you must define the libref GFONT0. See "Storing or Displaying User-Created Fonts: GFONT0 Libref" on page 1190.

Syntax

Form 1: Display a font:

PROC GFONT NAME=*SAS/GRAPH-font* | *device-resident-font* | *system-font*
NOBUILD<*options-for-displaying-fonts*>;

Form 2: Create a font:

PROC GFONT DATA= *font-data-set*NAME=*SAS/GRAPH-font*<*options-for-creating-fonts*>;

Summary of Optional Arguments

Creating fonts

BASELINE=*y*
specifies the vertical coordinate in the font data set that is the baseline of the characters.

CAPLINE=*y*
specifies the vertical coordinate in the font data set that is the capline of the characters.

CHARSPACETYPE=DATA | FIXED | NONE | UNIFORM
specifies the type of intercharacter spacing.

CODELEN=1 | 2
specifies the length in bytes of the CHAR variable.

FILLED
specifies that the characters in a polygon font are filled.

KERNDATA=*kern-data-set*
specifies the data set that contains kerning information.

MWIDTH=*character-width*
specifies the width of a character in a uniform font, where *character-width* is the number of font units.

NODISPLAY
specifies that the GFONT procedure is not to display the font that it is creating.

NOKEYMAP
specifies that the current key map is ignored when you create and then use the font that is created.

RESOL=1 | 2 | 3 | 4
controls the resolution of the fonts by specifying the number of bytes (1 through 4) for storing coordinates in the font.

ROMHEX
specifies that hexadecimal values are displayed automatically below the font characters when the GFONT procedure displays the font.

SHOWROMAN
specifies that character codes are displayed automatically below the font characters when the GFONT procedure displays the font.

SPACEDATA=*space-data-set*
specifies the SAS data set that contains font spacing information.

UNIFORM
specifies that characters are spaced uniformly rather than proportionately.

Displaying fonts

CTEXT=*text-color*
specifies a color for the body of the characters.

GOUT=*<libref.>output-catalog*
specifies the SAS catalog in which to save the graphics output generated by the display of the font.

HEIGHT=*character-height <units>*
specifies the height of the font characters in number of units, *n*.

NOKEYMAP
specifies that the current key map is ignored when displaying the font and its character codes or hexadecimal values.

NOROMAN
turns off the automatic display of character codes that are created when you use the SHOWROMAN option during font creation.

NOROMHEX

turns off the automatic display of hexadecimal values for single-byte characters.

REFCOL=*reference-line-color*
specifies a color for reference lines.

REFLINES
draws reference lines around each displayed character.

ROMCOL=*code-color*
specifies the color of the character codes or hexadecimal values that are displayed with the SHOWROMAN and ROMHEX options.

ROMFONT=*font*
specifies the font for character codes and hexadecimal values that are displayed by the SHOWROMAN and ROMHEX options.

ROMHEX
displays hexadecimal values below the font characters.

ROMHT=*height <units>*
specifies the height of the character codes and the hexadecimal values that are displayed with the SHOWROMAN and ROMHEX options in number of units, *n*.

SHOWALL
displays the font with a space for every possible character position whether a font character exists for that position.

SHOWROMAN
displays character codes below the font characters even if they are not displayed automatically with the font.

Required Arguments

DATA=*font-data-set*
specifies the SAS data set that the GFONT procedure uses to build the font. The data set must be sorted by the variables CHAR and SEGMENT.

Default The GFONT procedure uses the most recently created data set.

See "The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

Example "Example 2: Creating Figures for a Symbol Font" on page 1216

NAME=*SAS/GRAPH-font-name* | *device-resident-font-name* | *system-font-name*
when displaying a font, specifies the name of the font to be displayed. The name can be the name of any of the following types of fonts:

- a SAS/GRAPH font that is stored in the SASHELP.FONTS catalog or a SAS/GRAPH font that was created and stored in a GFONT*n* catalog. These fonts can be used only by SAS/GRAPH procedures or other procedures that generate SAS/GRAPH output files.
- a system font that can be used by any SAS procedure and by other software, such as Microsoft Word. SAS/GRAPH installs and registers a set of TrueType fonts, and it is recommended that you use these fonts whenever possible.
- a device-resident font that is burned into the chips in a device's hardware. These fonts are specific to the device being used and are not portable

between devices. Some device resident fonts such as Helvetica can also be present as system fonts.

Alias N=

Note Device resident font names must be enclosed in quotation marks.

NAME=*SAS/GRAPH-font-name*
when creating a font, specifies a name for the font that you are creating. *SAS/GRAPH-font-name* is the name of a catalog entry, and must be a valid SAS name of no more than eight characters. You cannot specify NONE, or the name of a SAS/GRAPH font that is shipped with SAS/GRAPH software.

Alias N=

Example "Example 2: Creating Figures for a Symbol Font" on page 1216

NOBUILD
specifies that the GFONT procedure is to display an existing font. The NOBUILD argument tells the GFONT procedure that no font is being created and not to look for an input data set.

Alias NB

Example "Example 1: Displaying Fonts with Character Codes" on page 1214

Options for Displaying Fonts

Options to display a font can be used when you create a font if you also display it (that is, you do not use the NODISPLAY option in the PROC GFONT statement). However, none of the display options affect the design and appearance of the stored font except the NOKEYMAP, SHOWROMAN, and ROMHEX options. See also "Specifying Units for Creating Fonts" on page 1190.

CTEXT=*text-color*
specifies a color for the body of the characters. If you do not use the CTEXT= option, a color specification is searched for in the following order:

- the CTEXT= option in the procedure statement
- the CTEXT= option in a GOPTIONS statement
- the color specified in the ODS style
- the first color in the color list

Alias CT=

Note The CTEXT= value is not stored as part of the font.

Example "Example 2: Creating Figures for a Symbol Font" on page 1216

GOUT=<*libref.*>*output-catalog*
specifies the SAS catalog in which to save the graphics output generated by the display of the font. You can use the GREPLAY procedure to view the output that is stored in the catalog.

If you omit the libref, SAS/GRAPH looks for the catalog in the temporary WORK library, and creates the catalog if it does not exist.

See "Specifying the Catalog Name and Entry Name for Your GRSEGs " on page 138

HEIGHT=*character-height <units>*
specifies the height of the font characters in number of units, *n*. Height is measured from the minimum font measurement to the capline.

Alias H=

Default 2

Example "Example 1: Displaying Fonts with Character Codes" on page 1214

NOKEYMAP
specifies that the current key map is ignored when displaying the font and its character codes or hexadecimal values. If you do not use the NOKEYMAP option when you display a font, the current key map remains in effect. If any characters in the font are not available through the current key map, they are not displayed and a warning is issued in the SAS log. This happens when not all characters in the font are mapped into the current key map.

Displaying a font using the NOKEYMAP option enables you to see all of the characters in the font, including those that are not mapped into your current key map.

Note Only the characters that are mapped into your current key map are available.

NOROMAN
turns off the automatic display of character codes that are created when you use the SHOWROMAN option during font creation.

Alias NR

NOROMHEX
turns off the automatic display of hexadecimal values for single-byte characters that are created when you use the ROMHEX option during font creation.

Alias NOHEX

REFCOL=*reference-line-color*
specifies a color for reference lines. If you do not use the REFCOL= option, a color specification is searched for in the following order:

- the CTEXT= option in a GOPTIONS statement
- the color specified in the ODS style
- the first color in the color list

REFLINES
draws reference lines around each displayed character. Vertical reference lines show the width of the character. Horizontal reference lines show the font maximum and the font minimum, as well as the baseline and the capline.

See "Font Terminology and Characteristics" on page 1188

ROMCOL=*code-color*

specifies the color of the character codes or hexadecimal values that are displayed with the SHOWROMAN and ROMHEX options. If you do not use the ROMCOL= option, a color specification is searched for in the following order:

- the color specified by the CTEXT= option in a GOPTIONS statement
- the color specified in the current style or, if the NOGSTYLE option is specified, then the default color is black for the Java and ActiveX devices and the first color in the color list for all the other devices

Alias RC=

Note The ROMCOL= value is not stored as part of the font.

Example "Example 1: Displaying Fonts with Character Codes" on page 1214

ROMFONT=*font*

specifies the font for character codes and hexadecimal values that are displayed by the SHOWROMAN and ROMHEX options. If you do not use the ROMFONT= option, a font specification is searched for in the following order:

- the value of the ODS STYLE variable
- the FTEXT= option in a GOPTIONS statement
- fonts supplied by SAS
- the device-resident font

Alias RF=

Example "Example 1: Displaying Fonts with Character Codes" on page 1214

ROMHEX

displays hexadecimal values below the font characters. If you use both the ROMHEX and SHOWROMAN options, both the character codes and the hexadecimal values are displayed. You can also use the ROMHEX option when you create a font.

Alias HEX

See ROMHEX"ROMHEX" on page 1200 (for creating fonts)

ROMHT=*height <units>*

specifies the height of the character codes and the hexadecimal values that are displayed with the SHOWROMAN and ROMHEX options in number of units, *n*. If you do not use the ROMHT= option, a height specification is searched for in the following order:

- the HEIGHT specified in the ODS STYLE
- the HTEXT= option in a GOPTIONS statement

Alias RH=

Default 1

Example "Example 1: Displaying Fonts with Character Codes" on page 1214

SHOWALL
displays the font with a space for every possible character position whether a font character exists for that position. The characters that are displayed are those available under your current key map, unless you use the NOKEYMAP option. The SHOWALL option usually is used in conjunction with the ROMHEX option, to display all possible hexadecimal values. If, under your current key map, a font character is available for a position, it is displayed above the hexadecimal value. If no character is available for a position, the space above the hexadecimal value is blank. You can use the SHOWALL option to show where undefined character positions fall in the font.

SHOWROMAN
displays character codes below the font characters even if they are not displayed automatically with the font. If you use both the SHOWROMAN option, and the ROMHEX option, both the character codes, and the hexadecimal values are displayed. You can also use the SHOWROMAN option when you create a font.

Alias SR

Example "Example 1: Displaying Fonts with Character Codes" on page 1214

Options for Creating Fonts

Options to display a font can be used when you create a font if you also display it (that is, you do not use the NODISPLAY option in the PROC GFONT statement). However, none of the display options affect the design and appearance of the stored font except the NOKEYMAP, SHOWROMAN, and ROMHEX options. See also "Specifying Units for Creating Fonts" on page 1190.

BASELINE=*y*
specifies the vertical coordinate in the font data set that is the baseline of the characters. The baseline is the line upon which the letters rest. If you do not use the BASELINE= option, the GFONT procedure uses the lowest vertical coordinate of the first character in the font data set.

Alias B=

CAPLINE=*y*
specifies the vertical coordinate in the font data set that is the capline of the characters. The capline is the highest point of normal that case the capline, and the font maximum are the same. See Figure 39.140 on page 1189 for an illustration of capline, and font maximum.

If you use the CAPLINE= option, when the height of a character is calculated, any part of the character that is above the capline is ignored in the calculation.

You can use this option to prevent an accented capital like A from being shortened to accommodate the accent. If you do not use the CAPLINE= option, the capline and the font maximum are the same. The A is shortened to make room for the accent below the capline. However, if the CAPLINE= option is used, the top of the letter A is at the capline, and the accent is drawn above the capline, and below the font maximum.

Alias C=

CHARSPACETYPE=DATA | FIXED | NONE | UNIFORM
specifies the type of intercharacter spacing. The following values are valid:

DATA
specifies that the first observation for each character sets the width of that character.

When CHARSPACETYPE=DATA, the PTYPE variable is required, and the observation that specifies the width of the character must have a PTYPE value of W. See "Overview: The Font Data Set" on page 1202 for details about the PTYPE variable.

Intercharacter spacing is included in the character's width. If the first observation for the letter A specifies a character width of 10 units, and the A occupies 8 units, the remaining 2 units serve as intercharacter spacing.

Note The character can extend beyond the width that you specified in the first observation if desired.

FIXED
adds a fixed amount of space between characters based on the font size.

The width of the individual character is determined by the data that generates the character.

NONE
specifies that no space is added between characters.

The width of the individual character is determined by the data that generates the character. This type of spacing is useful for script fonts in which the characters should appear connected.

UNIFORM
specifies that the amount of space that is used for each character is uniform, not proportional.

Each character occupies the same amount of space. In uniform spacing the letters m and i occupy the same amount of space, in proportional spacing m occupies more space than i. In uniform spacing, the character is always centered in the space, and a fixed space is added between characters.

When UNIFORM is specified, the amount of space that is used for each character is one of the following:

Alias CSP=

Default CHARSPACETYPE=FIXED

Note Specifying CHARSPACETYPE=UNIFORM is the same as using the UNIFORM option.

CODELEN=1 | 2
specifies the length in bytes of the CHAR variable. To specify double-byte character sets for languages such as Chinese, Japanese, or Korean, use CODELEN=2.

FILLED
specifies that the characters in a polygon font are filled.

Alias F

Default 1

Restriction If you specify a double-byte character set, the KERNDATA= option and SPACEDATA= option are ignored.

Example "Example 2: Creating Figures for a Symbol Font" on page 1216

KERNDATA=*kern-data-set*

specifies the data set that contains kerning information. When the KERNDATA= option is used during font creation, the data that is contained in the kern data set is applied and stored with the font.

Alias KERN=

Restriction If you specify kerning for a double-byte character set that is created by using the option CODELEN=2, then the KERNDATA= option is ignored.

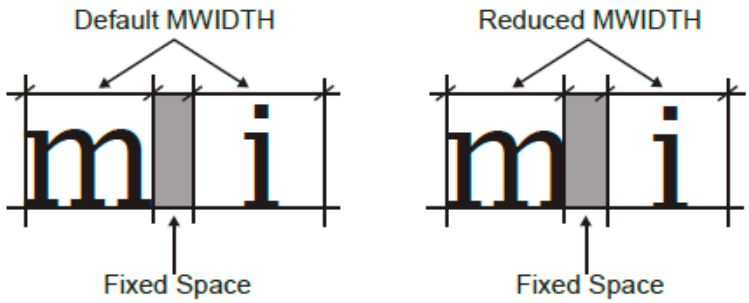
See "Overview: The Kern Data Set" on page 1210

MWIDTH=*character-width*

specifies the width of a character in a uniform font, where *character-width* is the number of font units. The MWIDTH= option is valid when you specify uniform spacing by using the UNIFORM option or when you specify CHARSPACETYPE=UNIFORM. If you omit the MWIDTH= option, the default is the width of the widest character in the font (usually the letter m).

The MWIDTH= option is typically used to tighten the spacing between characters. To do this, specify a smaller value for *character-width*. Figure 39.143 on page 1199 shows the effect of decreasing the space that is allowed for equally spaced characters.

Figure 39.4 *Using the MWIDTH= Option to Modify Spacing*



See CHARSPACETYPE= on page 1197 and UNIFORM on page 1201

NODISPLAY

specifies that the GFONT procedure is not to display the font that it is creating.

Alias ND

NOKEYMAP

specifies that the current key map is ignored when you create and then use the font that is created. The character codes that you enter are not mapped in any way before being displayed. As a result, the created font is *never* affected by any setting of the KEYMAP= graphics option.

By default, the NOKEYMAP option is *not* used. In that case, when you build a font, the current key map is applied to the values in the CHAR variable.

However, your current key map might not be symmetrical; two or more input character codes might be mapped to the same output character. For example, if A is mapped to B, then both A and B map to B, but nothing maps to A. In this

case, more than one code in your input data set can map to the same character in the resulting font. For example, if A and B are values of CHAR, both map to B. If this happens, a message that indicates the problem characters is displayed in the SAS log. To solve this problem, do one of the following tasks:

- change the character code of one of the characters
- eliminate one of the characters
- use the NOKEYMAP option

The NOKEYMAP option works correctly only if the end user's host or controller encoding is the same as the encoding used to create the input data set.

See NOKEYMAP on page 1195 (for displaying fonts)

CAUTION **Fonts created with the NOKEYMAP option are never affected by any setting of the KEYMAP= graphics option.**

RESOL=1 | 2 | 3 | 4

controls the resolution of the fonts by specifying the number of bytes (1 through 4) for storing coordinates in the font. The GFONT procedure provides three resolution levels (RESOL=3 produces the same resolution level as RESOL=4). By default, RESOL=1.

The higher the number, the closer together the points that define the character can be spaced. A high value specifies a denser set of points for each character so that the characters approximate smooth curved lines at very large sizes. RESOL=2 works well for most applications; RESOL=3 or 4 might be too dense to be practical.

The table below shows the resolution number and the maximum number of distinct points that can be defined horizontally or vertically.

Resolution	Number of Distinct Points
2	32,766
3	2,147,483,646
4	2,147,483,646

Alias R=

Example “Example 2: Creating Figures for a Symbol Font” on page 1216

ROMHEX

specifies that hexadecimal values are displayed automatically below the font characters when the GFONT procedure displays the font. If you use the ROMHEX option for a font that you create, you can later use the NOROMHEX option to suppress display of the hexadecimal values.

Alias HEX

See SHOWROMAN on page 1201, ROMHEX on page 1196 (for displaying fonts), and NOROMHEX on page 1195

SHOWROMAN
specifies that character codes are displayed automatically below the font characters when the GFONT procedure displays the font. If you use the SHOWROMAN option for a font that you create, you can later use the NOROMAN option to suppress display of the character codes.

Alias SR

See ROMHEX on page 1196, SHOWROMAN on page 1197 (for displaying fonts), and NOROMAN on page 1195

SPACEDATA=*space-data-set*
specifies the SAS data set that contains font spacing information. When you use the SPACEDATA= option during font creation, the data contained in the space data set is applied to the font and stored with it.

Alias SPACE=

Restriction If you specify the SPACEDATA= option for a double-byte character set that is created by using the option CODELEN=2, then the SPACEDATA= option is ignored.

See "Overview: The Space Data Set" on page 1212

UNIFORM
specifies that characters are spaced uniformly rather than proportionately. Using the UNIFORM option is the same as specifying CHARSPACETYPE=UNIFORM.

Alias U

See CHARSPACETYPE= on page 1197 and MWIDTH= on page 1199

Usage: GFONT Procedure

Font Data Sets and the GFONT Procedure

To create a font, you must create a data set that contains font information. Typically, you use a DATA step to create a SAS data set from which the GFONT procedure generates the font. The data set is referred to as the *font data set*, and you can specify it with the DATA= argument.

To produce the font, invoke the GFONT procedure and specify the data set that contains the font information. In addition, you can include options to modify the design and appearance of the font. For example, the following statement uses the data set FONTDATA to generate the font MYLOG:

```
proc gfont data=fontdata name=mylogo;
```

For a demonstration of the font creation process, see "Example 2: Creating Figures for a Symbol Font" on page 1216.

The GFONT procedure uses three types of data sets: the *font data set*, the *kern data set*, and the *space data set*. Each type of data set must contain certain variables and meet certain requirements. The following sections explain what each data set contains, how it is built, and what the requirements of the variables are.

The Font Data Set

Overview: The Font Data Set

The font data set consists of a series of observations that describe the font. The variables in the font data set must be assigned certain names and types. "Font Data Set Variables" on page 1203 describes the variables that are included in the font data set. These variables are described in "Font Data Set Variables" on page 1203 in more detail.

The font data set must include the horizontal and vertical coordinate values. It also includes line segment numbers that the GFONT procedure uses to generate each character. In addition, each observation must include a character code that is associated with the font character and is used to specify the font character in a text string. The font data set also determines whether the font is stroked or polygon. A font data set that generates a polygon font produces an outline font by default. You can use the FILLED option with the same data set to generate a filled font.

Specify the font data set with the DATA= argument, for example:

```
data mylib.fontdata;
 proc gfont data=fontdata name=mylogo;
run;
```

Table 39.1 *Table of Font Data Set Variables*

Variable	Description	Type	Length	Valid Values	With Stroked Fonts	With Polygon Fonts
CHAR	The character code associated with the font character	Character	1 or 2	Keyboard characters or hexadecimal values	Required	Required
LP	The type of line segment being drawn, either a line or a polygon	Character	1	L or P	Optional	Required
PTYPE	The type of data in the observation	Character	1	V or C or W	Optional	Optional

Variable	Description	Type	Length	Valid Values	With Stroked Fonts	With Polygon Fonts
SEGMENT	The number of the line segment or polygon being drawn	Numeric		Number	Required	Required
X	The horizontal coordinate	Numeric		Number	Required	Required
Y	The vertical coordinate	Numeric		Number	Required	Required

Font Data Set Variables

CHAR

provides a code for the character or figure that you are creating. CHAR is a character variable with a length of 1 or 2. CHAR is required for all fonts.

CAUTION
Using reserved or undefined hexadecimal codes as CHAR values might require the use of the NOKEYMAP option.

The CHAR variable takes any character as its value, including keyboard characters and hexadecimal values from '00'x to 'FF'x. (If you use hexadecimal values as CHAR values, your font might not work correctly under a key map that is different from the one under which the font was created. Positions that are not defined in one key map might be defined in another.)

When you specify the code for the character in a text string, the associated font character is drawn. For example, if you create a Roman alphabet font, typically the characters that you specify for CHAR are keyboard characters that match the character in the font. All of the observations that build the letter A have a CHAR value of A. When you specify 'A' in a text string, this creates an A in the output.

However, If you build a symbol font, the symbols might not have corresponding keyboard characters. In that case, you select a character or hexadecimal value to represent each symbol in the font and assign it to CHAR. For example, in the Special font, the letter G is assigned as the code for the fleur-de-lis symbol. When you specify the code in a text string, the associated symbol is displayed.

Note: If the CODELEN= option is set to 2, the values for CHAR represent two characters, such as AA, or a four-digit hexadecimal value, such as '00A5'x.

LP

tells the GFONT procedure whether the coordinates of each segment form a line or a polygon. LP is a character variable with a length of 1. The LP variable is required for polygon fonts but is optional for stroked fonts. You can assign the LP variable either of the following values:

L lines

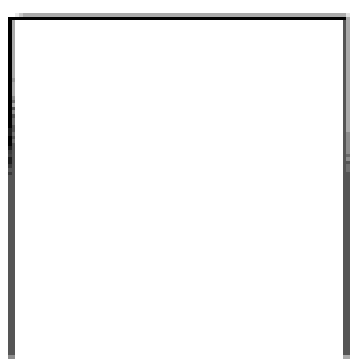
P polygons

Every group of line segments with an LP value of P is designated as a polygon. If the observations do not draw a completely closed figure, the program closes the figure automatically. For example, the following observations do not contain an LP variable. Observations that do not contain an LP variable create a shape like the one in Figure 39.144 on page 1204.

Table 39.2 *Observations to Create the Following Figure*

OBS	CHAR	SEG	X	Y
1	b	1	1	1
2	b	1	1	3
3	b	1	3	3
4	b	1	3	1

Figure 39.5 *LP Value of L*

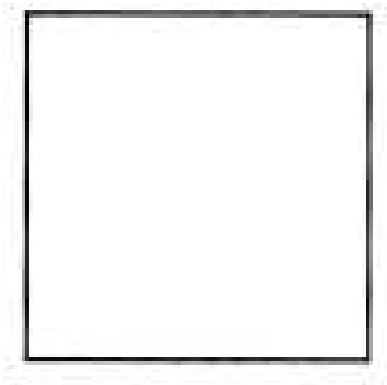


An LP variable with a value of P for all observations added to the data set produces a complete box like the one in Figure 39.145 on page 1205.

Table 39.3 *Observations to Create the Following Figure*

OBS	CHAR	SEG	X	Y	LP
1	b	1	1	1	P
2	b	1	1	3	P
3	b	1	3	3	P
4	b	1	3	1	P

Figure 39.6 *LP Value of P*

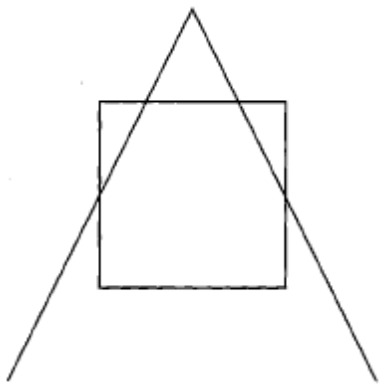


The LP variable enables you to mix lines and polygons. These observations create the figure consisting of a polygon and a line segment as shown in Figure 39.146 on page 1205.

Table 39.4 *Observations to Create the Following Figure*

OBS	CHAR	SEG	X	Y	LP
1	b	1	1	1	P
2	b	1	1	3	P
3	b	1	3	3	P
4	b	1	3	1	P
5	b	2	0	0	L
6	b	2	2	4	L
7	b	2	4	0	L

Figure 39.7 *Mixing LP Values of Line and Polygon*



PTYPE

tells the GFONT procedure what type of data is in the observation. PTYPE is a character variable of length 1 that is optional. For each observation, the PTYPE

variable assigns a characteristic to the point that is determined by the X and Y values. You can assign the PTYPE variable to any of these values:

V

normal point in the line segment

Note If a PTYPE variable is not specified, then all points are assumed to be V-type points.

C

center of a circular arc joining two V points

Restriction Arcs are limited to 106 degrees or less.

Example Figure 39.147 on page 1207

W

width value for CHARSPACETYPE=DATA. An observation with a PTYPE value of W must always be the first observation for a character. The observation gives the minimum and maximum X values for the character. The Y variable observation contains the maximum X value. Usually, these values include a little extra space for intercharacter spacing.

Restriction Use a PTYPE of W only if you have specified CHARSPACETYPE=DATA. Otherwise, the points are ignored.

See CHARSPACETYPE= on page 1197

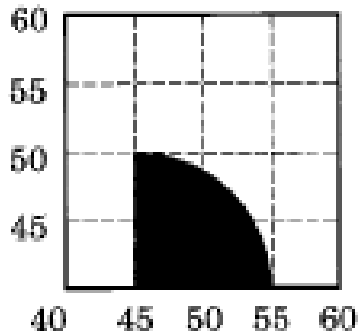
The following observations illustrate how the PTYPE variable is used to draw an arc similar to Figure 39.147 on page 1207.

OBS	CHAR	SEG	X	Y	LP	PTYPE	Comment
1	a	1	40	60	P	W	Define width of character as 20 font units, which is the number of units from left margin, 40, to right margin, 60
2	a	1	45	40	P	V	Start line segment at position 45,40
3	a	1	45	50	P	V	Draw a line to position 45,50, which is start point of arc
4	a	1	45	40	P	C	Draw an arc whose center is at 45,40

OBS	CHAR	SEG	X	Y	LP	PTYPE	Comment
5	a	1	55	40	P	V	Finish drawing the arc at 55,40

After the figure was generated, a grid was overlaid on it to show the location of the points.

Figure 39.8 *Using the PTYPE Variable to Create an Arc*



Note the following:

- Three observations are required to draw an arc: observation 3 and observation 5 denote the start point and the end point of the arc. Observation 4 locates the center of the arc.
- The figure is closed because the line segments have an LP value of P (polygon).
- The font that contains the figure of the arc was created with a similar PROC GFONT statement:

```
proc gfont data=arc name=arcfig charspacetype=data filled;
```

  The CHARSPACETYPE= DATA option specifies that the first observation sets the width of the character. The FILLED option fills the area of the arc.

SEGMENT

numbers the line segments that compose a character or symbol. SEGMENT is a required numeric variable. All observations for a given line segment have the same segment number. To start a new line segment, change the segment number.

When the GFONT procedure draws a stroked character with more than one line segment (for example, the letter E), or a polygon character with a hole (for example, the letter A), it needs to know when one line stops and where the next line begins. To indicate when one line stops and where the next line begins you can do either of the following:

- Change the segment number when a new line begins. If the value of LP is L (line), a change in segment number tells the GFONT procedure not to connect the last point in line segment 1 and the first point in line segment 2.
  - The last point in line segment 1 ends the line.
  - The first point in line segment 2 starts a new line.

If the value of LP is P (polygon), a change in segment numbers causes the following:

- The last point in line segment 1 joins to the first point in line segment 1, which closes the polygon.
- A new polygon starts. If the value of CHAR has not changed, the new polygon is part of the same character.

  Use this method for characters that consist of two polygons such as a question mark. This method is preferred, unless you are creating a polygon character with a hole in it.

- Keep the same segment number for all lines. Insert an observation with missing values for X and Y. Insert the new observation between the observation that marks the end of the first line, and the observation that begins the next line. For example, if you are drawing the letter O, insert an observation with a missing value between the line that draws the outer circle and the beginning of the line that draws the inner circle.

The first method is preferred, unless you are creating a polygon character with a hole in it. The second method is preferred when creating a polygon with a character with a hole in it. In this case, you should separate the lines with a missing value and keep the same segment numbers.

If you use separate line segments when you create a polygon with a hole, the results can be unpredictable. For example, the observations from a data set called BOXES were used to draw the hollow square in Figure 39.148 on page 1209.

OBS	CHAR	SEG	X	Y	LP
1	b	1	1	1	P
2	b	1	1	3	P
3	b	1	3	3	P
4	b	1	3	1	P
5	b	1	-	-	P
6	b	1	0	0	P
7	b	1	0	4	P
8	b	1	4	4	P
9	b	1	4	0	P

Note the observation 5 has missing values for X and Y. This separates the observations that draw the inner box from those that draw the outer box. The segment number is the same for all the observations.

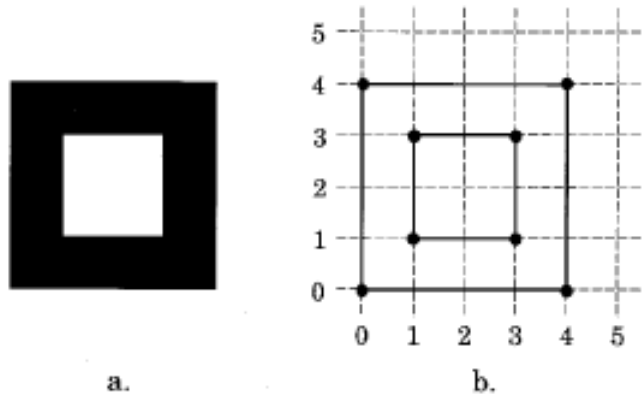
Figure 39.148 on page 1209 was created with a similar GFONT statement:

```
proc gfont data=boxes name=boxes filled;
```

The FILLED option is included, and only the space between the two squares is filled.

The data points that form the figure are laid out on a grid shown next to the square.

Figure 39.9 *Drawing Nested Polygons*



X and Y

specify the horizontal and vertical coordinates of the points for each character. Their values describe the position of the points on the character. These variables have the following characteristics:

- They must be numeric.
- They must be named X and Y for the horizontal and vertical coordinates, respectively.
- The values specified by them can be in any range.
- They both must describe the character in the same scale or font units.
- Vertical (Y) coordinates for all characters should be defined on the same baseline.

Interaction When you specify PTYPE=W, both X and Y contain horizontal coordinate values.

Creating a Font Data Set

Create a font data set by digitizing the shape of the characters or figures either manually or with special digitizing equipment. To create a font data set by digitizing the characters manually:

1. Determine the coordinate points for each line segment by drawing the characters on a grid.
2. Lay out the observations for each character. Each observation describes a move from one point to another along a line segment. For each line segment, enter the coordinate points in the order in which they are drawn. For a stroked font, when you start a new line segment, change the segment number. For a polygon font, when you start a new polygon, change the line segment number.

   If the polygon has a hole in it, as in the letter O, keep the line segment number and separate the lines with a missing value. Use the same value for CHAR for all of the observations that describe one character.

3 Create a SAS data set that contains the variables CHAR, SEGMENT, X, and Y, and read in the data for each observation. Include the variables LP and PTYPE if necessary.

4 Sort the data set by CHAR and SEGMENT.

5 Assign the font data set with the DATA= argument.

This process is illustrated in "Example 2: Creating Figures for a Symbol Font" on page 1216.

The Kern Data Set

Overview: The Kern Data Set

The kern data set consists of observations that specify how much space to add or remove between any two characters when they appear in combination. This process, called *kerning*, increases or decreases space between the characters. Kerning usually is applied to certain pairs of characters that have too much space between them. Reducing the space between characters might result in part of one character extending over the body of the next. Examples of some combinations that should be kerned are AT, AV, AW, TA, VA, and WA.

You can apply kerning to the intercharacter spacing that you specify with the CHARSPACETYPE= option (except for uniform fonts). Assign the kern data set with the KERNDATA= option.

Kern Data Set Variables

The kern data set must contain these variables:

CHAR1
: specifies the first character in the pair to be kerned. CHAR1 is a character variable with a length of 1.

CHAR2
: specifies the second character in the pair to be kerned. CHAR2 is a character variable with a length of 1.

XADJ
: specifies the amount of space to add or remove between the two characters. XADJ is a numeric variable that uses the same font units as the font data set. The value of XADJ specifies the horizontal adjustment to be applied to CHAR2 whenever CHAR1 is followed immediately by CHAR2. Negative numbers decrease the spacing, and positive numbers increase the spacing.

Creating a Kern Data Set

Each observation in a kern data set names the pair of characters to be kerned and the amount of space to be added or deleted between them. To create a kern data set, follow these steps:

1. Select the pairs of characters to be kerned, and specify the space adjustment (in font units) for each pair as a positive number (more space) or negative number (less space).
2. Create a SAS data set that contains the variables CHAR1, CHAR2, and XADJ. Define one observation for each pair of characters and the corresponding space adjustment, for example:

```
data kern1;
  input char1 $ char2 $ xadj;
  datalines;
A T -4
D A -3
T A -4
;
```

3. Assign the kern data set with the KERNDATA= option as follows:

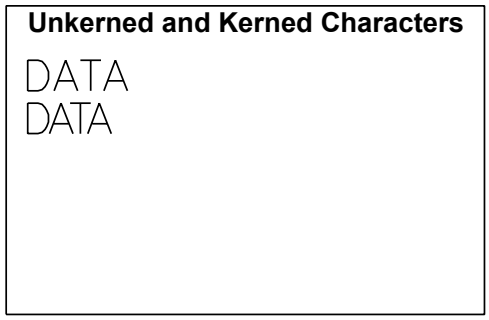
```
proc gfont data=fontdata
           name=font2
           charspacetype=data
           kerndata=kern1
           nodisplay;
run;
```

"Creating a Kern Data Set" on page 1211 illustrates how to use the KERNDATA= option to create a font in which the space between specified pairs of letters is reduced. The characters A, D, and T are shown as the word DATA. The first line uses the unkerned font, FONT1, and the second line uses the kerned font, FONT2. Note that the characters in FONT2 are spaced more closely than the characters in FONT1.

The following TITLE statements specify the kerned and unkerned fonts and are used with the GSLIDE procedure to produce Figure 39.149 on page 1211.

```
title2 lspace=6 f=font1 h=10 j=l "DATA";
title3 lspace=4 f=font2 h=10 j=l "DATA";
```

Figure 39.10 *Comparison of Kerned and Unkerned Text*



The Space Data Set

Overview: The Space Data Set

As the height (point size) of a font increases, less space is required between letters in relation to their height. If the height decreases, more space might be needed. The space data set tells the GFONT procedure how much to increase or decrease the intercharacter spacing for a given point size. Like kerning, spacing is added to or subtracted from the intercharacter spacing that is specified by the CHARSPACETYPE= option. Spacing is applied uniformly to all characters.

Values that are specified in the space data set are added to the normal intercharacter spacing and any kerning data. Normal intercharacter spacing is determined by the CHARSPACETYPE= option.

Space Data Set Variables

The space data set must contain these variables:

SIZE
: specifies the point size of the font. SIZE is a numeric variable.

ADJ
: specifies the spacing adjustment for the point size in hundredths (1/100) of a point. (A point is equal to 1/72 of an inch.) ADJ is a numeric variable. Positive values for ADJ increase the space between characters. Negative values for ADJ reduce the space between characters.

Creating a Space Data Set

Each observation in a space data set specifies a point size (SIZE) and the amount of space (ADJ) to be added or subtracted between characters when a font of that point size is requested. When you specify a point size that is not in the space data set, the adjustment for the next smaller size is used. To create a space data set, follow these steps:

1. Determine the amount of adjustment that is required for typical point sizes.
2. Create a data set that contains the variables SIZE and ADJ. Create one observation for each point size and corresponding space adjustment as follows:

```
data space1;
  input size adj;
  datalines;
 6   40
```

```
12    0
18  -40
24  -90
30 -150
36 -300
42 -620
;
```

3 Assign the space data set with the SPACEDATA= option as follows:

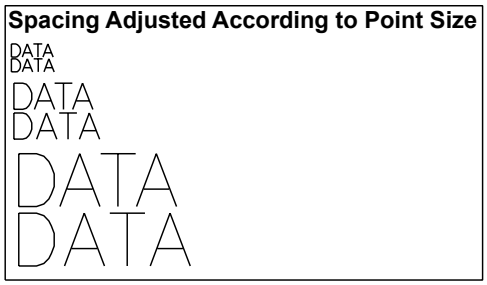
```
proc gfont data=fontdata
           name=font3
           charspacetype=data
           spacedata=space1
           nodisplay;
run;
```

Figure 39.150 on page 1213 illustrates how to use the SPACEDATA= option to create a font in which intercharacter spacing is adjusted according to the height of the characters. The characters A, D, and T are shown as the word DATA. Each pair of lines displays the word DATA and at the same size uses first the font with spacing adjustment (FONT3) and then the original font (FONT1). Note that as the size of the characters increases, the space between them decreases.

The following TITLE statements are used with the GSLIDE procedure to produce Figure 39.150 on page 1213:

```
title2;
title3 f=font3 h=.25in j=l "DATA"; /* 18 points */
title4 f=font1 h=.25in j=l "DATA";
title5;
title6 f=font3 h=.50in j=l "DATA"; /* 36 points */
title7 f=font1 h=.50in j=l "DATA";
title8;
title9 f=font3 h=1.0in j=l "DATA"; /* 72 points */
title10 f=font1 h=1.0in j=l "DATA";
```

Figure 39.11 *Comparison of Text with and without Spacing Adjustments*



Examples: GFONT Procedure

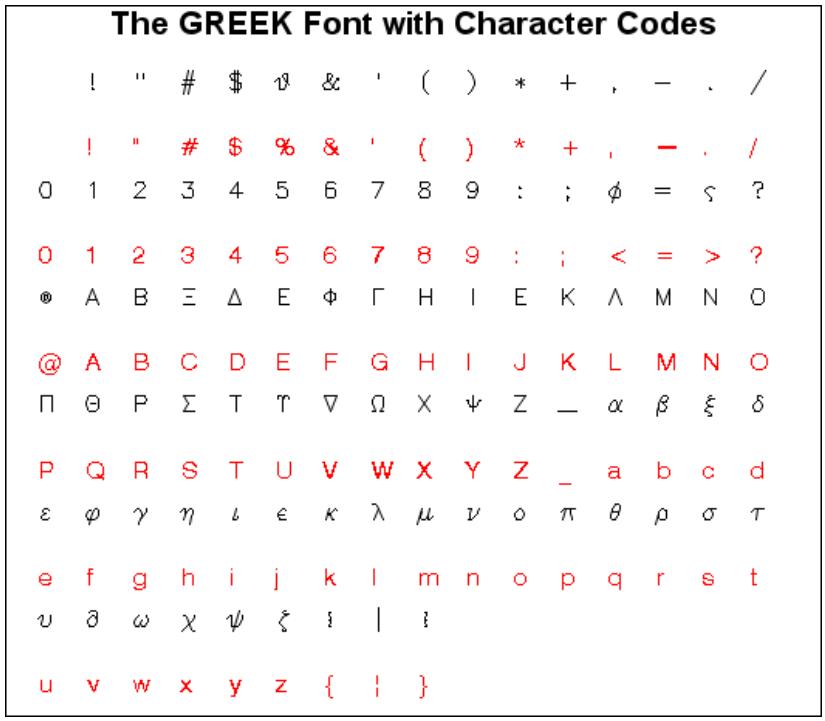
Example 1: Displaying Fonts with Character Codes

Features: GFONT statement options
- HEIGHT=
- NOBUILD
- ROMCOL=
- ROMFONT=
- ROMHT=
- SHOWROMAN

Sample library member: GFODISFO

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This illustrates the SHOWROMAN option, which displays the character codes that are associated with the font characters that are being displayed. This display shows which keyboard character you enter to produce the Greek character that you want displayed. The example also illustrates how to modify the appearance of both the font characters, and the character codes.



Program

```
goptions border hsize=5.5in vsize=4.8in device=png;

title height=2.5 font="Arial/bold" "The GREEK Font with Character
Codes";

proc gfont name=greek
           nobuild
           height=1.8
           romcol=red
           romfont=swiss
           romht=1.8
           showroman;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions border hsize=5.5in vsize=4.8in device=png;
```

Define the title.

```
title height=2.5 font="Arial/bold" "The GREEK Font with Character
Codes";
```

Display the GREEK font with character codes. NOBUILD indicates that the font specified in the NAME= argument is an existing font. HEIGHT= specifies the height of the Greek characters. ROMCOL=, ROMFONT=, and ROMHT= assign the color,

type style, and height of the character codes. SHOWROMAN displays the character codes.

```
proc gfont name=greek
           nobuild
           height=1.8
           romcol=red
           romfont=swiss
           romht=1.8
           showroman;
run;
quit;
```

Example 2: Creating Figures for a Symbol Font

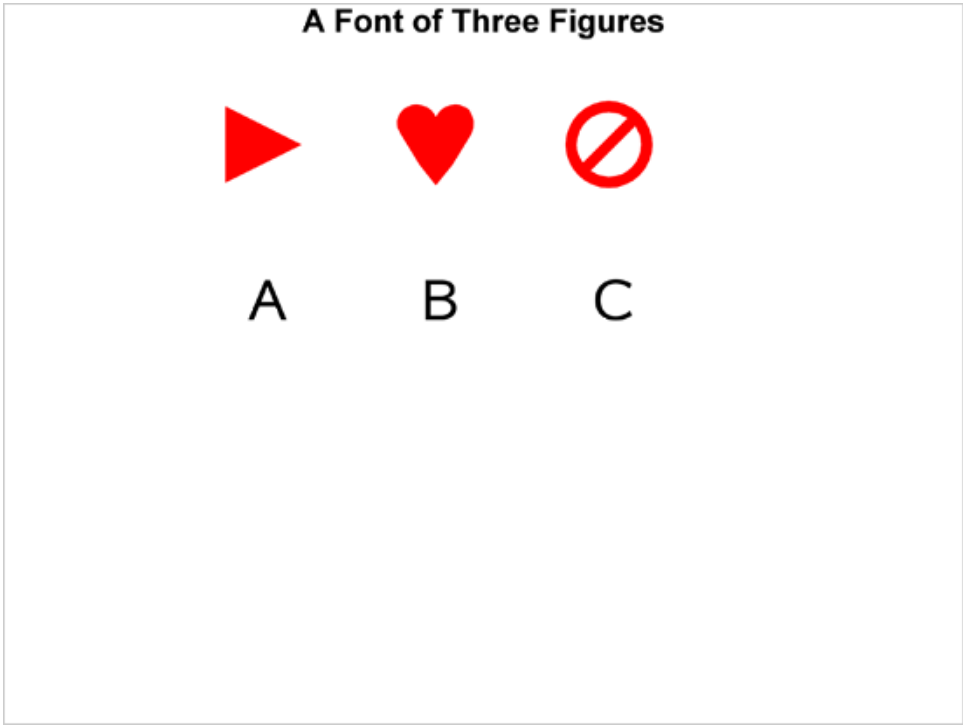
Features: GFONT statement options
- CTEXT=
- DATA=
- FILLED
- NAME=
- RESOL=

LIBNAME statement

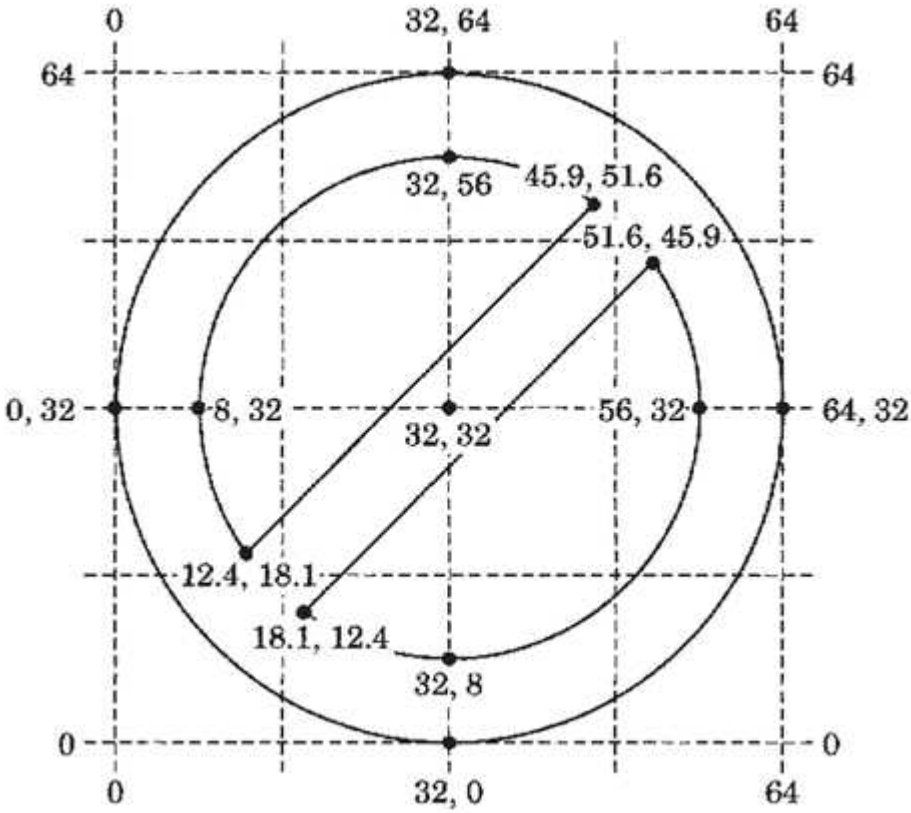
Sample library member: GFOCRFIG

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

Create three simple figures for a symbol font. Each figure is laid out on a grid that is 64 font units square. The third figure is a circle with a slash through it. Output 39.26 on page 1217 shows the figure and some of its coordinate points laid out on a grid.



Output 39.1 *Diagram of Circle with Slash Figure*



Program

```
LIBNAME gfont0 ".";
goptions reset=all border gunit=pct
   device=png;

data figures;
   input char $ ptype $ x y segment lp $;
   datalines;
A    W    0    64   0    P  /* triangle pointing right */
A    V    4     4   1    P
A    V   60    32   1    P
A    V    4    60   1    P
A    V    4     4   1    P

B    W    0    64   0    P  /* heart */
B    V   32     2   1    P
B    V   44    17   1    P
B    V   58    40   1    P
B    C   46    47   1    P
B    V   56    58   1    P
B    C   46    47   1    P
B    V   32    52   1    P
B    C   18    47   1    P
B    V    8    58   1    P
B    C   18    47   1    P
B    V    6    40   1    P
B    V   20    17   1    P
B    V   32     2   1    P

C    W    0    64   0    P  /* circle with slash */
C    V   32    64   1    P
C    C   32    32   1    P
C    V   64    32   1    P
C    C   32    32   1    P
C    V   32     0   1    P
C    C   32    32   1    P
C    V    0    32   1    P
C    C   32    32   1    P
C    V   32    64   1    P
C    V    .     .   1    P
C    V   12.4  18.1 1    P
C    C   32    32   1    P
C    V    8    32   1    P
C    C   32    32   1    P
C    V   32    56   1    P
C    C   32    32   1    P
C    V   45.9  51.6 1    P
C    V    .     .   1    P
C    V   51.6  45.9 1    P
C    C   32    32   1    P
C    V   56    32   1    P
C    C   32    32   1    P
C    V   32     8   1    P
C    C   32    32   1    P
C    V   18.1  12.4 1    P
;
```

```
title "A Font of Three Figures";

proc gfont data=figures
           name=figures
           filled
           height=.75in
           ctext=red
           showroman
           romht=.5in
           resol=2;
run;
quit;
```

Program Description

Assign the librefs and set the graphics environment. The LIBNAME statement associates the libref GFONT0 with the SAS library in which the font catalog is stored.

```
LIBNAME gfont0 ".";
goptions reset=all border gunit=pct
   device=png;
```

Create the font data set FIGURES for a triangle, a heart, and a circle with slash. The first figure, a right-pointing triangle that is assigned the character code A, is a polygon drawn with three straight lines.

```
data figures;
   input char $ ptype $ x y segment lp $;
   datalines;
A    W    0    64   0    P  /* triangle pointing right */
A    V    4     4   1    P
A    V   60    32   1    P
A    V    4    60   1    P
A    V    4     4   1    P
```

The second figure, a heart that is assigned the character code B, uses the PTYPE variable combination V-C-V to draw the arcs that make up the top of the heart. Each side requires two arcs. Because the arcs are continuous, the observation that marks the end of one arc is also the beginning of the next arc. The heart drawing begins at the bottom point and continues counterclockwise.

```
B    W    0    64   0    P  /* heart */
B    V   32     2   1    P
B    V   44    17   1    P
B    V   58    40   1    P
B    C   46    47   1    P
B    V   56    58   1    P
B    C   46    47   1    P
B    V   32    52   1    P
B    C   18    47   1    P
B    V    8    58   1    P
B    C   18    47   1    P
B    V    6    40   1    P
B    V   20    17   1    P
B    V   32     2   1    P
```

The third figure, a circle with a slash through it, assigned the character code C, consists of three polygons: a circle and two empty arcs. An observation with missing values separates the observations defining each of the three polygons. The outer circle is defined by the first group of observations. The empty arcs are drawn with three continuous arcs using the PTYPE variable pattern V-C-V-C-V-C-V. The straight line that closes the arc is drawn automatically by the GFONT procedure in order to complete the polygon. Because all the polygons are part of one character, the continuous space that they define is filled.

```
C    W    0    64   0    P  /* circle with slash */
C    V   32    64   1    P
C    C   32    32   1    P
C    V   64    32   1    P
C    C   32    32   1    P
C    V   32     0   1    P
C    C   32    32   1    P
C    V    0    32   1    P
C    C   32    32   1    P
C    V   32    64   1    P
C    V    .     .   1    P
C    V   12.4  18.1 1    P
C    C   32    32   1    P
C    V    8    32   1    P
C    C   32    32   1    P
C    V   32    56   1    P
C    C   32    32   1    P
C    V   45.9  51.6 1    P
C    V    .     .   1    P
C    V   51.6  45.9 1    P
C    C   32    32   1    P
C    V   56    32   1    P
C    C   32    32   1    P
C    V   32     8   1    P
C    C   32    32   1    P
C    V   18.1  12.4 1    P
;
```

Define the title.

```
title "A Font of Three Figures";
```

Create and display the font FIGURES. DATA= argument names the input data set. The NAME= the font that the procedure creates. FILLED specifies a filled polygon. HEIGHT= font height. CTEXT=red the color of the figures. RESOL=2 improves the resolution of the lines.

```
proc gfont data=figures
           name=figures
           filled
           height=.75in
           ctext=red
           showroman
           romht=.5in
           resol=2;
run;
quit;
```

Chapter 40

GKPI Procedure

Overview: GKPI Procedure

About KPI Charts and the GKPI Procedure

The GKPI procedure creates graphical key performance indicator (KPI) charts. KPIs are metrics that help a business monitor its performance and measure its progress toward specific goals. The procedure produces five KPI chart types:

- slider (vertical or horizontal)
- bullet graph (vertical or horizontal)
- dial
- speedometer
- traffic light (vertical or horizontal)

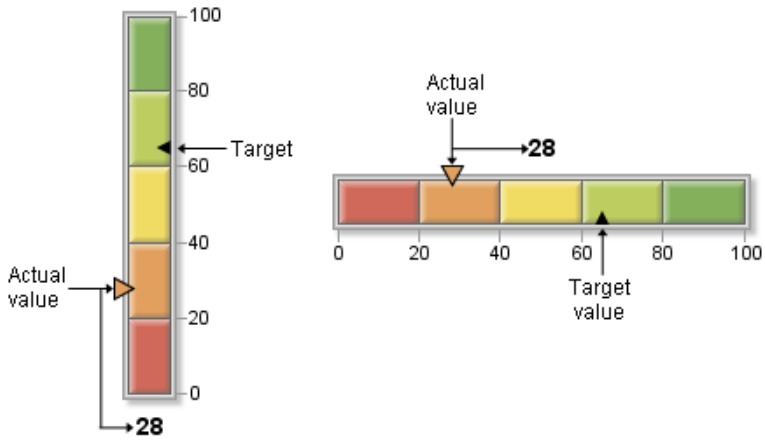
The GKPI procedure produces a KPI chart based on a series of qualitative range boundaries and an actual KPI value that you specify. If you specify a target value, the KPI chart also displays the target value. The procedure uses a set of default colors for the KPI chart, but you can specify your own colors.

Note: The only device supported for the GKPI procedure is JAVAIMG. If you do not specify DEVICE=JAVAIMG, then SAS/GRAPH sets the DEVICE option to JAVAIMG.

Note: To use output from the GKPI procedure in a dashboard generated with the GREPLAY procedure, you must first create a GRSEG containing the GKPI procedure output. You can use the IBACK="*gkpiImage*.png" option in the GOPTIONS statement with the GSLIDE or GANNO procedures to generate the GRSEG.

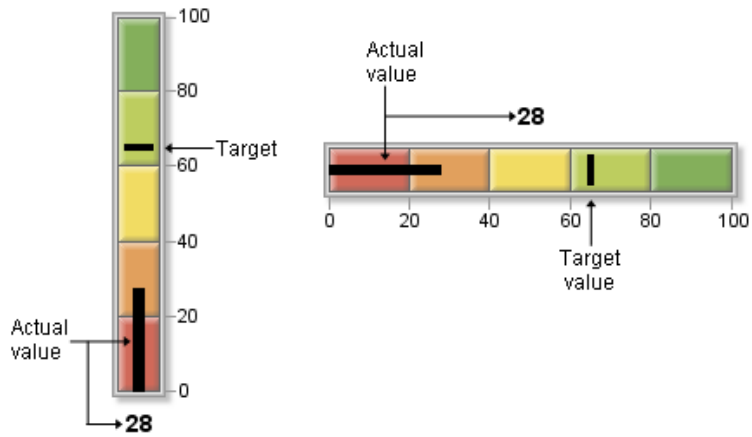
Slider KPI Charts

Slider KPI charts display a bar divided into qualitative ranges according to the boundary values that you specify. The actual value of the KPI is indicated with a triangle pointer the left (for a vertical slider) or on the top (for a horizontal slider). This actual value indicator is the same color as the range that contains the actual KPI value. The target value, if it is specified, is displayed as a smaller triangle on the right side (or bottom) of the slider.



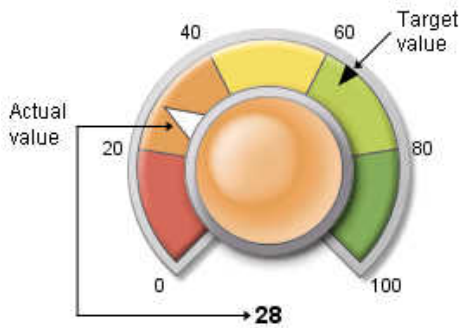
Bullet Graph KPI Charts

Bullet graphs display a bar divided into qualitative ranges according to the boundary values that you specify. The actual value of the KPI is indicated with a black line, or bullet, down the center of the graph. The target value, if it is specified, is displayed as a vertical line (in a horizontal bullet graph) or a horizontal line (in a vertical bullet graph) across the graph.



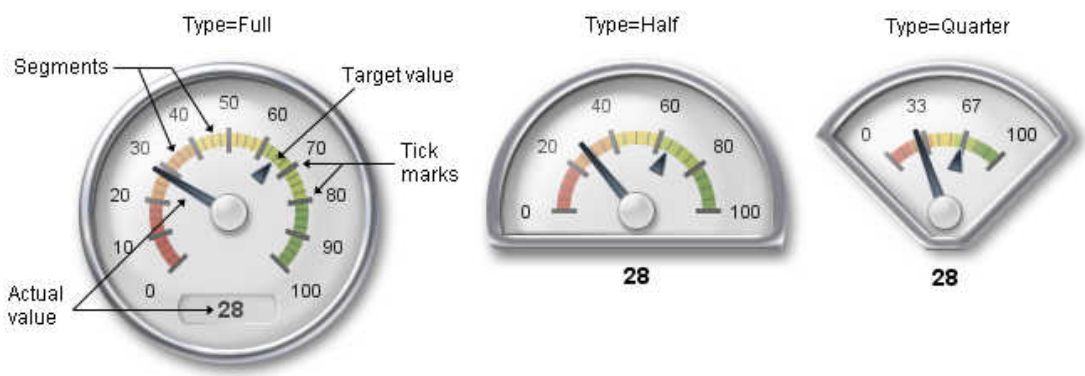
Dial KPI Charts

Dial KPI charts display a dial divided into qualitative ranges according to the boundary values. The actual value of the KPI is indicated with a large, white triangle pointer. The target value, if it is specified, is displayed as a small, black triangle. The center of the dial is the same color as the range that contains the actual KPI value.



Speedometer KPI Charts

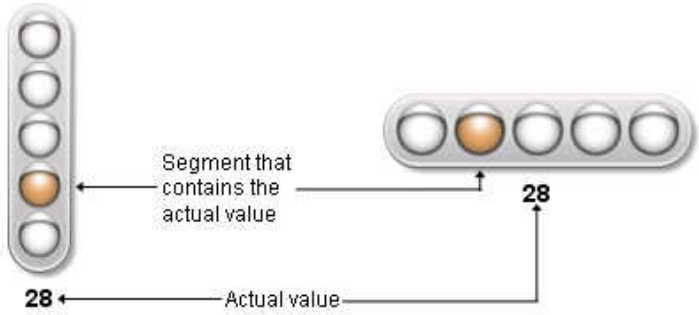
Speedometer KPI charts display a speedometer with the tick marks evenly spaced around the dial and colored ranges that correspond to the range boundaries that you specify. Speedometers can be displayed as a full speedometer, as a half speedometer, or as a quarter speedometer. The actual value of the KPI is indicated by a long pointer. The target value, if it is specified, is displayed as a small, black triangle.



In each display type, tick marks are evenly spaced but do not correspond to colored range boundaries. The numbered band in the full speedometer is always divided into ten sections (using 11 tick marks). The numbered band in the half speedometer is divided into five sections (six tick marks), and the quarter speedometer is divided into three sections (four tick marks).

Traffic Light KPI Charts

Traffic light KPI charts display a traffic light that contains one light for each qualitative range. The range that contains the actual value is displayed in color. The remaining ranges are gray. In other words, only one “light” is “turned on” at a time. Traffic lights do not display target values.

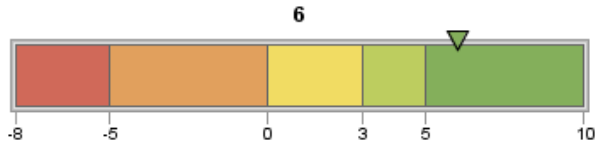


Concepts: GKPI Procedure

Specifying Qualitative Range Boundaries and Actual KPI Values

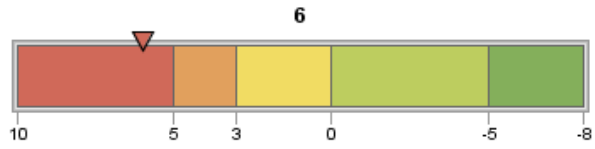
To generate a KPI chart, you must specify a list of range boundaries using the BOUNDS= option and an actual KPI value using the ACTUAL= option. The values can be positive numbers, negative numbers, or missing (ACTUAL=.), but the BOUNDS= list must be in either ascending or descending order and must contain at least two numbers (in order to define a single range). For example, the following code defines a horizontal slider with range boundaries in ascending order from –8 to 10 and an actual KPI value of 6:

```
goptions device=javaimg;
proc gkpi;
   hslider actual=6 bounds=(-8 -5 0 3 5 10);
run;
quit;
```



The boundaries can also be specified in descending order, for example:

```
hslider actual=6 bounds=(10 5 3 0 -5 -8)
```

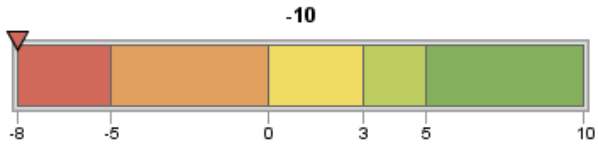


The order in which colors are applied is not affected by whether boundaries are specified in ascending or descending order. See "Defining Active and Inactive Color Lists" on page 1228 for information about controlling range colors.

The actual KPI value can fall outside of the highest or lowest boundaries, but the GKPI procedure treats such values as if they occur at the edge of the highest or lowest boundaries. For example, suppose the actual KPI value is –10, but the lowest boundary value is –8:

```
hslider actual=-10 bounds=(-8 -5 0 3 5 10)
```

PROC GKPI displays the actual KPI value indicator at –8.



If you specify a missing value for the actual KPI value (ACTUAL=.), then the GKPI procedure does not generate a KPI chart.

Controlling the Display of Boundary and Tick Mark Values

In some cases, there might not be enough space to display all of the boundary values or, for speedometers, tick mark values without some of the values colliding together. In these cases, the GKPI procedure typically drops some or all of the boundary or tick mark values, depending on the amount of space available, the font size being used, and the values that need to be displayed.

If the GKPI procedure drops values, you can try the following solutions:

- increasing the size of the KPI chart using the XPIXELS=/YPIXELS= or HSIZE=/VSIZE= options in the GOPTIONS statement
- reducing the size of the boundary value font using the BFONT= option
- applying a SAS format to the boundary values using the FORMAT= option

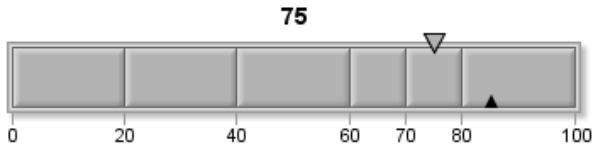
See "Example 3: Creating a Dial KPI Chart" on page 1243 for an example that uses the XPIXELS=, YPIXELS=, BFONT=, and FORMAT= options.

Controlling Qualitative Range Colors

Default Colors

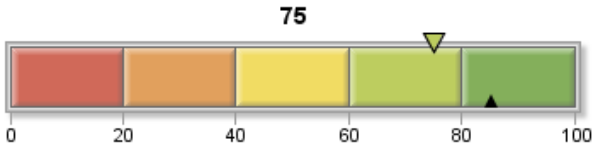
If you define only one range or more than five ranges, the GKPI procedure uses the same value of gray (hexadecimal RGB value B2B2B2) for all ranges.

Figure 40.1 GKPI Procedure Gray Scale Default



If you define two to five ranges, the GKPI procedure uses some or all of the colors shown in Figure 40.152 on page 1227 as the default colors.

Figure 40.2 GKPI Procedure Default Colors



For example, if you define only three ranges, the GKPI procedures uses the red, yellow, and green colors shown in Figure 40.152 on page 1227.

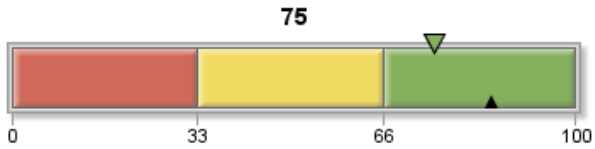


Table 40.90 on page 1227 lists the hexadecimal values for each of these default colors.

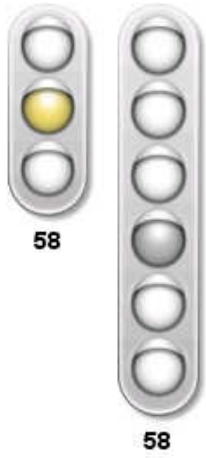
Table 40.1 Hexadecimal Values for GKPI Procedure Default Colors

Color	Hexadecimal RGB Value
Red	cxD06959
Orange	cxE1A05D
Yellow	cxF1DC63
Yellow-Green	cxBDCD5F
Green	cx84AF5B

The traffic light also uses the default colors, but it applies them slightly differently. If you define only one range, then the procedure displays the light for that range in gray. If you define two to five ranges, the traffic light uses the default color listed in Table 40.90 on page 1227 only for the light corresponding to the range that contains the actual KPI value. All other lights are gray. If you define more than five ranges, all lights are displayed in gray, but the light corresponding to the range that contains

the actual KPI value is displayed in dark gray. In other words, only one light in a traffic light is “turned on” at a time. All other lights are “turned off.”

Figure 40.3 *Traffic Light Default Colors*



Colors are applied in the same direction, regardless of whether the range boundaries are in ascending or descending order. Colors for horizontal sliders and bullet graphs are applied from left to right. Colors for traffic lights, vertical sliders, and vertical bullet graphs are applied from bottom to top. Colors for dials and speedometers are applied clockwise.

Defining Active and Inactive Color Lists

You can define two different color lists: a list of active colors and a list of inactive colors. The active color list is defined with the ACTIVECOLORS= option, and the inactive color list is defined with the COLORS= option. Neither option is required.

Each color in a list corresponds to a range in the KPI chart. That is, the first color is applied to the first range, the second color is applied only to the second range, and so on. A range is displayed in its active color if the actual KPI value falls in that range. All ranges that do not contain the actual KPI value are displayed in their inactive colors. If you do not specify an active color list, then the range that contains the actual KPI value is also displayed in its inactive color.

Note: The TRAFFICLIGHT statement supports both the COLORS= option and the ACTIVECOLORS= option. However, if both options are specified, the COLORS= option is ignored. All ranges that do not contain the actual KPI value appear gray.

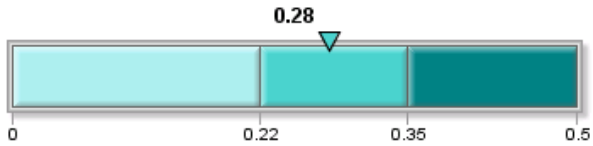
You can specify colors for the ranges using any of the color-naming schemes supported by SAS/GRAPH. See "Specifying Color Names" on page 1231 .

If you specify the COLORS= option, then you must specify a color for each range. That is, the number of entries in the COLORS= list must be one less than the number of entries in the BOUNDS= list. If you specify the ACTIVECOLORS= list, you do not have to specify a color for every range. See "Specifying Active Colors Only for Specific Ranges (Using Null Colors)" on page 1230 for more information.

Example: Specifying an Inactive Color List

The following example uses color names defined in the SAS registry to specify the inactive color list. The GKPI procedure uses these colors instead of the default colors shown in Figure 40.152 on page 1227 .

```
goptions reset=all device=javaimg;
proc gkpi mode=raised;
   hslider actual=0.28
   bounds=(0 .22 .35 .50) /
   colors=(PaleTurquoise MediumTurquoise Teal);
   run;
quit;
```



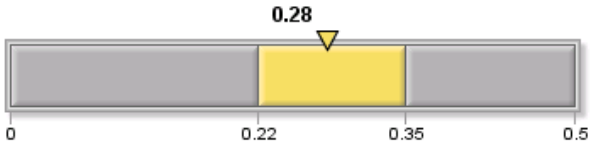
The actual KPI value falls into the second range, and there are no active colors specified, so the second color in the COLORS= list, MediumTurquoise, is used for the second range and for the actual KPI value indicator.

Example: Specifying an Active Color List

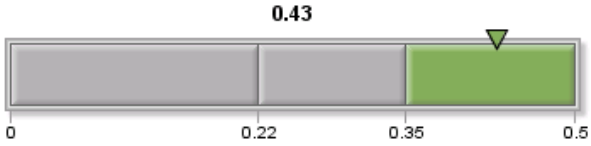
The following example defines the inactive colors for all of the ranges to be the same value of gray, cxB2B2B2. For the active colors, it specifies the default values for the red, yellow, and green colors listed in Table 40.90 on page 1227 .

```
goptions reset=all device=javaimg;
proc gkpi mode=raised;
   hslider actual=0.28
   bounds=(0 .22 .35 .50) /
   colors=(cxB2B2B2 cxB2B2B2 cxB2B2B2)
   activecolors=(cxD06959 cxF1DC63 cx84AF5B);
run;
quit;
```

The actual KPI value is 0.28, which falls into the second range, so the second color listed in the ACTIVECOLORS= color list, cxF1DC63, which is yellow, is used as the color for the second range instead of gray.



If the actual KPI value is changed to 0.43, then the third color in the ACTIVECOLORS= color list, cx84AF5B, which is green, is used for the third range instead of gray.



Specifying Active Colors Only for Specific Ranges (Using Null Colors)

If you specify a null color for a range in the ACTIVECOLORS= list, then either the default color or the color in the COLORS= list, if one is specified, is used for that range even if it contains the actual KPI value.

To specify a null color, you can specify `null` for the color or enter a comma-delimited list with no space between the commas. For example, if you have five ranges and you want red to be used only for the lowest and highest ranges, then you can specify the ACTIVECOLORS= list in either of the following forms:

```
activecolors=(red,null,null,null,red)
activecolors=(red,,,,red)
```

The default colors (or the color in the COLORS= list) are used for ranges two through four even if the actual KPI value falls into one of these ranges.

The ACTIVECOLORS= list does not have to specify a color for each range, but the one-to-one correspondence between the colors that are specified and the ranges is maintained. For example, supposed you define five ranges and you specify the following:

```
activecolors=(red green)
```

The GKPI procedure treats this specification as if you had entered the following:

```
activecolors=(red,green,null,null,null)
```

Red applies only to the first range, and green applies only to the second range. Default colors (or COLORS= colors) apply to all the other ranges.

Specifying Color Names

You can specify colors for the ranges using any of the color-naming schemes supported by SAS/GRAPH. For a complete description of these color-naming schemes, see Table 40.91 on page 1231 . The following table shows examples of each of the color naming schemes:

Table 40.2 *Examples of Specifying Colors*

Color-Naming Scheme	Example
RGB	`COLORS=(cx98FB98 cxDDA0DD cxFFDAB9 cxDB7093 cxB0E0E6)`
RGBA[1]	`COLORS=(a98FB9833 aDDA0DD66 aFFDAB999 aDB7093CC aB0E0E6FF)`
CMYK	`COLORS=("FF00FF00" "00FFFF00" "FFFFFF00")`
HLS	`COLORS=(H14055FF H0F060FF H0B485FF H07880FF)`
HSV	`COLORS=(V0F055FF v010FFFF v03BFFFF v12C55E8)`
Gray Scale	`COLORS=(GRAY4F GRAY6D GRAY8A GRAYC3)`
SAS Registry Colors	`COLORS=(palegreen plum peachpuff palevioletred powderblue)`
CNS Color Names	`COLORS=("very light purplish blue" "light vivid green" "medium strong yellow" "dark grayish green")`

1 RGBA color mode is supported by the ACTIVEX and ACTXIMG devices when the output is used in Microsoft applications.

Syntax: GKPI Procedure

Restriction: The only device supported for the GKPI procedure is JAVAIMG. If you do not specify DEVICE=JAVAIMG, then the procedure sets this option automatically.

Requirement: At least one DIAL, HSLIDER, HBULLET, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, or VSLIDER statement is required.

Supported Statements: FOOTNOTE, GOPTIONS (BORDER, VSIZE, HSIZE, XPIXEL, IBACK, CBACK, CTEXT, HTEXT, FTEXT only), TITLE

z/OS specifics: Starting with SAS 9.4M8, the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices are no longer supported in the z/OS operating environment. If you attempt to use one of these

devices on z/OS, an error results. Because the GKPI procedure supports only the JAVAIMG device, starting with SAS 9.4M8, the GKPI procedure no longer runs on z/OS.

Supports: Run-group processing

Tips: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, `ODS PDF; procedure-code; ODS PDF CLOSE;`). See Chapter 7, "Using Run-Group Processing," on page 75 for more information.

When the graphics output is sent to a graphics output file, this procedure uses ODS Graphics Indexing when necessary to avoid overwriting existing output files. See "About Filename Indexing" on page 137. To reset the current index value and overwrite output files that already exist, use the following statement:

```
ods graphics / reset=index;
```

See: Overview: GKPI Procedure on page 1222

PROC GKPI <MODE= BASIC | MODERN | RAISED>;
<DIAL ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
<HBULLET | BULLET ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
<VBULLET ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
<HSLIDER | SLIDER ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
<VSLIDER ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
<HTRAFFICLIGHT ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
<VTRAFFICLIGHT | TRAFFICLIGHT ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
<SPEEDOMETER ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
QUIT;

PROC GKPI Statement

Identifies the display mode.

Syntax

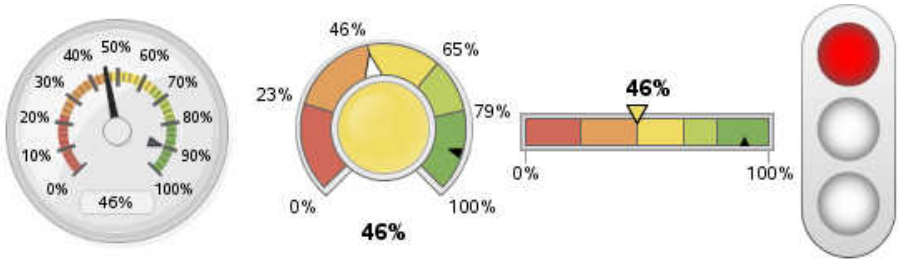
PROC GKPI <MODE= BASIC | MODERN | RAISED>;

Optional Argument

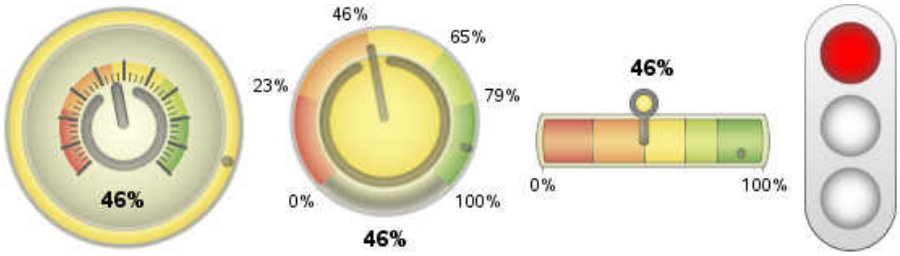
PROC GKPI statement options affect all graphs produced by the procedure.

MODE=BASIC | MODERN | RAISED
specifies the display mode of KPI chart images.

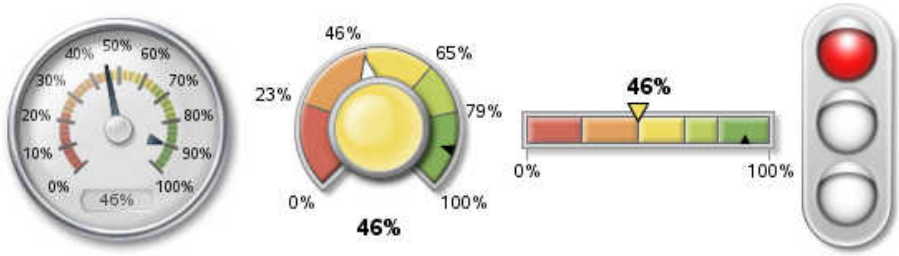
BASIC
creates a two-dimensional image.



MODERN
creates an image with a streamlined appearance.



RAISED
creates a three-dimensional version of the basic KPI.



Default BASIC

DIAL, HBULLET, HSLIDER, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, and VSLIDER Statement

Creates a chart in one of seven display types.

Requirement: The ACTUAL= value and the BOUNDS= list are required.

Syntax

<DIAL ACTUAL=*value* BOUNDS=*bounds-list* </ *options*>>;
<HBULLET | BULLET ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>
<VBULLET ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>

<HSLIDER | SLIDER ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>

<VSLIDER ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>

<HTRAFFICLIGHT ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>

<VTRAFFICLIGHT | TRAFFICLIGHT ACTUAL=*value* BOUNDS=*bounds-list* </ *options;*>>

<SPEEDOMETER ACTUAL=*value* BOUNDS=*bounds-list* </ *options*>>;

Summary of Optional Arguments

Appearance options

ACTIVECOLORS=(*color–1 color–2 ...color–n*)
: specifies the list of active colors for each qualitative range.

AFONT=(<F="*fontname*"</BOLD></ITALIC>> <C=*color*> <H=*text-height*<*units*>>)
: specifies the name, color, and text height for the font used for the actual KPI value label.

AVALUE

NOAVALUE
: specifies whether to display the actual KPI value label.

BFONT=(<F="*fontname*"</BOLD></ITALIC>> <C=*color*> <H=*text-height*<*units*>>)
: specifies the name, color, and text height of the font used for the boundary and tick mark values.

BVALUE

NOBVALUE
: specifies whether to display the boundary values.

COLORS=(*color–1 color–2 ...color–n*)
: specifies the list of inactive colors for each qualitative range.

FORMAT="*SAS-format*"
: specifies a SAS format for the boundary and actual values.

LABEL= "*string*"
: specifies a label for the graphic.

LFONT=(<F="*fontname*"</BOLD></ITALIC>> <C=*color*> <H=*text-height*<*units*>> <J=LEFT | RIGHT | CENTER>)
: specifies the name, color, and text height of the font to use for the label that is specified by the LABEL= option.

LOWBOUNDARY

NOLOWBOUNDARY
: specifies whether the KPI chart displays as if the KPI value falls in the lower range or the upper range when the actual KPI value falls directly on a range boundary.

TARGET= *data-value*
: specifies the numeric value of the target key performance indicator.

TYPE=FULL | HALF | QUARTER
: specifies the size of the display for speedometers.

Drill-down option

LINK="*url*"
: adds a drill-down link to the KPI chart.

Output file description options

DESCRIPTION= "*description*"
: specifies the description of the output.

NAME="*name*"
: specifies the name of the graphics output file.

Required Arguments

ACTUAL= *data-value*

specifies the actual value of the key performance indicator. The actual data value can fall outside the bounds specified with the BOUNDS= option, but the GKPI procedure will display the actual value indicator at the outermost edge of the KPI chart. If you specify a missing value (`ACTUAL=.`), then the GKPI procedure does not generate a KPI chart.

See "Specifying Qualitative Range Boundaries and Actual KPI Values" on page 1225

BOUNDS=(*bound–1 bound-2 ...bound-n*)

specifies a list of defined boundary values. The values can be negative or positive, but you must specify the list in either ascending or descending order. Separate each boundary value with a blank space.

See "Specifying Qualitative Range Boundaries and Actual KPI Values" on page 1225

Optional Arguments

You can specify as many options as needed and list them in any order.

ACTIVECOLORS=(*color–1 color–2 ...color–n*)

specifies the list of active colors for each qualitative range. This list is the colors that you want to use for each range when that range contains the actual KPI value. You do not have to specify a color for each range in the KPI chart. The default colors shown in Figure 40.152 on page 1227 (or the colors specified by the COLORS= option) are used for each range for which active colors are not specified. The number of entries in the ACTIVECOLORS= list cannot exceed the number of ranges that are defined. That is, the maximum number of active colors is one less than the number of entries in the BOUNDS= list. Separate each color with either a blank space or a comma.

See "Controlling Qualitative Range Colors " on page 1226

Example "Example 1: Using the Default Colors as the Active Colors" on page 1240

AFONT=(<F="*fontname*"</BOLD></ITALIC>> <C=*color*> <H=*text-height*<*units*>>)

specifies the name, color, and text height for the font used for the actual KPI value label.

Style reference Font attribute of the GraphLabelText element

See "Font Suboptions" on page 1239

Example `AFONT=(f="Comic Sans MS" c=red h=15pt)`

Example "Example 3: Creating a Dial KPI Chart" on page 1243

AVALUE | NOAVALUE
specifies whether to display the actual KPI value label.

Alias AVAL | NOAVAL

Default AVALUE

Example "Example 6: Creating a Traffic Light" on page 1247

BFONT=(<F="*fontname*"</BOLD></ITALIC>> <C=*color*> <H=*text-height*<*units*>>)
specifies the name, color, and text height of the font used for the boundary and tick mark values. If you increase the size of the font to the point where labels would collide, then the intermediate labels are not displayed. The GKPI procedure displays only the lowest and highest boundary labels.

Style reference Font attribute of the GraphValueText element

See "Font Suboptions" on page 1239

Example `BFONT=(font="Arial" color=H14055FF height=.25in)`

Example "Example 3: Creating a Dial KPI Chart" on page 1243

BVALUE | NOBVALUE
specifies whether to display the boundary values.

Alias BVAL | NOBVAL

Default BVALUE

COLORS=(*color–1 color–2 ...color–n*)
specifies the list of inactive colors for each qualitative range. This list is the colors that you want to be used for each range when that range does not contain the actual KPI value. You must specify a color for each range in the KPI chart. That is, the number of entries in the COLORS= list must be one less than the number of entries in the BOUNDS= list. Separate each color with either a blank space or a comma.

For all KPI charts except traffic lights, if you define two to five ranges, the GKPI procedure applies a default set of colors ranging from red to green. If you define only one or more than five ranges, the default color for all ranges is gray.

For traffic lights, the default color for all ranges is gray. This option is ignored by the TRAFFICLIGHT statement if the ACTIVECOLORS= option is specified.

See "Controlling Qualitative Range Colors " on page 1226

Examples "Example 1: Using the Default Colors as the Active Colors" on page 1240

"Example 2: Creating a Gray Scale Bullet Graph" on page 1242

"Example 5: Defining a Speedometer with Reversed Colors" on page 1246

"Example 6: Creating a Traffic Light" on page 1247

DESCRIPTION= "*description*"

specifies the description of the output. The maximum length for the description is 256 characters. The description does not appear on the graph. The default is "Key performance indicator".

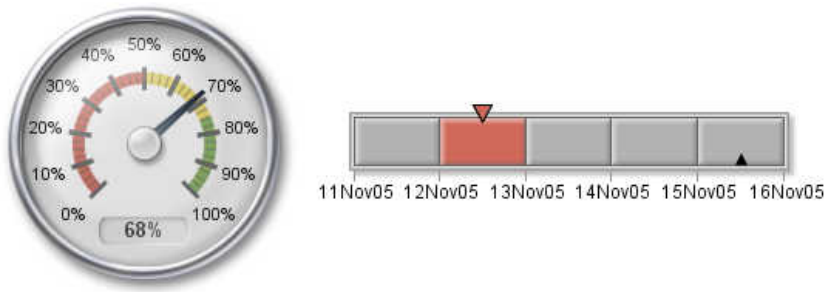
The descriptive text is shown in the "description" portion of each of the following:

- the Results window.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement.
- the chart description for web output. See "Chart Descriptions for Web Presentations" on page 251 for more information.

Alias DES=

FORMAT="*SAS-format*"

specifies a SAS format for the boundary and actual values. The default format is BEST. For example, you can use `format="percent8.0"` to display values as percentages or `format="datetime7."` to display SAS datetime values in the format **ddmmmyy**. You can also specify a user-defined format.



See "SAS Formats Supported in JAVAIMG Output" on page 1679

Examples "Example 3: Creating a Dial KPI Chart" on page 1243

"Example 4: Defining a Speedometer" on page 1244

LABEL= "*string*"

specifies a label for the graphic. The label is displayed at the top of graph, beneath the title, if a title is specified.

Note By default, labels are displayed at the top center of the graphics output area, and the KPI chart is displayed in the center of the output area. To reduce the space between labels and the KPI chart, reduce the size of the graphics output area by specifying the XPIXELS=/YPIXELS= or HSIZE=/VSIZE= options in the GOPTIONS statement. See "The Graphics Output and Device Display Areas" on page 80 for more information.

Examples "Example 4: Defining a Speedometer" on page 1244

"Example 5: Defining a Speedometer with Reversed Colors" on page 1246

"Example 6: Creating a Traffic Light" on page 1247

LFONT=(<F="*fontname*"</BOLD></ITALIC>> <C=*color*> <H=*text-height<units>*> <J=LEFT | RIGHT | CENTER>)

specifies the name, color, and text height of the font to use for the label that is specified by the LABEL= option. You can also specify whether the label is left-justified, centered, or right-justified within the graphics output area.

Style reference Font attribute of the GraphLabelText element

See "LABEL= "*string*"" on page 1237 and "Font Suboptions" on page 1239

Example
```
LFONT=(f="Albany AMT/italic" c=cornflowerblue height=.25cm j=right)
```

Example "Example 4: Defining a Speedometer" on page 1244

LINK="*url*"

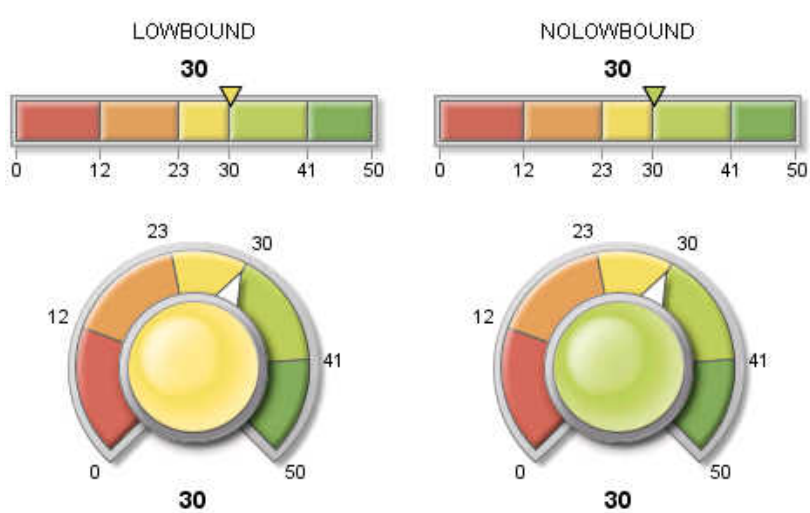
adds a drill-down link to the KPI chart. Enclose the URL in quotation marks.

Restriction The length of the URL specified must be no greater than 256 characters. Specifying a URL in excess of this limit leads to truncation of the URL in the output.

LOWBOUNDARY | NOLOWBOUNDARY

specifies whether the KPI chart displays as if the KPI value falls in the lower range or the upper range when the actual KPI value falls directly on a range boundary. This option controls the color that is used for dial centers, traffic lights, and actual value indicators. It also controls which range is displayed in the active color, if an active color list is specified. The default is LOWBOUNDARY, which tells the GKPI procedure to use the color of the lower range. If you specify NOLOWBOUNDARY, then PROC GKPI uses the color of the higher range. Figure 40.154 on page 1238 illustrates the effect of this option on dial centers and on the actual KPI value indicator in a horizontal slider when both a range boundary and the actual KPI value is 30.

Figure 40.4 *LOWBOUND and NOLOWBOUND Effect on Indicator Colors*



Alias LOWBOUND | NOLOWBOUND

Default LOWBOUNDARY

Example "Example 3: Creating a Dial KPI Chart" on page 1243

NAME="*name*"
specifies the name of the graphics output file.

The following applies to *name*:

- **For the graphics output file name:**
  - All characters are represented in lowercase.
  - The JAVAIMG device must be used in order to generate a graphics output file.
  - The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.
  - The name can include special characters.
  - Each time the graph is generated in a SAS session, SAS/GRAPH adds a number to the name, or increments the last number used to create a unique file name for the output. See "About Filename Indexing" on page 137.

Default graph.png

TARGET= *data-value*
specifies the numeric value of the target key performance indicator. If you specify a missing value (TARGET=.), then the GKPI procedure generates a KPI chart without a target value indicator.

Restriction Not supported by the TRAFFICLIGHT statement

Examples "Example 2: Creating a Gray Scale Bullet Graph" on page 1242

"Example 3: Creating a Dial KPI Chart" on page 1243

"Example 4: Defining a Speedometer" on page 1244

TYPE=FULL | HALF | QUARTER
specifies the size of the display for speedometers.

Default FULL

Restriction Valid for SPEEDOMETER statement only

See "Speedometer KPI Charts" on page 1224

Font Suboptions

You can control the fonts that are used to display the boundary and tick mark values, the actual KPI values, and the labels with BFONT=, AFONT=, and LFONT= options, respectively. Each of these options takes a font specification on the FONT= suboption.

<FONT="*fontname*</BOLD></ITALIC>"> <COLOR=*color*>
<HEIGHT=*text-height*<*units*>> <JUSTIFICATION=LEFT | CENTER | RIGHT>

Note: The JUSTIFICATION= suboption is supported by the LFONT= option only.

FONT= "*fontname*</BOLD**><**/ITALIC**>"**
specifies the font name. You can specify only system fonts (such as TrueType and UNIX system fonts), not SAS/GRAPH fonts. For more information, see "SAS/GRAPH, System, and Device-Resident Fonts" on page 350.

Alias F=

See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349

COLOR=*color*
specifies the text color. You can specify the color in any of the color-naming schemes recognized by SAS/GRAPH.

Alias C=

See "Specifying Color Names" on page 1231

HEIGHT= *text-height*<*units*>
specifies the font height in *units*. You can specify the text height in units of points (PT), centimeters (CM), inches (IN), or percentage of the graphics output area (PCT).

Alias H=

Default Units are PCT (percentage of graphics output area)

JUSTIFICATION= LEFT | RIGHT | CENTER
specifies whether the text is left-justified, centered, or right-justified within the graphics output area. You can specify LEFT, RIGHT, or CENTER.

Alias J=

Default CENTER

Restriction This suboption is supported by the LFONT= option only.

Examples: GKPI Procedure

Example 1: Using the Default Colors as the Active Colors

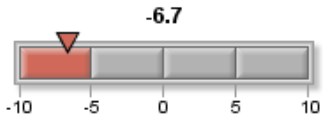
Features: PROC GKPI statement option: MODE=RAISED
HSLIDER statement options
COLORS=

ACTIVECOLORS=

Sample library member: GKPGRSLD

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

The default colors described in "Default Colors" on page 1226 can be used as the active colors instead of the inactive colors. This example uses the same value of gray for all ranges for the inactive color. It uses the red, orange, yellow-green, and green colors shown in Figure 40.152 on page 1227 as the active colors.



Program

```
goptions reset=all device=javaimg xpixels=180 ypixels=110;

proc gkpi mode=raised;
hslider actual=-6.7 bounds=(-10 -5 0 5 10) /
   colors=(cxB2B2B2 cxB2B2B2 cxB2B2B2 cxB2B2B2)
   activecolors=(cxD06959 cxE1A05D cxBDCD5F cx84AF5B);
run;

quit;
```

Program Description

Set the graphics environment. Use the XPIXELS and YPIXELS options in the GOPTIONS statement to scale the KPI charts to a size that would be appropriate for use in a dashboard.

```
goptions reset=all device=javaimg xpixels=180 ypixels=110;
```

Generate the KPI chart. Specify the range boundaries, actual KPI value, and colors. Boundary values can be positive or negative or both, but must be specified in either ascending or descending order. All colors are specified as hexadecimal RGB values. The same value of gray, cxB2B2B2, is used as the inactive color for all ranges. The default colors listed in Table 40.90 on page 1227 are used as the active colors.

```
proc gkpi mode=raised;
hslider actual=-6.7 bounds=(-10 -5 0 5 10) /
   colors=(cxB2B2B2 cxB2B2B2 cxB2B2B2 cxB2B2B2)
   activecolors=(cxD06959 cxE1A05D cxBDCD5F cx84AF5B);
run;
```

End the procedure. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

```
quit;
```

Example 2: Creating a Gray Scale Bullet Graph

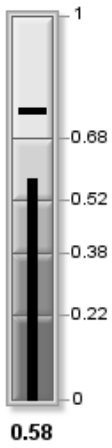
Features: PROC GKPI statement option: MODE=RAISED

VBULLET statement options
- COLORS=
- TARGET=

Sample library member: GKPGRBUL

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example creates a vertical bullet graph. It uses a gray scale color scheme that provides a good contrast between ranges. This color scheme can be used in output that is included in publications that are not in color.



Program

```
goptions reset=all device=javaimg xpixels=130 ypixels=250;

proc gkpi mode=raised;
vbullet
   actual=.58  bounds=(0 .22 .38 .52 .68 1) / target=.75
   colors=(cx747474 cx8C8C8C cxB2B2B2 cxD2D2D2 cxE6E6E6);
run;

quit;
```

Program Description

Set the graphics environment. Use the XPIXELS and YPIXELS options in the GOPTIONS statement to scale the KPI charts to a size that would be appropriate for use in a dashboard.

```
goptions reset=all device=javaimg xpixels=130 ypixels=250;
```

Generate the KPI chart. Specify the range boundaries, actual KPI value, target value, and colors. The gray scale colors are specified using hexadecimal RGB values.

```
proc gkpi mode=raised;
vbullet
   actual=.58  bounds=(0 .22 .38 .52 .68 1) / target=.75
   colors=(cx747474 cx8C8C8C cxB2B2B2 cxD2D2D2 cxE6E6E6);
run;
```

End the procedure. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

```
quit;
```

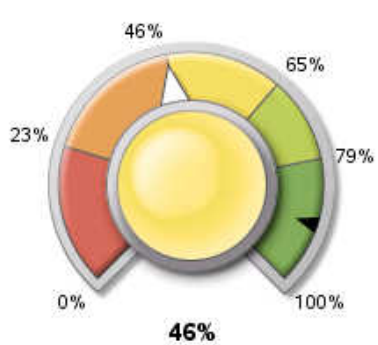
Example 3: Creating a Dial KPI Chart

Features: PROC GKPI statement option: MODE=RAISED

DIAL statement options
- AFONT=
- BFONT=
- FORMAT=
- NOLOWBOUND
- TARGET=

Sample library member: GKPDIAL

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.



Program

```
goptions reset=all device=javaimg xpixels=240 ypixels=200;

proc gkpi mode=raised;
dial actual=.46 bounds=(0 .23 .46 .65 .79 1) /
   target=.9  nolowbound  format="percent8.0"
   afont=(f="Albany AMT" height=.5cm)
   bfont=(f="Albany AMT" height=.4cm) ;
run;

quit;
```

Program Description

Set the graphics environment. Use the XPIXELS and YPIXELS options in the GOPTIONS statement to scale the KPI charts to a size that would be appropriate for use in a dashboard.

```
goptions reset=all device=javaimg xpixels=240 ypixels=200;
```

Generate the KPI chart. Specify the range boundaries, actual KPI value, and target value. In this case, the target value falls on a range boundary. The NOLOWBOUNDARY option specifies that the KPI chart behaves as if the actual KPI value falls in the higher range. The AFONT= and BFONT= options specify the fonts for the actual value and the range boundary values, respectively. The FORMAT= option specifies the SAS format for the values in the chart.

```
proc gkpi mode=raised;
dial actual=.46 bounds=(0 .23 .46 .65 .79 1) /
   target=.9  nolowbound  format="percent8.0"
   afont=(f="Albany AMT" height=.5cm)
   bfont=(f="Albany AMT" height=.4cm) ;
run;
```

End the procedure. The You must close the destination to generate output.

```
quit;
```

Example 4: Defining a Speedometer

Features: PROC GKPI statement option: MODE=RAISED

SPEEDOMETER statement options
- COLORS=
- FORMAT=
- LABEL=
- LFONT=
- TARGET=

Sample library member: GKPSPD

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form

from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

Program

```
goptions reset=all device=javaimg xpixels=210 ypixels=200;

proc gkpi mode=raised;
   speedometer actual=.72 bounds=(0 .40 .60 1) /   target=.85
   lfont=(f="Albany AMT" height=.5cm) label="Average Capacity"
   format="percent8.0";
run;

quit;
```

Program Description

Set the graphics environment. The XPIXELS and YPIXELS graphics options reduce the size of the graphics output area and therefore reduce both the size of the KPI chart and the distance between the label and the KPI chart. These options scale the KPI charts to a size that would be appropriate for use in a dashboard.

```
goptions reset=all device=javaimg xpixels=210 ypixels=200;
```

Generate the KPI chart. Specify the range boundaries, actual KPI value, and target value. The LFONT= option specifies the font for the label. The FORMAT= option specifies the SAS format for the values in the chart.

```
proc gkpi mode=raised;
   speedometer actual=.72 bounds=(0 .40 .60 1) /   target=.85
   lfont=(f="Albany AMT" height=.5cm) label="Average Capacity"
   format="percent8.0";
run;
```

End the procedure. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

```
quit;
```

Example 5: Defining a Speedometer with Reversed Colors

Features: PROC GKPI statement option: MODE=BASIC

SPEEDOMETER statement options
- COLORS=
- LABEL=

Sample library member: GKPSPCLR

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

Program

```
goptions reset=all device=javaimg xpixels=210 ypixels=200;

proc gkpi mode=basic;
speedometer actual=12 bounds=(0 25 50 100) /
   colors=(cx84AF5B cxF1DC63 cxD06959)
   label="Cancellations";
run;

quit;
```

Program Description

Set the graphics environment. The XPIXELS and YPIXELS graphics options reduce the size of the graphics output area and therefore reduce both the size of the KPI chart and the distance between the label and the KPI chart. These options scale the KPI charts to a size that would be appropriate for use in a dashboard.

```
goptions reset=all device=javaimg xpixels=210 ypixels=200;
```

Generate the KPI chart. Specify the range boundaries, actual KPI value, target value, and colors. The green, yellow, and red colors listed in Figure 40.152 on page 1227 are specified in reverse order so that green begins at zero.

```
proc gkpi mode=basic;
speedometer actual=12 bounds=(0 25 50 100) /
   colors=(cx84AF5B cxF1DC63 cxD06959)
   label="Cancellations";
run;
```

End the procedure. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

```
quit;
```

Example 6: Creating a Traffic Light

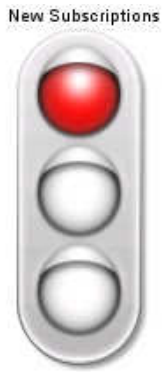
Features: PROC GKPI statement option: MODE=RAISED

TRAFFICLIGHT statement options
- COLORS=
- LABEL=
- NOAVALUE

Sample library member: GKPTRAFF

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example creates a traffic light that uses primary green, yellow, and red colors. Colors are applied to vertical KPI charts from the bottom up, so to get red at the top, you must specify red last in the list of colors.



Program

```
goptions reset=all device=javaimg xpixels=120 ypixels=210;

proc gkpi mode=raised;
```

```
   trafficlight actual=598 bounds=(1500 900 600 0) /
   colors=(green yellow red)  noavalue
   label="New Subscriptions";
run;

quit;
```

Program Description

Set the graphics environment. The XPIXELS and YPIXELS graphics options reduce the size of the graphics output area and therefore reduce both the size of the KPI chart and the distance between the label and the KPI chart. These options statement to scale the KPI charts to a size that would be appropriate for use in a dashboard.

```
goptions reset=all device=javaimg xpixels=120 ypixels=210;
```

Generate the KPI chart. Specify the range boundaries, actual KPI value, and colors. The NOAVALUE option turns off the display of the actual KPI value. The colors are specified as SAS Registry Color names.

```
proc gkpi mode=raised;
   trafficlight actual=598 bounds=(1500 900 600 0) /
   colors=(green yellow red)  noavalue
   label="New Subscriptions";
run;
```

End the procedure. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

```
quit;
```

Chapter 41

GOPTIONS Procedure

Overview: GOPTIONS Procedure

What Does the GOPTIONS Procedure Do?

The GOPTIONS procedure provides information about the values of graphics options and the global statement definitions that are currently in effect in your session. The values displayed are either the defaults of the current device driver or user-defined values that have been assigned in your SAS session. You can use the GOPTIONS procedure to do the following tasks:

- list the current values of all of the graphics options or of one specified option
- display the values of all of the AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, and TITLE definitions that are currently in effect

Note: Do not confuse the GOPTIONS procedure with the GOPTIONS statement. The GOPTIONS procedure lists the values that are defined in a GOPTIONS statement as well as in any other global statement definitions. See "GOPTIONS Statement" on page 430 for a list of the graphics options that you can set with the

GOPTIONS statement. See "Graphics Options and Device Parameters Dictionary" on page 577 for a complete description of each graphics option.

The list of graphics options is displayed in the SAS Log window and includes the names of the options, the current values, and a brief description of each one. You can use PROC GOPTIONS statement options to control what information is listed and where it appears in the Log window. Output 41.27 on page 1250 contains part of a sample Log listing.

Note: The information returned by the GOPTIONS procedure does not reflect any style settings that are in effect.

Output 41.1 *Partial Output from the GOPTIONS Procedure*

```
                            SAS/GRAPH software options and parameters
                              (executing in DMS Process environment)

NOACCESSIBLE                  Create accessible graphs
NOADMGDF                      GDDM driver output an ADMGDF file
ALTDESC                       Enable the use of the proc description for default
alt text
ASPECT=                       Aspect ratio (width/height) for software characters
NOAUTOCOPY                    Automatic hard copy after display
NOAUTOFEED                    Automatic paper feed after plot
AUTOSIZE=                     Change character cell size to preserve device
catalog rows and
                              columns
BINDING=DEFAULTEDGE           Binding edge
NOBORDER                      Draw a border around display or plot
CBACK=                        Background color
CBY=                          BY line color
CELL                          Hardware characters must be on cell boundaries
CHARACTERS                    Use hardware characters
CHARTYPE=                     Default hardware font
CIRCLEARC                     Use hardware circle/arc generator
NOCOLLATE                     Collate output
COLORS=( )                    Default color list
CPATTERN=                     Default pattern color
CSYMBOL=                      Default symbol color
CTEXT=                        Default text color
CTITLE=                       Default title, footnote and note color
DASH                          Use hardware dashed line generator
DASHSCALE=                    Dash pattern scale factor
DELAY=                        Animation delay time in 100ths of a second
DEVADDR=                      IBM Device address, qname, or node name
DEVICE=                       Graphics output device
DEVMAP=DEFAULT                Output character map for hardware text
DISPLAY                       Display graph on device
DISPOSAL=NONE                 Image animation disposal method
DRVINIT=                      Host command executed before driver initialization
DRVTERM=                      Host command executed after driver termination
NODUPLEX                      Duplex printing
NOERASE                       Erase graph upon completion
EXTENSION=                    Driver preferred file extension
NOFASTTEXT                    Use quicker, less precise, integer font rendering
routines;
                              generally unsuitable for multiple device or
templated replay
                              situations.
```

Note: All of the graphics options that are displayed by the GOPTIONS procedure are described in "Graphics Options and Device Parameters Dictionary" on page 577.

Syntax: GOPTIONS Procedure

PROC GOPTIONS <*options*>;

PROC GOPTIONS Statement

Lists the graphics options, and their values and descriptions in the Log window. Can also list the currently defined global statements. By default, each listed item is displayed on a separate line.

Syntax

PROC GOPTIONS <*options*>;

Summary of Optional Arguments

Item request options

AXIS
: requests a list of all current AXIS definitions.

FOOTNOTE
: requests a list of all of the current FOOTNOTE and TITLE definitions.

LEGEND
: requests a list of all of the current LEGEND definitions.

OPTION=*graphics-option*
: requests information about the specified graphics option.

PATTERN
: requests a list of all of the current PATTERN definitions.

SYMBOL
: requests a list of all of the current SYMBOL definitions.

TITLE
: requests a list of all of the current TITLE and FOOTNOTE definitions.

Listing format options

CENTIMETERS

displays the values of the HORIGIN=, HSIZE=, PAPERFEED=, PAPERLIMIT=, VORIGIN=, and VSIZE= graphics options in units of centimeters (CM).

NOLIST
suppresses the display of graphics options.

NOLOG
displays the output in the OUTPUT window instead of the Log window.

SHORT
suppresses the descriptions of the graphics options and displays the graphics options values in an alphabetical list in paragraph form.

Optional Arguments

You can specify as many options as you want and list them in any order.

AXIS
requests a list of all current AXIS definitions. AXIS also lists the current values for all graphics options, unless you use the NOLIST option. If you have not defined any AXIS statements, the GOPTIONS procedure issues a message.

Alias A

CENTIMETERS
displays the values of the HORIGIN=, HSIZE=, PAPERFEED=, PAPERLIMIT=, VORIGIN=, and VSIZE= graphics options in units of centimeters (CM). These graphics options use units of IN or CM only, and their values are always stored as inches even if you specify CM. Therefore, the GOPTIONS procedure displays these values in inches, unless you specify the CENTIMETERS option.

Alias CM

Note The CENTIMETERS option does not affect the graphics options that can use unit specifications of CELLS, CM, IN, PCT, and PT.

FOOTNOTE
requests a list of all of the current FOOTNOTE and TITLE definitions. FOOTNOTE also lists the current values for all of the graphics options, unless you use the NOLIST option. If you have not defined any FOOTNOTE or TITLE statements, the GOPTIONS procedure issues a message.

Alias F

Example “Example 1: Displaying TITLE and FOOTNOTE Statements” on page 1254

LEGEND
requests a list of all of the current LEGEND definitions. LEGEND lists the current values for all of the graphics options, unless you use the NOLIST option. If you have not defined any LEGEND statements, the GOPTIONS procedure issues a message.

Alias L

NOLIST
suppresses the display of graphics options. Use the NOLIST option in conjunction with the appropriate statement request option when you want to list

only the current AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, or TITLE definitions.

Alias N

Example "Example 1: Displaying TITLE and FOOTNOTE Statements" on page 1254

NOLOG
displays the output in the OUTPUT window instead of the Log window.

OPTION=*graphics-option*
requests information about the specified graphics option. For these options, requesting the information about one option also displays the value of its corresponding option, as follows:

- HSIZE= and VSIZE=
- HPOS= and VPOS=
- XMAX= and YMAX=
- XPIXELS= and YPIXELS=

PATTERN
requests a list of all of the current PATTERN definitions. The PATTERN option lists the current values for all of the graphics options unless you use the NOLIST option. If you have not defined any PATTERN statements, the GOPTIONS procedure issues a message.

Alias P

SHORT
suppresses the descriptions of the graphics options and displays the graphics options values in an alphabetical list in paragraph form.

Example "Example 2: Displaying Graphics Options without the Description" on page 1255

SYMBOL
requests a list of all of the current SYMBOL definitions. The SYMBOL option lists the current values for all of the graphics options, unless you use the NOLIST option. If you have not defined any SYMBOL statements, the GOPTIONS procedure issues a message.

Alias S

TITLE
requests a list of all of the current TITLE and FOOTNOTE definitions. The TITLE option lists the current values for all of the graphics options, unless you use the NOLIST option. If you have not defined any FOOTNOTE or TITLE statements, the GOPTIONS procedure issues messages.

Alias T

Examples: GOPTIONS Procedure

Example 1: Displaying TITLE and FOOTNOTE Statements

Features:	PROC GOPTIONS statement options FOOTNOTE and NOLIST
Sample library member:	GOPTIFT

This example uses the FOOTNOTE option to display the current definitions of both the FOOTNOTE and TITLE statements. It also uses the NOLIST option to suppress the list of graphics options. Output 41.28 on page 1254 shows the listing that appears in the Log window.

Output 41.2 *Using the NOLIST Option (GOPTIFT)*

```
TITLE1 HEIGHT=6 COLOR=BLUE FONT=SWISSB "Production Quality" ;
TITLE2 HEIGHT=4 COLOR=BLUE FONT=SWISSB "January through June";

FOOTNOTE1 HEIGHT=3 COLOR=GREEN FONT=SWISS "Data from SASDATA.QUALITY" ;

FOOTNOTE2 HEIGHT=3 COLOR=GREEN FONT=SWISS "* denotes approximations";
```

Program

```
goptions reset=all;

title1 h=6 c=blue f=swissb "Production Quality";
title2 h=4 c=blue f=swissb "January through June";
footnote1 h=3 c=green f=swiss "Data from SASDATA.QUALITY";
footnote2 h=3 c=green f=swiss "* denotes approximations";

proc goptions nolist footnote;
run;
```

Program Description

Clear all global statements.

```
goptions reset=all;
```

Define titles and footnotes.

```
title1 h=6 c=blue f=swissb "Production Quality";
title2 h=4 c=blue f=swissb "January through June";
footnote1 h=3 c=green f=swiss "Data from SASDATA.QUALITY";
footnote2 h=3 c=green f=swiss "* denotes approximations";
```

Produce the listing. The NOLIST and FOOTNOTE options control the information that appears in the Log window.

```
proc goptions nolist footnote;
run;
```

Example 2: Displaying Graphics Options without the Description

Features: PROC GOPTIONS statement option SHORT

Sample library member: GOPSHORT

This example uses the SHORT option to display only the values of graphics options without the description of each graphics option. Output 41.29 on page 1255 shows the listing that appears in the Log window.

Output 41.3 *Using the SHORT Option (GOPSHORT)*

```
                          SAS/GRAPH software options and parameters
                            (executing in DMS Process environment)
NOACCESSIBLE NOADMGDF ALTDESC ASPECT= NOAUTOCOPY NOAUTOFEED AUTOSIZE= BINDING=DEFAULTEDGE
NOBORDER CBACK= CBY= CELL CHARACTERS CHARTYPE= CIRCLEARC NOCOLLATE COLORS=( ) CPATTERN=
CSYMBOL= CTEXT= CTITLE= DASH DASHSCALE= DELAY= DEVADDR= DEVICE= DEVMAP=DEFAULT DISPLAY
DISPOSAL=NONE DRVINIT= DRVTERM= NODUPLEX NOERASE EXTENSION= NOFASTTEXT FBY= FCACHE=3 FILECLOSE=
FILEONLY FILL FILLINC= FONTRES=NORMAL FTEXT= FTITLE= FTRACK=TIGHT GACCESS= GCLASS=G GCOPIES=(0,
20) GDDMCOPY=FSCOPY GDDMNICKNAME= GDDMTOKEN= GDEST=LOCAL GEND= GEPILOG= GFORMS= GOUTMODE=APPEND
GPROLOG= GPROTOCOL= GRAPHRC GSFLEN= GSFMODE=PORT GSFNAME= NOGSFPROMPT GSIZE= GSTART=
GUNIT=CELLS GWAIT= GWRITER=SASWTR HANDSHAKE= HBY= HORIGIN= HPOS= HSIZE= HTEXT= HTITLE= IBACK=
IMAGEPRINT IMAGESTYLE=TILE INTERPOL= ITERATION= NONINTERLACED KEYMAP=DEFAULT LFACTOR=
OFFSHADOW=(0.0625 in., -0.0625 in.) PAPERDEST= PAPERFEED= PAPERLIMIT= PAPERSIZE= PAPERSOURCE=
PAPERTYPE= PENMOUNTS= PENSORT PIEFILL NOPCLIP POLYGONCLIP POLYGONFILL POSTGEPILOG= POSTGRAPH=
POSTGPROLOG= PPDFILE= PREGEPILOG= PREGRAPH= PREGPROLOG= PROMPT PROMPTCHARS='000A010D05000000'X
RENDER=MEMORY RENDERLIB=WORK REPAINT= NOREVERSE NOROTATE SIMFONT= SPEED= NOSWAP
SWFONTRENDER=DEFAULT SYMBOL TARGETDEVICE= NOTRANSPARENCY TRANTAB= UCC= NOUSERINPUT NOV6COMP
VORIGIN= VPOS= VSIZE= XMAX= XPIXELS= YMAX= YPIXELS=
```

Program

```
goptions reset=all;

proc goptions short;
run;
```

Program Description

Set the graphics environment. The values of the graphics options specified in this statement appear in the Log listing.

```
goptions reset=all;
```

Produce the listing. The SHORT option suppresses the display of the description of each graphics option.

```
proc goptions short;
run;
```

Chapter 42

GPLOT Procedure

Overview: GPLOT Procedure

About the GPLOT Procedure

The GPLOT procedure plots the values of two or more variables on a set of coordinate axes (X and Y). The coordinates of each point on the plot correspond to two variable values in an observation of the input data set. The procedure can also generate a separate plot for each value of a third (classification) variable. It can also generate bubble plots in which circles of varying proportions representing the values of a third variable are drawn at the data points.

The procedure produces a variety of two-dimensional graphs including the following plots:

- simple scatter plots
- overlay plots in which multiple sets of data points are displayed on one set of axes
- plots against a second vertical axis
- bubble plots
- logarithmic plots (controlled by the AXIS statement)

In conjunction with the SYMBOL statement, the GPLOT procedure can produce join plots, high-low plots, needle plots, and plots with simple or spline-interpolated lines. The SYMBOL statement can also display regression lines on scatter plots.

The GPLOT procedure is useful for the following tasks:

- displaying a long series of data, and showing trends and patterns
- interpolating between data points
- extrapolating beyond existing data with the display of regression lines and confidence limits

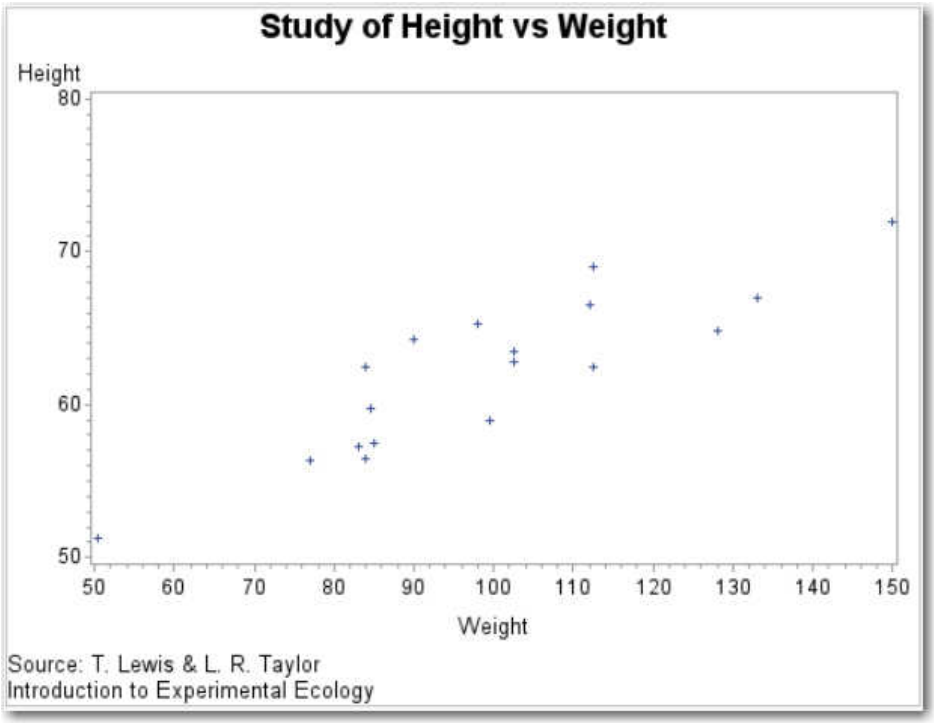
About Plots of Two Variables

Plots of two variables display the values of two variables as data points on one horizontal axis (X) and one vertical axis (Y). Each pair of X and Y values forms a data point.

The following figure shows a simple scatter plot that plots the values of the variable HEIGHT on the vertical axis and the variable WEIGHT on the horizontal axis. By default, the PLOT statement scales the axes to include the maximum and minimum data values and displays a symbol at each data point. It labels each axis with the

name of its variable or an associated label and displays the value of each major tick mark.

Figure 42.1 *Scatter Plot of Two Variables (GPLVRBL1(a))*



The program for this plot is in "Example 4: Plotting Two Variables" on page 1345. For more information about producing scatter plots, see "PLOT Statement" on page 1296.

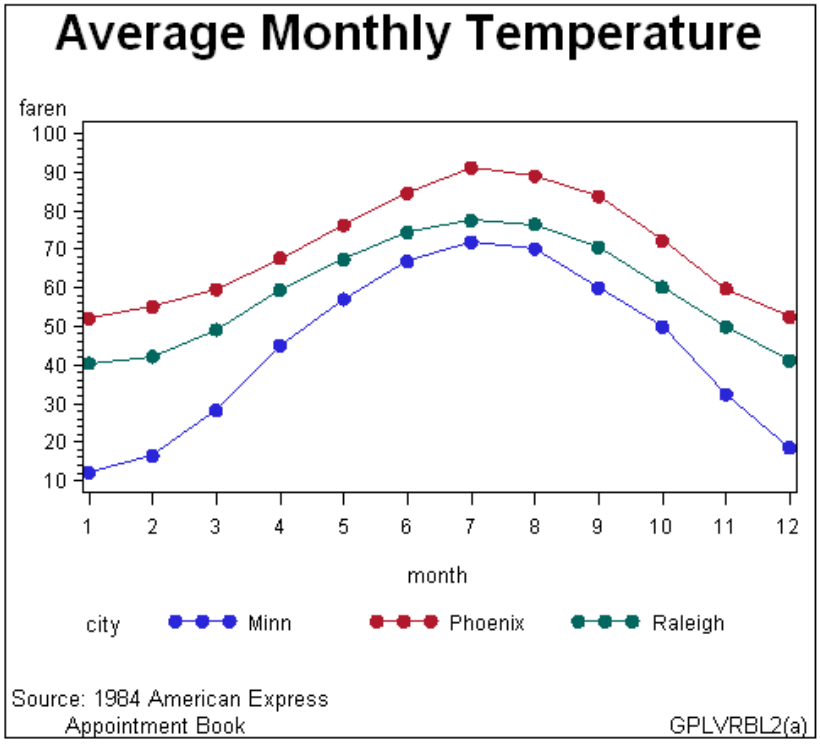
You can also overlay two or more plots (multiple sets of data points) on a single set of axes. You can also apply a variety of interpolation techniques to these plots. See "About Interpolation Methods" on page 1262.

About Plots with a Classification Variable

Plots that use a classification variable produce a separate set of data points for each unique value of the classification variable and display all sets of data points on one set of axes.

The following figure shows multiple line plots that compare yearly temperature trends for three cities. The legend explains the values of the classification variable, CITY.

Figure 42.2 *Plot of Three Variables with Legend (GPLVRBL2(a))*



By default, plots with a classification variable generate a legend. In the code that generates the plot for "Example 8: Plotting Three Variables" on page 1358, a SYMBOL statement connects the data points. It also specifies the plot symbol that is used for each value of the classification variable (CITY). For more information about how to produce plots with a classification variable, see "PLOT Statement" on page 1296.

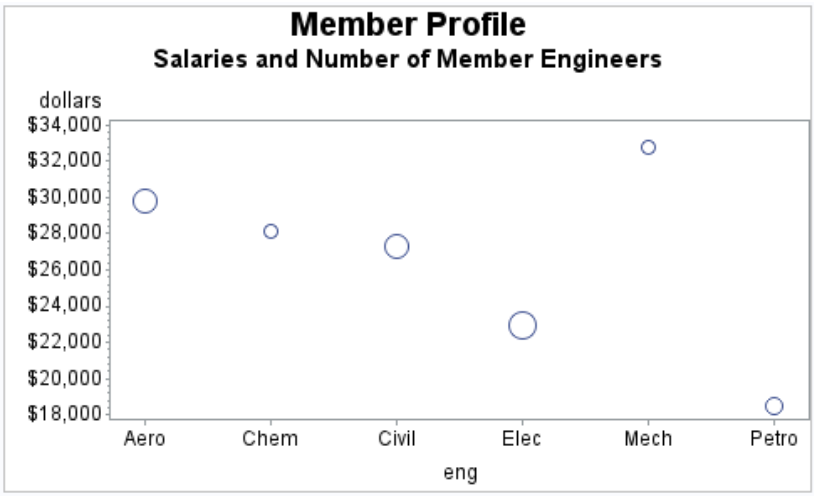
About Bubble Plots

Bubble plots represent the values of three variables by drawing circles of varying sizes at points that are plotted on the vertical and horizontal axes. Two of the variables determine the location of the data points, while the values of the third variable control the size of the circles.

Figure 42.157 on page 1261 shows a bubble plot in which each bubble represents a category of engineer that is shown on the horizontal axis. The location of each bubble in relation to the vertical axis is determined by the average salary for the category. The size of each bubble represents the number of engineers in the category relative to the total number of engineers in the data.

By default, the BUBBLE statement scales the axes to include the maximum and minimum data values and draws a circle at each data point. It labels each axis with the name of its variable or an associated label and displays the value of each major tick mark.

Figure 42.3 *Bubble Plot (GPLBUBL1)*



The program for this plot is in "Example 1: Generating a Simple Bubble Plot" on page 1337. For more information about producing bubble plots, see "BUBBLE Statement" on page 1268.

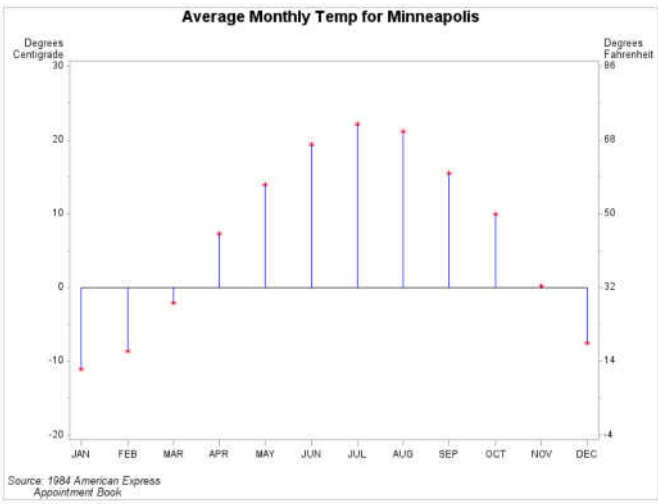
About Plots with Two Vertical Axes

Plots with two vertical axes have a right vertical axis that can do the following:

- display the same variable values as the left axis
- display left axis values in a different scale
- plot a second response (Y) variable, thereby producing one or more overlay plots

In the following figure, the right axis displays the values of the vertical coordinates in a different scale from the scale that is used for the left axis.

Figure 42.4 *Plot with a Right Vertical Axis (GPLSCVL1)*



The program for this plot is in “Example 9: Plotting with Different Scales of Values” on page 1363. For more information about how to produce plots with a right vertical axis, see “PLOT2 Statement” on page 1318 and “BUBBLE2 Statement” on page 1284.

About Interpolation Methods

In addition to these graphs, you can produce other types of plots such as box plots or high-low-close plots by specifying various interpolation methods with the SYMBOL statement. Use the SYMBOL statement to do the following tasks:

- connect the data points with straight lines
- specify regression analysis to fit a line to the points and can display lines for confidence limits
- connect the data points to the zero line on the vertical axis
- display the minimum and maximum values of Y at each X value and mark the mean value. Also, display standard deviations that connect the data points with lines or bars, generate box plots, or plot high-low-close stock market data
- specify that a pattern fills the polygon that is defined by data points
- smooth plot lines with spline interpolation
- use a step function to connect the data points

The “SYMBOL Statement” on page 470 describes all interpolation methods.

Concepts: GPLOT Procedure

Parts of a Plot

Some terms used with GPLOT procedure are illustrated in Figure 42.159 on page 1263 and Figure 42.160 on page 1264.

Figure 42.5 GPLOT Procedure Terms

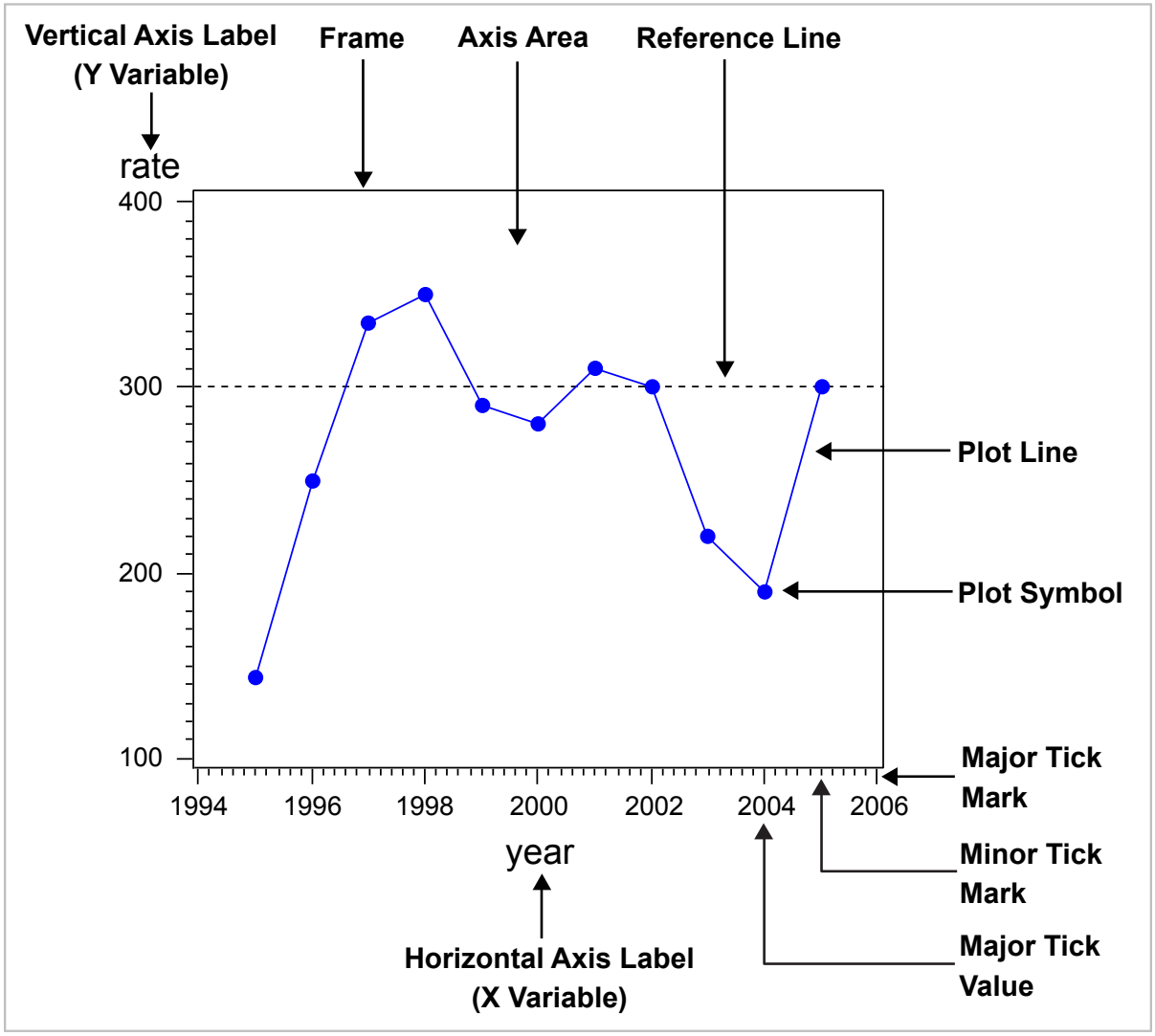
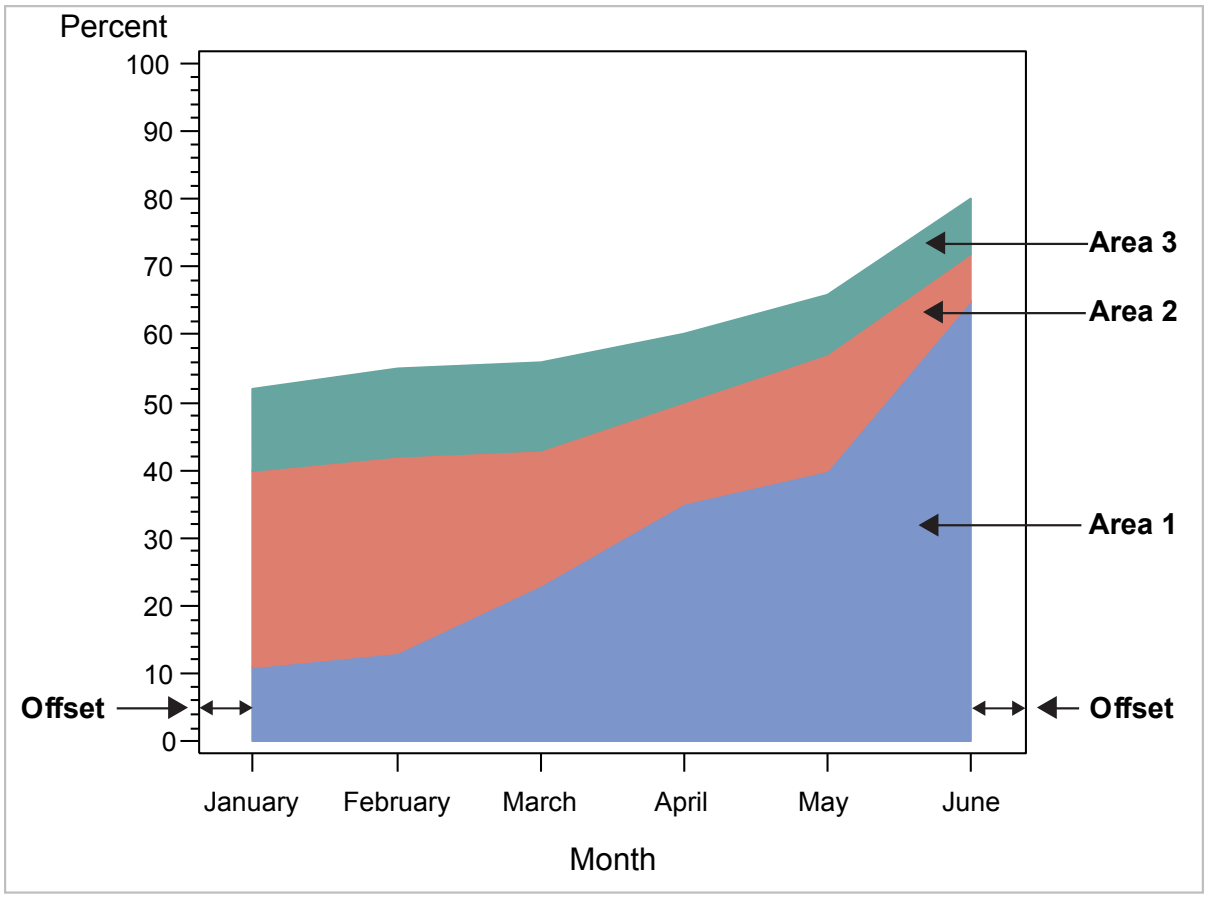


Figure 42.6 *Additional GPLOT Procedure Terms*



About the Input Data Set

Data Set Requirements

The input data set that is used by the GPLOT procedure must contain at least one variable to plot on the horizontal axis and one variable to plot on the vertical axis. Typically, the horizontal axis shows an independent variable (time, for example), and the vertical axis shows a dependent variable (temperature, for example). With the exception of the Java and ActiveX device drivers, variables can be either character or numeric. For the ActiveX and Java device drivers, the horizontal axis can contain either character or numeric values, but the vertical axis can contain only numeric variables. Graphs are automatically scaled to the values of the character data or to include the values of numeric data. However, you can control scaling with procedure options or with associated AXIS statements.

Missing Values

If the value of either of the plot variables is missing, the GPLOT procedure does not include the observation in the plot. If you specify interpolation with a SYMBOL definition, the plot is not broken at the missing value. To break the plot line or area fill at the missing value, use the PLOT statement's SKIPMISS option. The SKIPMISS option is enabled only for JOIN interpolation.

Values Out of Range

Data values can be excluded from a graph by restricting the range of axis values with the VAXIS= or HAXIS= options or with the ORDER= option in an AXIS statement. When an observation contains a value outside of the specified axis range, the GPLOT procedure excludes the observation from the plot and issues a message to the log.

If you specify interpolation with a SYMBOL definition, by default values outside of the axis range are excluded from interpolation calculations. As a result, interpolated values for the plot might change. Values that are omitted from interpolation calculations have a particularly noticeable effect on the high-low interpolation methods: HILO, STD, and BOX. In addition, regression lines and confidence limits represent only part of the original data.

To specify that values out of range are included in the interpolation calculations, use the MODE= option in a SYMBOL statement. When MODE=INCLUDE, values that fall outside of the axis range are included in interpolation calculations but excluded from the plot. The default (MODE=EXCLUDE) omits observations that are outside of the axis range from interpolation calculations. See the MODE= option in the “SYMBOL Statement” on page 470 for details.

Sorted Data

Data points are plotted in the order in which the observations are read from the data set. Therefore, if you use any type of interpolation that generates a line, sort your data by the horizontal axis variable.

Logarithmic Axes

If your data contain logarithmic values or if the data values vary over a wide range or contain large values, you might want to specify a logarithmic axis for the horizontal or vertical axis. Logarithmic axes can be specified with the AXIS statement options LOGBASE= and LOGSTYLE=. See the “AXIS Statement” on page 399 for a complete discussion.

Syntax: GPLOT Procedure

Requirements: An input data set is required. If not specified, an input data set must have been created before issuing the PROC GPLOT statement.

At least one PLOT or BUBBLE statement is required. A PLOT2 or BUBBLE2 statement can be used in conjunction with a PLOT or BUBBLE statement.

Supported Statements: AXIS, FOOTNOTE, LEGEND, PATTERN, TITLE

Supports: RUN-group processing

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 153.

The procedure can include the SAS/GRAPH statement BY on page 424, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See “SAS/GRAPH Statements” on page 397 and *SAS DATA Step Statements: Reference* for more information.

Tip: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, `ODS PDF; procedure-code; ODS PDF CLOSE;`). See Chapter 7, “Using Run-Group Processing,” on page 75 for more information.

PROC GPLOT DATA=*input-data-set*
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP= *output-data-set*>
<UNIFORM>;
  BUBBLE *plot-request(s)* <*/ options*>;
    BUBBLE2 *plot-request(s)* <*/ options*>;
  PLOT *plot-request(s)* <*/ options*>;
    PLOT2 *plot-request(s)* <*/ options*>;

PROC GPLOT Statement

Identifies the data set that contains the plot variables. Can specify uniform axis scaling for all graphs as well as an annotate data set and an output catalog.

Requirement: An input data set is required. If not specified, an input data set must have been created before issuing the PROC GPLOT statement.

Syntax

PROC GPLOT DATA=*input-data-set*

<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>
<UNIFORM>;

Required Argument

DATA=*input-data-set*
specifies the SAS data set that contains the variables to plot. By default, the procedure uses the most recently created SAS data set.

See "The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

"About the Input Data Set" on page 1264

Optional Arguments

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

Alias ANNO=

See Chapter 28, "Using Annotate Data Sets," on page 721

GOUT=<*libref.* >*output-catalog*
specifies the SAS catalog in which to save the graphics output that is produced by the GPLOT procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

See "Specifying the Catalog Name and Entry Name for Your GRSEGs " on page 138

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255 and "Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" on page 262

UNIFORM
specifies that the same axis scaling is used for all graphs that are produced by the procedure. The same axis scaling would apply across Run-groups as well. By default, the range of axis values for each axis is based on the minimum and maximum values in the data. Therefore, they can vary from graph to graph and

among BY groups. Using the UNIFORM option forces the value range for each axis to be the same for all graphs. Take for example, a procedure that produces multiple graphs with both left and right vertical axes. The UNIFORM option scales all of the left axes the same and all of the right axes the same. The scaling is based on the minimum and maximum data values.

In addition, UNIFORM forces the assignment of SYMBOL statements for the category variable without regard to the BY-group variable. If a legend is generated, UNIFORM makes the legend the same across graphs.

Restriction Partially supported by Java and ActiveX

BUBBLE Statement

Creates bubble plots in which a third variable is plotted against two variables represented by the horizontal and vertical axes; the value of the third variable controls the size of the bubble.

Requirement: At least one plot request is required.

Supported Statements: AXIS, FOOTNOTE, TITLE

Syntax

BUBBLE *plot-request(s)* <*/ options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate all graphs that are produced by the GPLOT procedure.

Bubble appearance options

BCOLOR=*bubble-color*
: specifies the color for the bubbles.

BFILL=SOLID | GRADIENT
: enables you to generate solid or gradient-filled bubbles.

BFONT=*font*
: specifies the font to use for bubble labels.

BLABEL
: labels the bubbles with the values of the third variable.

BSCALE=AREA | RADIUS
: specifies whether the bubble-scaling proportion is based on the area of the circles or the radius measure.

BSIZE=*multiplier*

specifies an overall scaling factor for the bubbles so that you can increase or decrease the size of all bubbles by this factor.

Catalog options

DESCRIPTION="*description*"
specifies a description of the output.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Horizontal axis options

AUTOHREF
draws reference lines at all major tick marks on the horizontal axis.

CAUTOHREF=*reference-line-color*
specifies colors for reference lines drawn at major tick marks on the horizontal axis, as specified by the AUTOHREF option.

CHREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the horizontal axis.

HAXIS=*value-list* | AXIS<1 – 99>
specifies major tick mark values for the horizontal axis or assigns an axis definition.

HMINOR=*number-of-minor-ticks*
specifies the number of minor tick marks that are drawn between each major tick mark on the horizontal axis.

HREF=*value* | (*value*) | (*value-list*)
draws one or more reference lines perpendicular to the horizontal axis at points that are specified by *value-list*.

HREVERSE
specifies that the order of the values on the horizontal axis be reversed.

HZERO
specifies that tick marks on the horizontal axis begin in the first position or end in the last position with a value of zero, depending on the horizontal variable values.

LAUTOHREF=*reference-line-type*
specifies the line type for reference lines at major tick marks on the horizontal axis, as specified by the AUTOHREF option.

LHREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines drawn perpendicular to the horizontal axis.

WAUTOHREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the horizontal axis as determined by the AUTOHREF option.

WHREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines as determined by the horizontal axis.

ODS options

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*

specifies a character variable whose values are URLs.

Plot appearance options

CAXIS=*axis-color*
specifies the color for the axis line and all major and minor tick marks.

CFRAME=*background-color*
fills the axis area with the specified color.

CTEXT=*text-color*
specifies the color for all text on the axes, including tick mark values, axis labels, and bubble labels.

DATAORDER='*entry-description*'
plots character of midpoint-type data in data order instead of the default alphabetical order.

FRAME
specifies that a line is drawn around the axis area.

FRONTREF
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

GRID
draws reference lines at all major tick marks on both axes.

IFRAME=*fileref* | '*external-file*'
identifies the image file you that want to apply to the axis area of the plot.

IMAGESTYLE=TILE | FIT
specifies whether to use multiple instances of an image to fill the axis area (TILE) or to stretch a single instance of an image to fill the axis area (FIT).

NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

NOFRAME
specifies that a line is not drawn around the axis area.

NOFRAME
specifies that a line is not drawn around the axis area.

Vertical axis options

AUTOVREF
draws reference lines at all major tick marks on the vertical axis.

CAUTOVREF=*reference-line-color*
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option.

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the vertical axis.

LAUTOVREF=*reference-line-type*
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option.

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines drawn perpendicular to the vertical axis.

VAXIS=*value-list* | AXIS<1 – 99>
specifies the major tick mark values for the vertical axis or assigns an axis definition.

VMINOR=*number-of-minor-ticks*

specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis.

VREF=*value* | (*value*) | (*value-list*)
draws one or more reference lines perpendicular to the vertical axis at specified points.

VREVERSE
specifies that the order of the values on the vertical axis should be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

WAUTOVREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option.

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines as determined by the vertical axis.

Required Argument

plot-request(s)
each specifies the variables to plot and produces a separate graph. All variables must be in the input data set. Multiple plot requests are separated with blanks. A plot request must have this form:

y-variable*x-variable=bubble-size
plots the values of two variables and draws a circle (bubble) at each data point. The value of the third variable determines the size of the bubble.

y-variable
variable plotted on the left vertical axis.

x-variable
variable plotted on the horizontal axis.

bubble-size
variable that specifies the size of the bubbles. *Bubble-size* must be numeric. If the value of *bubble-size* is positive, bubbles are drawn with a solid line. If it is negative, bubbles are drawn with a dashed line.

Notes If you specify the JAVAIMG, ACTIVEX, or ACTXIMG device drivers, then either the *x-variable* or the *y-variable* must be numeric.

If you specify the *x-variable* as a character and the *y-variable* as numeric, SAS/GRAPH converts the *X axis* to display the character values and the *Y axis* to display the numeric values.

Optional Arguments

Options in a BUBBLE statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

Alias ANNO=

See Chapter 28, “Using Annotate Data Sets,” on page 721

AUTOHREF
draws reference lines at all major tick marks on the horizontal axis. LAUTOHREF=, CAUTOHREF=, and WAUTOHREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the HAXIS= option.

AUTOVREF
draws reference lines at all major tick marks on the vertical axis. LAUTOVREF=, CAUTOVREF=, and WAUTOVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

BCOLOR=*bubble-color*
specifies the color for the bubbles. You might not specify the BCOLOR= option. In that case, the bubble color becomes the color of the default style (GSTYLE) or the color specified by the current ODS style (if used).

Style reference ContrastColor attribute of the GraphOutline, GraphData1, and TwoColorAltRamp elements

Examples “Example 2: Labeling and Sizing Plot Bubbles” on page 1339

“Example 3: Adding a Right Vertical Axis” on page 1342

BFILL=SOLID | GRADIENT
enables you to generate solid or gradient-filled bubbles. By default, the Java and ActiveX devices create solid bubbles.

- BFILL=SOLID fills the bubbles with the color specified by the BCOLOR= option. If the BFILL option is not specified, then the color is specified by the current style. If you are using specific ODS style, the color comes from the contrast color attribute within the GraphData1 style element.
- BFILL=GRADIENT starts with the current axis area color and gradually transitions to the color specified with the BCOLOR= option or the color of the current style. If you are using an ODS style, the colors are controlled by the `startcolor` and `endcolor` attributes of the TwoColorAltRamp style element.

Restriction Not supported by Java and ActiveX

Note The SAS/GRAPH ActiveX control displays negative values as empty circles.

BFONT=*font*

specifies the font to use for bubble labels. See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349 for details about how to specify *font*. If you omit the BFONT= option, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the font specified by the current style
3. the default hardware font

Style reference Font attribute of the GraphValueText element

Restriction Not supported by Java and ActiveX

See the option "BLABEL" on page 1273 for information about the location and color of labels

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

BLABEL

labels the bubbles with the values of the third variable. If the variable has a format, the formatted value is used. By default, bubbles are not labeled.

The procedure normally places labels directly outside of the circle at 315 degrees rotation. If a label in this position does not fit in the axis area, other 45-degree placements (that is, 45, 135, and 225 degrees) are attempted. If the label cannot be placed at any of the positions (45, 135, 225, or 315 degrees) without being clipped, the label is omitted. However, labels can collide with other bubbles or previously placed labels.

Labels are displayed in the color specified by the CTEXT= option. If you omit the CTEXT=option, the default is the color of the current style.

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

BSCALE=AREA | RADIUS

specifies whether the bubble-scaling proportion is based on the area of the circles or the radius measure. By default, BSCALE=AREA.

The value that is assigned to the BSCALE= option affects how large the bubbles appear in relation to each other. For example, suppose the third variable value is twice as big for one bubble as it is for another. If BSCALE=AREA, the area of the larger bubble is twice the area of the smaller bubble. If BSCALE=RADIUS, the radius of the larger bubble is twice the radius of the smaller bubble and the larger bubble has more than twice the area of the smaller bubble.

Restriction Not supported by Java and ActiveX

BSIZE=*multiplier*

specifies an overall scaling factor for the bubbles so that you can increase or decrease the size of all bubbles by this factor.

In web output, the Java applets and the ActiveX Control override the default value. To prevent this override, specify a value for the BSIZE= option, rather than relying on the default value.

Restriction Partially supported by Java and ActiveX

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

CAUTOHREF=*reference-line-color*
specifies colors for reference lines drawn at major tick marks on the horizontal axis, as specified by the AUTOHREF option. The default color is either the value of the CAXIS= option or the first color in the color list.

Style reference Color attribute of the GraphGridLines element

CAUTOVREF=*reference-line-color*
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option. If you do not specify the CAUTOREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Style reference Color attribute of the GraphGridLines element

CAXIS=*axis-color*
specifies the color for the axis line and all major and minor tick marks. By default, the procedure uses the color of the current style.

The CAXIS= option is overridden by the COLOR= option in an AXIS definition. The COLOR= option in an AXIS definition is overridden by the COLOR= suboption of the MAJOR= and MINOR= options in an AXIS definition.

Alias CA=

Style reference Color attribute of the GraphAxisLines attribute

CFRAME=*background-color*
fills the axis area with the specified color. If the FRAME option is also in effect, the procedure determines the color of the frame according to the precedence list given for the FRAME option description. If the IFRAME= option is in effect, the specified image fills the axis area instead of the specified color.

Style reference Color attribute of the GraphWalls element

CHREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the horizontal axis. This option affects reference lines drawn with the AUTOHREF, HREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. Specifying in parentheses a single color applies that color only to the first reference line drawn with the HREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the HREF= option. The syntax of the color list requires parentheses and each color separated by spaces (*color1 color2 ... colorN*). If you do not specify the CHREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias CH=

Style reference Color attribute of the GraphReference element

Interaction The CAUTOHREF= option overrides the CHREF= option for lines drawn with the AUTOHREF option

CTEXT=*text-color*
specifies the color for all text on the axes, including tick mark values, axis labels, and bubble labels. The GPLOT procedure searches for a color specification in this order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the PLOT statement
2. the color specified by the CTEXT= option in the PLOT statement
3. the color specified by the CTEXT= option in the GOPTIONS statement
4. the color specified in the current style, or a NOGSTYLE option specification. With NOGSTYLE specified, the default color is black for the Java and ActiveX devices and the first color in the color list for all other devices

In an AXIS statement, the COLOR= suboption of either a LABEL= option or a VALUE= option overrides the CTEXT= option. It also determines the color of the axis label, or the color of the tick mark values, respectively.

Alias C=, CT=

Style reference Color attributes of the GraphValueText and the GraphLabelText elements

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the vertical axis. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the VREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the VREF= option. The syntax of the color list requires parentheses and each color separated by spaces (*color1 color2 ... colorN*). If you do not specify the CVREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias CV=

DATAORDER='*entry-description*'
plots character of midpoint-type data in data order instead of the default alphabetical order.

Restriction Supported by Java and ActiveX only

DESCRIPTION="*description*"
specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See “Chart Descriptions for Web Presentations” on page 251for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.

- the description and properties for the catalog entry in the SAS Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default Based on the specified value of *bubble-size* in the *plot-request*, either BUBBLE PLOT OF *chart-variable* by *chart-variable* or BUBBLE PLOT OF *chart-variable* by *chart-variable* sized by *chart-variable*

FRAME

specifies that a line is drawn around the axis area. For the frame color, a specification is searched for in this order:

1. the CAXIS= option
2. the COLOR= option in the AXIS definition assigned to the vertical axis
3. the COLOR= option in the AXIS definition assigned to the horizontal axis
4. the default, which is the color defined by the current style

Alias FR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1274 option.

To fill the axis area with an image, use the "IFRAME=*fileref* | '*external-file*'" on page 1278 option.

See To override the default and specify that no line is drawn around the axis area, use the "NOFRAME " on page 1277 option.

FRONTREF

specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars. By default, reference lines are drawn on the back plane of the axis.

GRID

draws reference lines at all major tick marks on both axes. You get the same result when you use all of these options in a BUBBLE statement: AUTOHREF, AUTOVREF, FRAME, LVREF=34, and LHREF=34. The line type for GRID is 34.

The line color is the color of the axis.

HAXIS=*value-list* | AXIS<1 – 99>

specifies major tick mark values for the horizontal axis or assigns an axis definition. By default, the procedure scales the axis and provides an appropriate

number of tick marks. For a description of *value-list*, see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. To assign labels to horizontal reference lines, specify an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines drawn with the AUTOHREF and HREF= options.

If you assign an axis definition that does not currently exist, the option is ignored.

If data values fall outside of the range that is specified by the HAXIS= option, then by default the outlying data values are not used in interpolation calculations.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the "AXIS Statement" on page 399.

Restriction Partially supported by Java and ActiveX

See "About the Input Data Set" on page 1264 for more information about values out of range

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

HMINOR=*number-of-minor-ticks*

specifies the number of minor tick marks that are drawn between each major tick mark on the horizontal axis. Minor tick marks are not labeled. The HMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias HM=

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

HREF=*value* | (*value*) | (*value-list*)

draws one or more reference lines perpendicular to the horizontal axis at points that are specified by *value-list*. For a description of *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. To specify labels for these reference lines, use the HAXIS= option.

Values can be listed in any order, but should be within the range of values represented by the chart response axis. A warning is written to the SAS log if any of the points are off of the axis, and no reference line is drawn for such points. You can use the AUTOREF option to draw reference lines automatically at all of the major tick marks

The LHREF=, CHREF=, and WHREF= options can be used to change the line types, colors, and widths of these reference lines, respectively.

NOFRAME

specifies that a line is not drawn around the axis area. To override the default and specify that no line is to be drawn, use the NOFRAME option.

Alias NOFR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1274 option.

To fill the axis area with an image, use the "IFRAME=*fileref* | '*external-file*'" on page 1278 option.

See To specify that a line is drawn around the axis area, use the "FRAME" on page 1276option.

HREVERSE

specifies that the order of the values on the horizontal axis be reversed. For web output that is generated with a Java device driver, the horizontal axis data must be numeric.

Restriction Partially supported by Java and ActiveX

HTML=*variable*

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction For Java and ActiveX, only the ACTXIMG device is supported for the BUBBLE2 statement.

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

HZERO

specifies that tick marks on the horizontal axis begin in the first position or end in the last position with a value of zero, depending on the horizontal variable values.

When all horizontal variable values are positive, the first tick mark on the horizontal axis has a value of zero. When all horizontal variable values are negative, the last tick mark on the horizontal axis has a value of zero.

The HZERO request is ignored if the horizontal variable contains a mix of positive and negative values. It is ignored when it is ordered with the HAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the horizontal variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the HZERO option is ignored, the last tick mark on the horizontal axis has a value of zero.

IFRAME=*fileref* | '*external-file*'

identifies the image file you that want to apply to the axis area of the plot.

Restriction Not supported by Java

Interactions This option is overridden by the NOIMAGEPRINT goption. For information about the NOIMAGEPRINT goption see "IMAGEPRINT, NOIMAGEPRINT Graphics Options" on page 653.

This option is ignored if you specify the NOFRAME option or if you specify the STYLE=0 option in the AXIS statement.

See "Displaying Images on Data Elements" on page 389

"IMAGESTYLE=TILE | FIT" on page 1279

IMAGESTYLE=TILE | FIT
specifies whether to use multiple instances of an image to fill the axis area (TILE) or to stretch a single instance of an image to fill the axis area (FIT). The TILE value is the default.

See "IFRAME=*fileref* | '*external-file*'" on page 1278

LAUTOHREF=*reference-line-type*
specifies the line type for reference lines at major tick marks on the horizontal axis, as specified by the AUTOHREF option. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style. If the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

Style reference LineStyle attribute of the GraphGridLines element

LAUTOVREF=*reference-line-type*
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

Style reference LineStyle attribute of the GraphGridLines element

LHREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines drawn perpendicular to the horizontal axis. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOHREF, HREF, and GRID options. Specifying without parentheses a single line type applies that line type to all reference lines. The LAUTOHREF= option overrides the LHREF= option for lines drawn with the AUTOHREF option. Specifying in parentheses a single line type applies that line type only to the first reference line drawn with the HREF= option. Specifying a line-type list applies line types in sequence to successive reference lines drawn with the HREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (*type1 type2 ... typeN*). The default line type is retrieved from the current style. If the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

Alias LH=

Style reference LineStyle attribute of the GraphGridLines element

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines drawn perpendicular to the vertical axis. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parenthesis a single line type applies that line type to all reference lines. The LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single line type applies that line type only to the first line drawn by the VREF= option. Specifying

a line-type list applies line types in sequence to successive reference lines drawn with the VREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (*type1 type2 ... typeN*). The default line type is retrieved from the current style. If the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these references lines, use the CVREF= option. To specify labels for these reference lines, use the VAXIS= option.

Alias LV=

Style reference LineStyle attribute of the GraphGridLines element

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
  - If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

  Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

  - The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GPLOT

NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias NOAXES

NOFRAME
specifies that a line is not drawn around the axis area. To override the default and specify that no line is to be drawn, use the NOFRAME option.

Alias NOFR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1274 option.

To fill the axis area with an image, use the "IFRAME=*fileref* | '*external-file*'" on page 1278 option.

See To specify that a line is drawn around the axis area, use the "FRAME" on page 1276option.

URL=*character-variable*
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only

Not supported by the JAVAIMG and ACTXIMG devices when using the BUBBLE2 statement

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

Note You can specify this option starting with SAS 9.4M1. The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

VAXIS=*value-list* | AXIS<1 – 99>
specifies the major tick mark values for the vertical axis or assigns an axis definition. For a description of the *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. To assign labels to reference lines, specify an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOVREF and VREF= options.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the "AXIS Statement" on page 399.

Restriction Partially supported by Java and ActiveX

Examples "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

"Example 3: Adding a Right Vertical Axis" on page 1342

VMINOR=*number-of-minor-ticks*
specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis. Minor tick marks are not labeled. The VMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias VM=

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

VREF=*value* | (*value*) | (*value-list*)
draws one or more reference lines perpendicular to the vertical axis at specified points. For a description of *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. The LVREF=, CVREF=, and WVREF= options can be used to change the line types, colors, and widths of these reference lines, respectively. To specify labels for these reference lines, use the VAXIS= option.

VREVERSE
specifies that the order of the values on the vertical axis should be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

When all vertical variable values are positive, the first tick mark on the vertical axis has a value of zero. When all vertical variable values are negative, the last tick mark on the vertical axis has a value of zero.

The VZERO request is ignored if the vertical variable contains a mix of positive and negative values. It is ignored when it is ordered with the VAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the vertical variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the VZERO option is ignored, the last tick mark on the vertical axis has a value of zero.

WAUTOHREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the horizontal axis as determined by the AUTOHREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option "CAUTOHREF=*reference-line-color*" on page 1274.

Style reference LineThickness attribute of the GraphGridLines element

WAUTOVREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these

reference lines, use the option "CAUTOVREF=*reference-line-color*" on page 1274.

Style reference LineThickness attribute of the GraphGridLines element

WHREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)

specifies line widths for reference lines as determined by the horizontal axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOHREF and HREF= options. Note that the WAUTOHREF= option overrides WHREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the HREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the HREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the option "CHREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)" on page 1274.

Style reference LineThickness attribute of the GraphGridLines element

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)

specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOVREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the VREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the option "CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)" on page 1275.

Style reference LineThickness attribute of the GraphGridLines element

Details

Description

The BUBBLE statement specifies one or more plot requests that name the horizontal and left vertical axis variables and the variable that controls the size of the bubbles. This statement automatically does the following:

- centers each circle at a data point that is determined by the values of the vertical and horizontal axes variables

- scales the axes to include the maximum and minimum data values
- labels each axis with the name of its variable or associated label
- displays each major tick mark value
- draws circles for values that are located within the axes

You can use statement options to control axis scaling, draw reference lines, or modify the appearance of axes. You can also use statement options to control the display of the bubbles, specify an axis area color or image, and specify annotation.

In addition, you can use global statements to modify axes (AXIS statement), and add text to the graph (TITLE, NOTE, and FOOTNOTE statements). You can also use the Annotate data set to enhance the plot.

Controlling the Display of Bubbles

The BUBBLE statement draws circles only for values that are located within the axes. Observations with values that lie outside of the axis area are not plotted. If a bubble size value causes a bubble to overlap the axis, the bubble is clipped against the axis line. The bubbles for the highest axis value and lowest axis value might be clipped unless you modify the axes in either of the following ways:

- by offsetting the first and last values
- by adding values to the range that is represented by the axis

Specify the range of values on an axis with the HAXIS= or VAXIS= option, or with AXIS definitions.

To add a right vertical axis, use a BUBBLE2 statement.

BUBBLE2 Statement

Creates a second vertical axis on the right side of a graph produced by an accompanying BUBBLE or PLOT statement. A second variable can be plotted against this axis.

Requirement: You cannot use the BUBBLE2 statement alone. You can use it only with a BUBBLE or PLOT statement. At least one plot request is required.

Supported Statements: AXIS, FOOTNOTE, TITLE

Syntax

BUBBLE2 *plot-request(s)* <*/ options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*

specifies a data set to annotate all graphs that are produced by the GPLOT procedure.

Bubble appearance options

BCOLOR=*bubble-color*
specifies the color for the bubbles.

BFILL=SOLID | GRADIENT
enables you to generate solid or gradient-filled bubbles.

BFONT=*font*
specifies the font to use for bubble labels.

BLABEL
labels the bubbles with the values of the third variable.

BSCALE=AREA | RADIUS
specifies whether the bubble-scaling proportion is based on the area of the circles or the radius measure.

BSIZE=*multiplier*
specifies an overall scaling factor for the bubbles so that you can increase or decrease the size of all bubbles by this factor.

ODS options

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Plot appearance options

CAXIS=*axis-color*
specifies the color for the axis line and all major and minor tick marks.

CFRAME=*background-color*
fills the axis area with the specified color.

CTEXT=*text-color*
specifies the color for all text on the axes, including tick mark values, axis labels, and bubble labels.

DATAORDER='*entry-description*'
plots character of midpoint-type data in data order instead of the default alphabetical order.

FRAME
specifies that a line is drawn around the axis area.

FRONTREF
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

GRID
draws reference lines at all major tick marks on both axes.

NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

NOFRAME
specifies that a line is not drawn around the axis area.

Vertical axis options

AUTOVREF

draws reference lines at all major tick marks on the vertical axis.

CAUTOVREF=*reference-line-color*
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option.

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the vertical axis.

LAUTOVREF=*reference-line-type*
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option.

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines drawn perpendicular to the vertical axis.

VAXIS=*value-list* | AXIS<1 – 99>
specifies the major tick mark values for the vertical axis or assigns an axis definition.

VMINOR=*number-of-minor-ticks*
specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis.

VREF=*value* | (*value*) | (*value-list*)
draws one or more reference lines perpendicular to the vertical axis at specified points.

VREVERSE
specifies that the order of the values on the vertical axis should be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

WAUTOVREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option.

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines as determined by the vertical axis.

Required Argument

plot-request(s)
each specifies the variables to plot and produces a separate graph. All variables must be in the input data set. Multiple plot requests are separated with blanks. A plot request must have this form:

y-variable*x-variable=bubble-size
plots the values of two variables and draws a circle (bubble) at each data point. The value of the third variable determines the size of the bubble. All of these variables must be in the input data set:

y-variable
variable plotted on the right vertical axis; typically it is different from *y-variable* in the accompanying BUBBLE or PLOT statement.

x-variable
variable plotted on the horizontal axis; it is the same as *x-variable* in the accompanying BUBBLE or PLOT statement.

bubble-size
specifies the size of the bubbles. *Bubble-size* must be numeric. If the value of *bubble-size* is positive, bubbles are drawn with a solid line. If it is negative, bubbles are drawn with a dashed line.

Optional Arguments

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

Alias ANNO=

See Chapter 28, "Using Annotate Data Sets," on page 721

AUTOVREF
draws reference lines at all major tick marks on the vertical axis. LAUTOVREF=, CAUTOVREF=, and WAUTOVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

BCOLOR=*bubble-color*
specifies the color for the bubbles. You might not specify the BCOLOR= option. In that case, the bubble color becomes the color of the default style (GSTYLE) or the color specified by the current ODS style (if used).

Style reference ContrastColor attribute of the GraphOutline, GraphData1, and TwoColorAltRamp elements

Examples "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

"Example 3: Adding a Right Vertical Axis" on page 1342

BFILL=SOLID | GRADIENT
enables you to generate solid or gradient-filled bubbles. By default, the Java and ActiveX devices create solid bubbles.

- BFILL=SOLID fills the bubbles with the color specified by the BCOLOR= option. If the BFILL option is not specified, then the color is specified by the current style. If you are using specific ODS style, the color comes from the contrast color attribute within the GraphData1 style element.
- BFILL=GRADIENT starts with the current axis area color and gradually transitions to the color specified with the BCOLOR= option or the color of the current style. If you are using an ODS style, the colors are controlled by the `startcolor` and `endcolor` attributes of the TwoColorAltRamp style element.

Restriction Not supported by Java and ActiveX

Note The SAS/GRAPH ActiveX control displays negative values as empty circles.

BFONT=*font*
specifies the font to use for bubble labels. See Chapter 22, "Specifying Fonts in SAS/GRAPH Programs," on page 349 for details about how to specify *font*. If you omit the BFONT= option, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the font specified by the current style
3. the default hardware font

Style reference Font attribute of the GraphValueText element

Restriction Not supported by Java and ActiveX

See the option "BLABEL" on page 1288 for information about the location and color of labels

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

BLABEL

labels the bubbles with the values of the third variable. If the variable has a format, the formatted value is used. By default, bubbles are not labeled.

The procedure normally places labels directly outside of the circle at 315 degrees rotation. If a label in this position does not fit in the axis area, other 45-degree placements (that is, 45, 135, and 225 degrees) are attempted. If the label cannot be placed at any of the positions (45, 135, 225, or 315 degrees) without being clipped, the label is omitted. However, labels can collide with other bubbles or previously placed labels.

Labels are displayed in the color specified by the CTEXT= option. If you omit the CTEXT=option, the default is the color of the current style.

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

BSCALE=AREA | RADIUS

specifies whether the bubble-scaling proportion is based on the area of the circles or the radius measure. By default, BSCALE=AREA.

The value that is assigned to the BSCALE= option affects how large the bubbles appear in relation to each other. For example, suppose the third variable value is twice as big for one bubble as it is for another. If BSCALE=AREA, the area of the larger bubble is twice the area of the smaller bubble. If BSCALE=RADIUS, the radius of the larger bubble is twice the radius of the smaller bubble and the larger bubble has more than twice the area of the smaller bubble.

Restriction Not supported by Java and ActiveX

BSIZE=*multiplier*

specifies an overall scaling factor for the bubbles so that you can increase or decrease the size of all bubbles by this factor.

In web output, the Java applets and the ActiveX Control override the default value. To prevent this override, specify a value for the BSIZE= option, rather than relying on the default value.

Restriction Partially supported by Java and ActiveX

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

CAUTOVREF=*reference-line-color*

specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option. If you do not specify the CAUTOREF option, the default color is the value of the CAXIS= option. If neither

option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Style reference Color attribute of the GraphGridLines element

CAXIS=*axis-color*
specifies the color for the axis line and all major and minor tick marks. By default, the procedure uses the color of the current style.

The CAXIS= option is overridden by the COLOR= option in an AXIS definition. The COLOR= option in an AXIS definition is overridden by the COLOR= suboption of the MAJOR= and MINOR= options in an AXIS definition.

Alias CA=

Style reference Color attribute of the GraphAxisLines attribute

CFRAME=*background-color*
fills the axis area with the specified color. If the FRAME option is also in effect, the procedure determines the color of the frame according to the precedence list given for the FRAME option description. If the IFRAME= option is in effect, the specified image fills the axis area instead of the specified color.

Style reference Color attribute of the GraphWalls element

CTEXT=*text-color*
specifies the color for all text on the axes, including tick mark values, axis labels, and bubble labels. The GPLOT procedure searches for a color specification in this order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the PLOT statement
2. the color specified by the CTEXT= option in the PLOT statement
3. the color specified by the CTEXT= option in the GOPTIONS statement
4. the color specified in the current style, or a NOGSTYLE option specification. With NOGSTYLE specified, the default color is black for the Java and ActiveX devices and the first color in the color list for all other devices

In an AXIS statement, the COLOR= suboption of either a LABEL= option or a VALUE= option overrides the CTEXT= option. It also determines the color of the axis label, or the color of the tick mark values, respectively.

Alias C=, CT=

Style reference Color attributes of the GraphValueText and the GraphLabelText elements

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the vertical axis. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the VREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the VREF= option. The syntax of the color list requires parentheses and each color separated by spaces (*color1 color2 ... colorN*). If you do not

specify the CVREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias CV=

DATAORDER='*entry-description*'
plots character of midpoint-type data in data order instead of the default alphabetical order.

Restriction Supported by Java and ActiveX only

FRAME
specifies that a line is drawn around the axis area. For the frame color, a specification is searched for in this order:

1. the CAXIS= option
2. the COLOR= option in the AXIS definition assigned to the vertical axis
3. the COLOR= option in the AXIS definition assigned to the horizontal axis
4. the default, which is the color defined by the current style

Alias FR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1289 option.

To fill the axis area with an image, use the "IFRAME=*fileref* | '*external-file*'" on page 1278 option.

See To override the default and specify that no line is drawn around the axis area, use the "NOFRAME " on page 1291 option.

FRONTREF
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars. By default, reference lines are drawn on the back plane of the axis.

GRID
draws reference lines at all major tick marks on both axes. You get the same result when you use all of these options in a BUBBLE statement: AUTOHREF, AUTOVREF, FRAME, LVREF=34, and LHREF=34. The line type for GRID is 34.

The line color is the color of the axis.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction For Java and ActiveX, only the ACTXIMG device is supported for the BUBBLE2 statement.

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

LAUTOVREF=*reference-line-type*

specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

Style reference LineStyle attribute of the GraphGridLines element

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)

specifies line types for reference lines drawn perpendicular to the vertical axis. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parenthesis a single line type applies that line type to all reference lines. The LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single line type applies that line type only to the first line drawn by the VREF= option. Specifying a line-type list applies line types in sequence to successive reference lines drawn with the VREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (*type1 type2 ... typeN*). The default line type is retrieved from the current style. If the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these references lines, use the CVREF= option. To specify labels for these reference lines, use the VAXIS= option.

Alias LV=

Style reference LineStyle attribute of the GraphGridLines element

NOAXIS

suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias NOAXES

NOFRAME

specifies that a line is not drawn around the axis area. To override the default and specify that no line is to be drawn, use the NOFRAME option.

Alias NOFR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1289 option.

To fill the axis area with an image, use the "IFRAME=*fileref* | '*external-file*'" on page 1278 option.

See To specify that a line is drawn around the axis area, use the "FRAME" on page 1290option.

URL=*character-variable*

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only

Not supported by the JAVAIMG and ACTXIMG devices when using the BUBBLE2 statement

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

Note You can specify this option starting with SAS 9.4M1. The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

VAXIS=*value-list* | AXIS<1 – 99>

specifies the major tick mark values for the vertical axis or assigns an axis definition. For a description of the *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. To assign labels to reference lines, specify an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOVREF and VREF= options.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the "AXIS Statement" on page 399.

Restriction Partially supported by Java and ActiveX

Examples "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

"Example 3: Adding a Right Vertical Axis" on page 1342

VMINOR=*number-of-minor-ticks*

specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis. Minor tick marks are not labeled. The VMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias VM=

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

VREF=*value* | (*value*) | (*value-list*)
draws one or more reference lines perpendicular to the vertical axis at specified points. For a description of *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. The LVREF=, CVREF=, and WVREF= options can be used to change the line types, colors, and widths of these reference lines, respectively. To specify labels for these reference lines, use the VAXIS= option.

VREVERSE
specifies that the order of the values on the vertical axis should be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

When all vertical variable values are positive, the first tick mark on the vertical axis has a value of zero. When all vertical variable values are negative, the last tick mark on the vertical axis has a value of zero.

The VZERO request is ignored if the vertical variable contains a mix of positive and negative values. It is ignored when it is ordered with the VAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the vertical variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the VZERO option is ignored, the last tick mark on the vertical axis has a value of zero.

WAUTOVREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option "CAUTOVREF=*reference-line-color*" on page 1288.

Style reference LineThickness attribute of the GraphGridLines element

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOVREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the VREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the option "CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)" on page 1289.

Style reference LineThickness attribute of the GraphGridLines element

Details

Options

Options for the BUBBLE2 statement are identical to the options for the BUBBLE statement with exception of the following, which are ignored if specified:

- AUTOHREF
- CAUTOHREF=
- CHREF=
- DESCRIPTION=
- HAXIS=
- HMINOR=
- HREF=
- HZERO=
- IFRAME=
- IMAGESTYLE =
- LAUTOHREF=
- LHREF=
- NAME=
- WAUTOHREF=
- WHREF=

See "BUBBLE2 Statement" on page 1284 for complete descriptions of options that you can use with the BUBBLE2 statement.

Description

The BUBBLE2 statement specifies one or more plot requests that name the horizontal and right vertical axis variables and the variable that controls the size of the bubbles. This statement automatically does the following:

- scales the axes to include the maximum and minimum data values
- labels each axis with the name of its variable or an associated label
- displays each major tick mark value
- draws circles for values that are located within the axes

You can use statement options to control right vertical axis scaling, draw reference lines on the right vertical axis, and control the display of the bubbles. You can also use statement options to display an axis area color or image, and specify annotation.

In addition, you can use global statements to modify the axes (AXIS statement), and add text to the graph (TITLE, NOTE, and FOOTNOTE statements). You can also use the Annotate data set to enhance the plot.

Coordinating BUBBLE and BUBBLE2 Plot Requests

The BUBBLE2 statement draws circles only for values that are located within the axes. Bubbles are not drawn for values that lie outside of the axis range. If a bubble size value causes a bubble to overlap the axis, the bubble is clipped against the axis line.

In the BUBBLE2 statement, either the *y-variable* or *bubble-size* can differ from the variables in the BUBBLE statement. Here are some possible combinations of plot requests for BUBBLE and BUBBLE2 statement pairs and how they affect the plot:

- The vertical axis variables Y and Y2 are different, but the bubble size variable, S, is the same in both:

  ```
  bubble y*x=s;
     bubble2 y2*x=s;
  ```

  These plot requests generate a plot in which both sets of bubbles have the same value (size) but different locations on the graph.

- The vertical axis variables are the same, Y, but the bubble size variables, S and S2, are different:

  ```
  bubble y*x=s;
     bubble2 y*x=s2;
  ```

  The resulting plot has two identical vertical axes and two sets of concentric bubbles of different sizes.

- Both the vertical axis variables, Y and Y2, and the bubble size variables, S and S2, are different:

  ```
  bubble y*x=s;
     bubble2 y2*x=s2;
  ```

  These plot requests produce the equivalent of an overlay plot in which two different sets of bubbles plotted against different vertical axes are displayed on the same graph.

The plot requests in the BUBBLE and BUBBLE2 statements must be evenly matched, for example:

```
bubble  y*x=s  b*a=c;
      bubble2 y2*x=s b2*a=c2;
```

These statements produce two graphs, and each has two vertical axes. The first pair of plot requests (Y*X=S and Y2*X=S) produce one graph. In this graph the variable X is plotted on the horizontal axis. The variable Y is plotted on the left axis and the variable Y2 is plotted on the right axis. In this pair, the value of S is the same for both requests. The second pair of plot requests (B*A=C and B2*A=C2) produce another graph. In this graph the variable A is plotted on the horizontal axis. The variable B is plotted on the left axis and the variable B2 is plotted on the right axis.

Any modifications to horizontal axes specifications must be identical for both statements. If they are different, the BUBBLE2 axis specification is ignored.

There might be a case where the scale of values for the left and right vertical axes is the same. You also want both axes to represent the same range of values. You would specify the range with a VAXIS= option in both the BUBBLE and BUBBLE2 statements.

PLOT Statement

Creates plots in which one variable is plotted on the horizontal axis and a second variable is plotted on the left vertical axis.

Requirement: At least one plot request is required.

Supported Statements: AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE

Supports: Drill-down functionality

Syntax

PLOT *plot-request(s)* <*/ options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate all graphs that are produced by the GPLOT procedure.

CAXIS=*axis-color*
: specifies the color for the axis line and all major and minor tick marks.

CBASELINE=*baseline-color*
: specifies the color of the horizontal baseline for reference lines of a plot.

CFRAME=*background-color*
: fills the axis area with the specified color.

COUTLINE=*outline-color*
: specifies the color of the outline that is drawn around filled areas.

CTEXT=*text-color*
: specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions.

FRAME
: specifies that a line is drawn around the axis area.

FRONTREF
: specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

IFRAME=*fileref* | '*external-file*'
: identifies the image file that you want to apply to the axis area of the plot.

IMAGESTYLE=TILE | FIT
: specifies whether to use multiple instances of the image to fill the axis area (TILE) or to stretch a single instance of the image to fill the (FIT). The TILE value is the default.

NOAXIS

suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

NOFRAME
specifies that a line is not drawn around the axis area.

Catalog options

DESCRIPTION="*description*"
specifies a description of the output.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Horizontal axis options

AUTOHREF
draws reference lines at all major tick marks on the horizontal axis.

CAUTOHREF=*reference-line-color*
specifies colors for reference lines drawn at major tick marks on the horizontal axis, as specified by the AUTOHREF option.

CHREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the horizontal axis.

HAXIS=*value-list* | AXIS<1 – 99>
specifies major tick mark values for the horizontal axis or assigns an axis definition.

HMINOR=*number-of-minor-ticks*
specifies the number of minor tick marks drawn between each major tick mark on the horizontal axis.

HREF=*value* | (*value*) | (*value-list*)
draws one or more reference lines perpendicular to the horizontal axis at points that are specified by *value-list*.

HREVERSE
specifies that the order of the values on the horizontal axis be reversed.

HZERO
specifies that tick marks on the horizontal axis begin in the first position or end in the last position with a value of zero, depending on the horizontal variable values.

LAUTOHREF=*reference-line-type*
specifies a line type for reference lines drawn at major tick marks on the horizontal axis, as specified by the AUTOHREF option.

LHREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines drawn perpendicular to the horizontal axis.

WAUTOHREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the horizontal axis as determined by the AUTOHREF option.

WHREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines as determined by the horizontal axis.

ODS options

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Plot options

AREAS=*n*
fills all the areas below plot line *n* with a pattern.

GRID
draws reference lines at all major tick marks on both axes.

LEGEND

LEGEND=LEGEND*1–99*
generates a legend or specifies the defined legend to use for the plot.

NOLASTAREA
removes the last area of a plot when the value specified by the AREAS= option exceeds the number of bounded areas in the plot.

NOLEGEND
suppresses the legend that is generated by a plot request of the type *y-variable*x-variable=third-variable*.

OVERLAY
places all the plots that are generated by the PLOT statement on one set of axes.

REGEQN
displays the regression equation that is specified in the INTERPOL= option of the SYMBOL statement in the lower left corner of the plot.

SKIPMISS
breaks a plot line or an area fill at occurrences of missing values of the Y variable.

Vertical axis options

AUTOVREF
draws reference lines at all of the major tick marks on the vertical axis.

CAUTOVREF=*reference-line-color*
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option.

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the vertical axis.

LAUTOVREF=*reference-line-type*
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option.

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies the type of reference lines drawn with the AUTOVREF, VREF, and GRID options.

VAXIS=*value-list* | AXIS<1 – 99>
specifies the major tick mark values for the vertical axis or assigns an axis definition.

VMINOR=*number-of-minor-ticks*

specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis.

VREF=*value* | (*value*) | (*value-list*)
draws one or more reference lines perpendicular to the vertical axis at specified points.

VREVERSE
specifies that the order of the values on the vertical axis be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

WAUTOVREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option.

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines as determined by the vertical axis.

Required Argument

plot-request(s)
each specifies the variables to plot and produces a separate graph, unless you specify OVERLAY. All variables must be in the input data set. Multiple plot requests are separated with blanks. You can plot character or numeric variables. A plot request can be any of these:

y-variable*x-variable<=n>
plots the values of two variables and can assign a SYMBOL definition to the plot.

y-variable
variable plotted on the left vertical axis.

x-variable
variable plotted on the horizontal axis.

n
number of the *n*th generated SYMBOL definition.

Note The *n*th generated SYMBOL definition is not necessarily the same as the *n*th SYMBOL statement. Plot requests of the form *y-variable*x-variable=n* assign the SYMBOL definition that is designated by *n* to the plot that is produced by *y-variable*x-variable*. For more information, see "About Plot Requests That Assign a SYMBOL Definition" on page 1318.

(*y-variable(s)*)*(*x-variable(s)*)
plots the values of two or more variables and produces a separate graph for each combination of Y and X variables. That is, each Y*X pair is plotted on a separate set of axes unless you specify OVERLAY.

y-variable(s)
variables plotted on the left vertical axes.

x-variable(s)
variables plotted on the horizontal axes.

If you use only one *y-variable* or only one *x-variable*, omit the parentheses for that variable, for example:

```
plot (temp rain)*month;
```

This plot request produces two plots, one of TEMP and MONTH and one of RAIN and MONTH.

y-variable*x-variable=third-variable
plots the values of two variables against a third classification variable

y-variable
variable plotted on the left vertical axis.

x-variable
variable plotted on the horizontal axis.

third-variable
classification variable against which *y-variable* and *x-variable* are plotted. *Third-variable* can be character or numeric, but numeric variables should contain discrete rather than continuous values, or should be formatted to provide discrete values.

A separate plot (set of data points) is produced for each unique value of *third-variable*. This means that all plots are drawn on the same set of axes. Also, a legend is automatically generated to show the plot symbol and color for each value of the classification variable.

The following plot request produces a graph with a plot line for each department and a legend that shows the plot symbol for each department:

```
plot sales*weekday=dept;
```

For an example of a plot that specifies a *third-variable*, see "Example 8: Plotting Three Variables" on page 1358.

Note If a BY statement is used to produce multiple plots, you can make the legend identical across graphs by specifying the UNIFORM option in the PROC GPLOT statement.

You can use more than one type of plot request in a single PLOT statement (provided that you do not specify OVERLAY), for example:

```
plot temp*month rain*month=2;
```

Optional Arguments

Options in a PLOT statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

Alias ANNO=

See Chapter 28, "Using Annotate Data Sets," on page 721

AREAS=*n*
fills all the areas below plot line *n* with a pattern. The value of *n* specifies which areas to fill:

- AREAS=1 fills the first area.
- AREAS=2 fills both the first and second areas, and so on.

You might specify a value for the AREAS= option that is greater than the number of bounded areas in the plot. In this case, the area between the top plot line and the axis frame is filled.

Before an area can be filled, the data points that border the area must be joined by a line. Use a SYMBOL statement with one of these interpolation methods to join the data points:

- INTERPOL=JOIN
- INTERPOL=STEP
- INTERPOL=R*series*
- INTERPOL=SPLINE | SM | L

See the "SYMBOL Statement" on page 470 for details about interpolation methods.

By default, the AREAS= option fills areas by rotating a solid fill through the list of colors defined in the current style. If the NOGSTYLE option is specified, the areas are filled by rotating a solid fill through the device's color list. If the graph needs more patterns, it rotates hatch patterns, beginning with the M2N0 pattern. See the "PATTERN Statement" on page 455 for more information about map and plot patterns. However, if color is limited to a single color with the CPATTERN= or COLORS= graphic options, the solid pattern is skipped and the first default pattern is M2N0. If the COLORS= graphic option specifies a single color, use as many SYMBOL statements as you have areas to fill. You must do this because the INTERPOL= setting does not automatically apply to multiple symbol definitions.

You can alter the default pattern behavior by specifying patterns and colors in PATTERN statements that specify map and plot patterns. A separate PATTERN definition is needed for each specified area.

If you specify the PATTERN statements, the AREAS= option uses the lowest numbered PATTERN statement first. If it runs out of patterns, it uses the default behavior for map and plot patterns. See the "PATTERN Statement" on page 455 for details.

Pattern definitions are assigned to the areas below the plot lines in the order the plots are drawn. The first area is that between the horizontal axis and the plot line that is drawn first. The second area is that above the first plot line and below the plot line that is drawn second, and so on. If the line that is drawn second lies below the line that is drawn first, the second area is hidden when the first is filled. The plots with the lower line values must be drawn first to prevent one area fill from overlaying another. If the lines cross, only the part of an area that is above the previous line is visible.

Therefore, you must order your PLOT statements when creating multiple plots in combination with the OVERLAY option. Order them so that the plot request that produces the lowest line value is first (leftmost). The plot request that produces the next lowest line value is the second plot request, and so on.

If you produce multiple plots with a *y-variable*x-variable=third-variable* plot request, the lines are plotted in order of increasing third variable values. In that case, you must recode your data so that the lowest value of the third variable produces the lowest plot line. Likewise, the next lowest value produces the next lowest plot line, and so on.

If you use the VALUE= option in the SYMBOL statement, some symbols might be hidden. If you also specify reference lines with the AREAS= option, they are drawn behind the pattern fill.

Restriction Partially supported by Java

Notes The AREAS= option works only if all plot lines are generated by the same PLOT or PLOT2 statement.

If you have specified the NOGSTYLE option and the first color in your device's default color list is black, color rotation begins with the second color in the list. That is, there are no solid black patterns. See "How Default Patterns and Outlines Are Generated" on page 465 for more information.

Example "Example 7: Filling Areas in an Overlay Plot" on page 1355

AUTOHREF
draws reference lines at all major tick marks on the horizontal axis. If the AREAS= option is also used, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option. LAUTOHREF=, CAUTOHREF=, and WAUTOHREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the HAXIS= option.

AUTOVREF
draws reference lines at all of the major tick marks on the vertical axis. If you also use the AREAS= option, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option in either the PROC GPLOT statement or the PLOT statement. LAUTOVREF=, CAUTOVREF=, and WAUTOVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

CAUTOHREF=*reference-line-color*
specifies colors for reference lines drawn at major tick marks on the horizontal axis, as specified by the AUTOHREF option. The default color is either the value of the CAXIS= option or the first color in the color list.

CAUTOVREF=*reference-line-color*
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option. If you do not specify the CAUTOVREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

CAXIS=*axis-color*
specifies the color for the axis line and all major and minor tick marks. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Alias CA=

CBASELINE=*baseline-color*
specifies the color of the horizontal baseline for reference lines of a plot. This option compensates for a reference line obscuring the axis. This can occur when using a combination of the AUTOVREF option, the CAUTOVREF= option, or the CVREF= option. They must be specifying any color other than black, and a vertical axis with no offset. For example, the light gray reference line drawn by

the AUTOVREF option and a CVREF= graydd specification obscures the default black color of the axis line. This option enables you to set the color of the baseline to match the axis color.

Specify the *baseline-color* as a name or a quoted string. Two examples are presented. The first indicates a name. The second specifies an RGB color code:

```
cbaseline=black
```

```
cb='CX0BB5FF'
```

Alias CB=

Default The value of the CAXIS= option is the default color if you do not specify an alternate color for the baseline. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Restriction This option is not supported by the ACTIVEX, ACTXIMG, and JAVAIMG devices.

Note The baseline is drawn after the plot elements are drawn, and either a default or specified color is applied. This means that the reference lines are drawn behind plot elements such as bars. Use the FRONTREF option if you want reference lines drawn in front of the bars.

See "Specifying Colors in SAS/GRAPH Programs" on page 364

"FRONTREF" on page 1306

CFRAME=*background-color*
fills the axis area with the specified color.

Style reference Color attribute of the GraphWalls element

Restriction CFRAME= is overridden by the NOFRAME option and the PLOT statement's IFRAME= option. If the FRAME= option is in effect, an image appears in the axis area instead of the color.

Interaction If the FRAME option is also in effect, the procedure behavior is modified. The procedure determines the color of the frame around the axis area according to the precedence list given within the FRAME option description.

CHREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the horizontal axis. This option affects reference lines drawn with the AUTOHREF, HREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOHREF= option overrides the CHREF= option for reference lines drawn with the AUTOHREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the HREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the HREF= option. The syntax of the color list requires parentheses and colors separated by spaces (*color1 color2 ...colorN*). If you do not specify the CHREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias CH=

Style reference Color attribute of the GraphGridLines element

COUTLINE=*outline-color*

specifies the color of the outline that is drawn around filled areas. The filled areas are generated when the SYMBOL statement specifies " INTERPOL=*map/plot-pattern* " on page 480 or the GOPTIONS statement specifies "INTERPOL Graphics Option" on page 655. The default outline color is specified in the current style. However, the NOGSTYLE option might be specified. In that case, the default color is the first color in the device's color list (the foreground color). Also, the default slice outline color is determined as follows:

- If you do not specify a PATTERN statement, the default outline color is the color defined in the current style.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

Alias CO=

Style reference Color attribute of the GraphOutlines element

Restriction Not supported by Java

Interaction The COUTLINE= option overrides the SYMBOL statement option CO=.

CTEXT=*text-color*

specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions. The GPLOT procedure searches for a color specification in this order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements. They override the CTEXT= option specified in the PLOT statement. This also includes colors specified by the POINTLABEL= option in the SYMBOL statement.
2. the color specified by the CTEXT= option in the PLOT statement
3. the color specified by the CTEXT= option in the GOPTIONS statement
4. the color specified in the current style, or a NOGSTYLE option specification. With NOGSTYLE specified, the default color is black for the Java and ActiveX devices and the first color in the color list for all other devices

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. The COLOR= color is still used for all other axis colors, such as tick marks.

Alias C=

Style reference Color attributes of the GraphValueText and the GraphLabelText elements

Note If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the vertical axis. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the VREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the VREF= option. The syntax of the color list requires parentheses and colors separated by spaces (*color1 color2 ... colorN*). If you do not specify the CVREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias CV=

Style reference Color attribute of the GraphGridLines element

DESCRIPTION="*description*"
specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default Based on whether a SYMBOL definition < =*n*> is specified in the plot request, either PLOT OF *chart-variable* by *chart-variable* or PLOT OF *chart-variable* by *chart-variable* identified by *chart-variable*

FRAME

specifies that a line is drawn around the axis area. For the frame color, a specification is searched for in this order:

1. the CAXIS= option
2. the COLOR= option in the AXIS definition assigned to the vertical axis
3. the COLOR= option in the AXIS definition assigned to the horizontal axis
4. the default, which is the color defined by the current style

Alias FR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1303 option.

To fill the axis area with an image, use the PLOT statement's "IFRAME=fileref | 'external-file'" option.

See To override the default and specify that no line is drawn around the axis area, use the "NOFRAME" on page 1312option.

FRONTREF

specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars. By default, reference lines are drawn on the back plane of the axis.

GRID

draws reference lines at all major tick marks on both axes. The line color is the color of the axis. When specified in a PLOT2 statement, the reference lines are drawn on the vertical axis on the right side of the plot.

HAXIS=*value-list* | AXIS<1 – 99>

specifies major tick mark values for the horizontal axis or assigns an axis definition. By default, the procedure scales the axis and provides an appropriate number of tick marks. To assign labels to horizontal reference lines, use an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOHREF and HREF= options.

The way you specify *value-list* depends on the type of variable:

- For numeric variables, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - *n* < ...*n*>
  - *n* TO *n* <BY *increment*>
  - *n* < ...*n*> TO *n* <BY *increment* > <*n* < ...*n*> >

  If a numeric variable has an associated format, the specified values must be the unformatted values.

- For date-time values, *value-list* includes any SAS date, time, or datetime value described for the SAS functions INTCK and INTNX, shown here as *SAS-value*:
  - '*SAS-value*'*i* < ...'*SAS-value*'*i*>
  - '*SAS-value*'*i* TO '*SAS-value*' *i*<BY *interval*>
- For character variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:
  - *'value-1'* < ...*'value-n'*>

  If a character variable has an associated format, the specified values must be the formatted values.

  You are able to specify up to 256 characters for each variable value.

For a complete description of *value-list*, see " ORDER=(*value-list*)" on page 407 in the AXIS statement.

If data values fall outside of the range that is specified by the HAXIS= option, then by default the outlying data values are not used in interpolation calculations. See "About the Input Data Set" on page 1264 for more information about values out of range.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the "AXIS Statement" on page 399.

Restriction Partially supported by Java and ActiveX

Examples "Example 4: Plotting Two Variables" on page 1345

"Example 5: Connecting Plot Data Points" on page 1348

"Example 9: Plotting with Different Scales of Values" on page 1363

HMINOR=*number-of-minor-ticks*

specifies the number of minor tick marks drawn between each major tick mark on the horizontal axis. Minor tick marks are not labeled. The HMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias HM=

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

HREF=*value* | (*value*) | (*value-list*)

draws one or more reference lines perpendicular to the horizontal axis at points that are specified by *value-list*. For a description of *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. To specify labels for these reference lines, use the HAXIS= option.

Values can be listed in any order, but should be within the range of values represented by the chart response axis. A warning is written to the SAS log if any of the points are off of the axis, and no reference line is drawn for such points. You can use the AUTOREF option to draw reference lines automatically at all of the major tick marks

If the AREAS= option is also used, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option.

The LHREF=, CHREF=, and WHREF= options can be used to change the line types, colors, and widths of these reference lines.

HREVERSE

specifies that the order of the values on the horizontal axis be reversed. For web output that is generated with a Java device driver, the horizontal axis data must be numeric. To specify line widths for these reference lines, use the WAUTOHREF= option.

Restriction Partially supported by Java and ActiveX

HTML=*variable*

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

For Java and ActiveX, the PLOT statement is partially supported by the ACTIVEX device. It is fully supported by the JAVAIMG and ACTXIMG devices. The PLOT2 statement is supported only by the ACTXIMG device.

Restriction Partially supported by Java and ActiveX

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java and ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

Example "Example 10: Creating Plots with Drill-down Functionality for the Web" on page 1367

HZERO

specifies that tick marks on the horizontal axis begin in the first position or end in the last position with a value of zero, depending on the horizontal variable values. When all horizontal variable values are positive, the first tick mark on the horizontal axis has a value of zero. When all horizontal variable values are negative, the last tick mark on the horizontal axis has a value of zero.

The HZERO request is ignored if the horizontal variable contains a mix of positive and negative values. It is ignored when it has been ordered with the HAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the horizontal variable contains any time-formatted values, such as date, time, or

datetime. In most of these cases when the HZERO option is ignored, the last tick mark on the horizontal axis has a value of zero.

IFRAME=*fileref* | '*external-file*'
identifies the image file that you want to apply to the axis area of the plot. *fileref* must be a valid SAS fileref up to eight characters long and must have been previously assigned with a FILENAME statement. *external-file* must specify the complete filename of the image file that you want to use. The format of *external-file* varies across operating environments.

Restriction Not supported by Java

Interactions This option is overridden by the NOIMAGEPRINT goption. For information about the NOIMAGEPRINT goption, see "IMAGEPRINT, NOIMAGEPRINT Graphics Options" on page 653.

This option is ignored if you specify the NOFRAME option or if you specify the STYLE=0 option in the AXIS statement.

See the IMAGESTYLE= option and "Displaying an Image in Graph Frame" on page 387.

IMAGESTYLE=TILE | FIT
specifies whether to use multiple instances of the image to fill the axis area (TILE) or to stretch a single instance of the image to fill the (FIT). The TILE value is the default.

Restriction Not supported by Java

See "IFRAME=*fileref* | '*external-file*'" on page 1309

LAUTOHREF=*reference-line-type*
specifies a line type for reference lines drawn at major tick marks on the horizontal axis, as specified by the AUTOHREF option. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

LAUTOVREF=*reference-line-type*
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

Style reference LineStyle attribute of the GraphGridLines element

LEGEND | LEGEND=LEGEND*1–99*
generates a legend or specifies the defined legend to use for the plot. Use LEGEND to produce a default legend.

Use the LEGEND option with an assigned number to first define the legend characteristics, and then assign the defined LEGEND*1–99* statement to the plot. Use any number in the range of 1–99, for example, LEGEND1.

Interaction A PLOT statement that includes the OVERLAY option does not automatically generate a legend. In these plot types, use LEGEND to produce a default legend, or LEGEND=LEGEND*1–99* to assign a

defined LEGEND statement to the plot. The default legend is centered below the axis frame and identifies which colors and plot symbols represent the *y-variables* that you specify for the plots. To control the order of the legend entries for overlaid plots, use the ORDER= option in the LEGEND statement. Specify the list of variables in quotation marks in the preferred order. For example, the following causes the legend entry for y3 to be displayed first, y1 next, and y2 last:

```
legend1 order=('y3' 'y1' 'y2');
proc gplot data=mydata2;
plot (y1 y2 y3)*x / overlay  legend=legend1;
run;
```

Note If you use the SHAPE= option in a LEGEND statement, the value SYMBOL is valid. If you use the PLOT statement's AREAS= option, SHAPE=BAR is also valid.

Tip A plot request of the form *y-variable*x-variable=third-variable* automatically generates a default legend that identifies which colors and plot symbols represent each value of the classification variable. In these plot types, override the default by using LEGEND=LEGEND*1–99* to assign a defined LEGEND statement to the plot.

See " ORDER=(*value-list*) | DESCENDING" on page 438 for more information.

"LEGEND Statement" on page 432

Example "Example 6: Generating an Overlay Plot" on page 1351

LHREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies line types for reference lines drawn perpendicular to the horizontal axis. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOHREF, HREF, and GRID options. Specifying without parentheses a single line type applies that line type to all reference lines. The LAUTOHREF= option overrides the LHREF= option for lines drawn with the AUTOHREF option. Specifying in parentheses a single line type applies that line type only to the first reference line drawn with the HREF= option. Specifying a line-type list applies line types in sequence to successive reference lines drawn with the HREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (*type1 type2* ... *typeN*). The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these references lines, use the CHREF= option. To specify labels for these reference lines, use the HAXIS= option.

Alias LH=

Style reference LineStyle attribute of the GraphGridLines element

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
specifies the type of reference lines drawn with the AUTOVREF, VREF, and GRID options. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID

options. Specifying without parentheses a single line type applies that line type to all reference lines. The LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single line type applies that line type only to the first line drawn with the VREF= option. Specifying a line-type list applies line types in sequence to successive reference lines drawn with the VREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (*type1 type2 ... typeN*). The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these references lines, use the CVREF= option. To specify labels for these reference lines, use the VAXIS= option.

You might generate a needle plot with a Java or ActiveX device driver. In this case, the value of the LVREF= option is not applied to the default reference line that is drawn at zero. This occurs when the minimum value of the vertical axis is less than zero. This line is solid (not dashed).

Alias	LV=
Style reference	LineStyle attribute of the GraphGridLines element
Restriction	Partially supported by Java and ActiveX
Example	"Example 5: Connecting Plot Data Points" on page 1348

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
  - If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.

- If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

  Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GPLOT

NOAXIS

suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias NOAXES

NOFRAME

specifies that a line is not drawn around the axis area.

To override the default and specify that no line is to be drawn, use the NOFRAME option.

Alias NOFR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1303 option.

To fill the axis area with an image, use the PLOT statement's "IFRAME=*fileref* | '*external-file*'" on page 1309 option.

See To specify that a line is drawn around the axis area, use the "FRAME" on page 1306option.

NOLASTAREA

removes the last area of a plot when the value specified by the AREAS= option exceeds the number of bounded areas in the plot. The last area is the area between the last plot line and the axis frame. You can use the NOLASTAREA option in PLOT and PLOT2 statements. However, when both PLOT and PLOT2 statements exist, the NOLASTAREA option is ignored in the PLOT statement. The NOLASTAREA option is also ignored in both PLOT and PLOT2 statements when the AREAS= option is not specified. It is also ignored when the number of bounded areas in the plot is less than or equal to the value specified by the AREAS= option.

Alias NOLASTFILL

Restriction Not supported by Java and ActiveX.

Note Certain conditions related to the PLOT2 statement might result in variations in the behavior of the NOLASTAREA option.

See "AREAS=*n*" on page 1300

NOLEGEND

suppresses the legend that is generated by a plot request of the type *y-variable*x-variable=third-variable*.

OVERLAY

places all the plots that are generated by the PLOT statement on one set of axes. The axes are scaled to include the minimum and maximum values of all of the variables. They also include the variable names or labels associated with the first pair of variables label the axes.

The OVERLAY option produces a legend if you include the LEGEND option or the LEGEND=*n* option in the PLOT statement.

OVERLAY is not enabled with plot requests of the form *y-variable*x-variable=third-variable*. However, you can achieve an overlay effect by using a PLOT and PLOT2 statement.

When generating output for the web with the JAVAIMG device driver, the following condition applies. The OVERLAY option cannot be used in the PLOT or PLOT2 statement if the PLOT or PLOT2 statement is combined with the global SYMBOL statement when the SYMBOL statement uses the INTERPOL= BOX, HILO, or STD.

Restriction Partially supported by Java

Examples "Example 6: Generating an Overlay Plot" on page 1351

"Example 7: Filling Areas in an Overlay Plot" on page 1355

REGEQN

displays the regression equation that is specified in the INTERPOL= option of the SYMBOL statement in the lower left corner of the plot. You cannot modify the format that is used for the equation.

The GPLOT regression equation is computed from the screen coordinates of the markers. Therefore, a graph might not be displayed if the chart area for the plot becomes so small that markers cannot be drawn. This is because there are no coordinates from which to build the regression equation. In such cases, the regression equation is no longer meaningful.

Restriction Not supported by ActiveX

Example "Example 4: Plotting Two Variables" on page 1345

SKIPMISS

breaks a plot line or an area fill at occurrences of missing values of the Y variable. By default, plot lines and area fills are not broken at missing values. The SKIPMISS option is available only with JOIN interpolation. If the SKIPMISS option is used, observations should be sorted by the independent (horizontal axis) variable. If the plot request is *y-variable*x-variable=third-variable*, observations should also be sorted by the values of the third variable.

See "About the Input Data Set" on page 1264

URL=*character-variable*

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only

Not supported by the JAVAIMG and ACTXIMG devices when using the PLOT2 statement

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

Note You can specify this option starting with SAS 9.4M1. The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

VAXIS=*value-list* | AXIS<1 – 99>

specifies the major tick mark values for the vertical axis or assigns an axis definition. For a description of *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. To assign labels to reference lines, use an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOVREF and VREF= options.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the "AXIS Statement" on page 399.

Restriction Partially supported by Java and ActiveX

Examples "Example 4: Plotting Two Variables" on page 1345

"Example 5: Connecting Plot Data Points" on page 1348

VMINOR=*number-of-minor-ticks*

specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis. Minor tick marks are not labeled. The VMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias VM=

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

VREF=*value* | (*value*) | (*value-list*)

draws one or more reference lines perpendicular to the vertical axis at specified points. For a description of *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. If the AREAS= option is also used, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option. LVREF=, CVREF=, and WVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

Example "Example 5: Connecting Plot Data Points" on page 1348

VREVERSE
specifies that the order of the values on the vertical axis be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values. When all vertical variable values are positive, the first tick mark on the vertical axis has a value of zero. When all vertical variable values are negative, the last tick mark on the vertical axis has a value of zero.

The VZERO request is ignored if the vertical variable contains a mix of positive and negative values. It is ignored when the vertical variable is ordered with the VAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the vertical variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the VZERO option is ignored, the last tick mark on the vertical axis has a value of zero.

WAUTOHREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the horizontal axis as determined by the AUTOHREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option "CAUTOHREF=*reference-line-color*" on page 1302.

Style reference LineThickness attribute of the GraphGridLines element

WAUTOVREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option "CAUTOVREF=*reference-line-color*" on page 1302.

Style reference LineThickness attribute of the GraphGridLines element

WHREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines as determined by the horizontal axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOHREF and HREF= options. Note that the WAUTOHREF= option overrides WHREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with theH REF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the HREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the option "CHREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)" on page 1303.

Style reference LineThickness attribute of the GraphGridLines element

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)

specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOVREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the VREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the option "CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)" on page 1305.

Style reference LineThickness attribute of the GraphGridLines element

Details

Description

The PLOT statement specifies one or more plot requests that name the horizontal and left vertical axis variables, and can specify a third classification variable. This statement automatically does the following:

- scales the axes to include the maximum and minimum data values
- plots data points within the axes
- labels each axis with the name of its variable and displays each major tick mark value.

You can use statement options to manipulate the axes, modify the appearance of your graph, and describe catalog entries. You can use SYMBOL definitions to modify plot symbols for the data points, join data points, draw regression lines, plot confidence limits, or specify other types of interpolations. For more information about the SYMBOL statement, see "About SYMBOL Definitions" on page 1317.

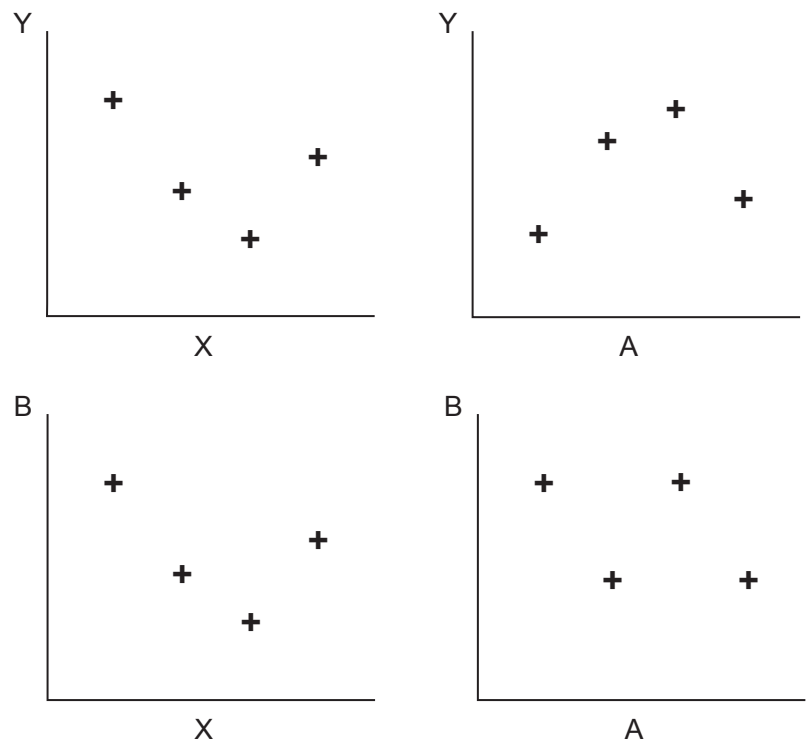
In addition, you can use global statements to modify the axes; add titles, footnotes, and notes to the plot; or modify the legend if one is generated by the plot. You can also use an Annotate data set to enhance the plot.

Plot Requests with Multiple Variables

Plot requests with multiple variables produce a separate plot for every Y*X pair, unless you specify OVERLAY. For example, this statement produces four plots (the actual plots are produced on separate pages). See Figure 42.161 on page 1317.

```
plot (y b)*(x a);
```

Figure 42.7 Graphs Generated by Multiple Plot Requests



About SYMBOL Definitions

SYMBOL statements control the appearance of plot symbols and lines, and define interpolation methods. They can specify the following:

- the shape, size, and color of the plot symbols that mark the data points
- plot line style, color, and width
- an interpolation method for plotting data
- how missing values are treated in interpolation calculations

SYMBOL definitions are assigned either by default by the GPLOT procedure or explicitly with a plot request.

If no SYMBOL definition is currently in effect, the GPLOT procedure produces a scatter plot of the data points using the default plot symbol. If you need more than one SYMBOL definition, the procedure rotates through the colors defined by the current style. If the NOGSTYLE option is specified, the procedure rotates through the device color list. If the current color list contains only one color, or if all the colors are used, additional plot symbols are used.

If SYMBOL definitions have been defined but not explicitly assigned by a plot request of the form *y-variable*x-variable=n*, the procedure assigns them in the order in which they are generated. For example, this statement creates three plots:

```
plot y*x b*a s*r;
```

The procedure assigns the first generated SYMBOL definition to Y*X, the second generated SYMBOL definition to B*A, and the third to S*R.

If more SYMBOL definitions are needed than have been defined, the procedure uses the default definitions for the plots that remain.

See the "SYMBOL Statement" on page 470.

About Plot Requests That Assign a SYMBOL Definition

Plot requests of the form *y-variable*x-variable=n* are useful when you use the OVERLAY option to produce multiple plots on one graph and you want to assign a particular SYMBOL definition to each plot.

With plot requests of this type it is important to remember that a single SYMBOL statement can generate multiple SYMBOL definitions, so that the SYMBOL definition that is designated by *n* might not be the same as the SYMBOL statement of the same number. That is, the third SYMBOL definition is not necessarily the same as the SYMBOL3 statement. For more information see the "SYMBOL Statement" on page 470.

PLOT2 Statement

Produces one or more plots with the vertical axis on the right side of the graph against which a second variable can be plotted.

Requirement: You cannot use the PLOT2 statement alone. It can be used only with a PLOT or BUBBLE statement. At least one plot request is required.

Supported Statements: AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE

Syntax

PLOT2 *plot-request(s)* </ *options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate all graphs that are produced by the GPLOT procedure.

CAXIS=*axis-color*
: specifies the color for the axis line and all major and minor tick marks.

CBASELINE=*baseline-color*
: specifies the color of the horizontal baseline for reference lines of a plot.

CFRAME=*background-color*
: fills the axis area with the specified color.

COUTLINE=*outline-color*

specifies the color of the outline that is drawn around filled areas.

CTEXT=*text-color*
specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions.

FRAME
specifies that a line is drawn around the axis area.

FRONTREF
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

NOFRAME
specifies that a line is not drawn around the axis area.

ODS options

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Plot options

AREAS=*n*
fills all the areas below plot line *n* with a pattern.

GRID
draws reference lines at all major tick marks on both axes.

LEGEND

LEGEND=LEGEND*1–99*
generates a legend or specifies the defined legend to use for the plot.

NOLASTAREA
removes the last area of a plot when the value specified by the AREAS= option exceeds the number of bounded areas in the plot.

NOLEGEND
suppresses the legend that is generated by a plot request of the type *y-variable*x-variable=third-variable*.

OVERLAY
places all the plots that are generated by the PLOT statement on one set of axes.

REGEQN
displays the regression equation that is specified in the INTERPOL= option of the SYMBOL statement in the lower left corner of the plot.

SKIPMISS
breaks a plot line or an area fill at occurrences of missing values of the Y variable.

Vertical axis options

AUTOVREF
draws reference lines at all of the major tick marks on the vertical axis.

CAUTOVREF=*reference-line-color*
: specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option.

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
: specifies the color of reference lines drawn perpendicular to the vertical axis.

LAUTOVREF=*reference-line-type*
: specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option.

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)
: specifies the type of reference lines drawn with the AUTOVREF, VREF, and GRID options.

VAXIS=*value-list* | AXIS<1 – 99>
: specifies the major tick mark values for the vertical axis or assigns an axis definition.

VMINOR=*number-of-minor-ticks*
: specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis.

VREF=*value* | (*value*) | (*value-list*)
: draws one or more reference lines perpendicular to the vertical axis at specified points.

VREVERSE
: specifies that the order of the values on the vertical axis be reversed.

VZERO
: specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

WAUTOVREF=*reference-line-width*
: specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option.

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
: specifies line widths for reference lines as determined by the vertical axis.

Required Argument

plot-request(s)
: each specifies the variables to plot and produces a separate graph, unless you specify the OVERLAY option. All variables must be in the input data set. Multiple plot requests are separated with blanks. A plot request can be any of these:

y-variable*x-variable<=n>
: plots the values of two variables and can assign a SYMBOL definition to the plot.

y-variable
: variable plotted on the right vertical axis.

x-variable
: variable plotted on the horizontal axis.

n
: number of the *n*th generated SYMBOL definition.

(*y-variable(s)*)*(*x-variable(s)*)
plots the values of two or more variable and produces a separate graph for each combination of Y and X variables.

y-variable(s)
variables plotted on the right vertical axes.

x-variable(s)
variables plotted on the horizontal axes.

y-variable*x-variable=third-variable
plots the values of two variables against a third classification variable

y-variable
variable plotted on the right vertical axis.

x-variable
variable plotted on the horizontal axis.

third-variable
classification variable against which *y-variable* and *x-variable* are plotted. *Third-variable* can be character or numeric, but numeric variables should contain discrete rather than continuous values, or should be formatted to provide discrete values.

For more information about plot requests, see "PLOT Statement" on page 1296.

In a PLOT2 plot request, the X variable for the horizontal axis must be the same as in the accompanying PLOT or BUBBLE statement. Typically, the Y variable for the right vertical axis is different.

Use the same types of plot requests with a PLOT2 statement that you use with a PLOT statement, but a PLOT2 statement always plots the values of *y-variable* on the right vertical axis.

Optional Arguments

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

Alias ANNO=

See Chapter 28, "Using Annotate Data Sets," on page 721

AREAS=*n*
fills all the areas below plot line *n* with a pattern. The value of *n* specifies which areas to fill:

- AREAS=1 fills the first area.
- AREAS=2 fills both the first and second areas, and so on.

You might specify a value for the AREAS= option that is greater than the number of bounded areas in the plot. In this case, the area between the top plot line and the axis frame is filled.

Before an area can be filled, the data points that border the area must be joined by a line. Use a SYMBOL statement with one of these interpolation methods to join the data points:

- INTERPOL=JOIN
- INTERPOL=STEP
- INTERPOL=R*series*
- INTERPOL=SPLINE | SM | L

See the "SYMBOL Statement" on page 470 for details about interpolation methods.

By default, the AREAS= option fills areas by rotating a solid fill through the list of colors defined in the current style. If the NOGSTYLE option is specified, the areas are filled by rotating a solid fill through the device's color list. If the graph needs more patterns, it rotates hatch patterns, beginning with the M2N0 pattern. See the "PATTERN Statement" on page 455 for more information about map and plot patterns. However, if color is limited to a single color with the CPATTERN= or COLORS= graphic options, the solid pattern is skipped and the first default pattern is M2N0. If the COLORS= graphic option specifies a single color, use as many SYMBOL statements as you have areas to fill. You must do this because the INTERPOL= setting does not automatically apply to multiple symbol definitions.

You can alter the default pattern behavior by specifying patterns and colors in PATTERN statements that specify map and plot patterns. A separate PATTERN definition is needed for each specified area.

If you specify the PATTERN statements, the AREAS= option uses the lowest numbered PATTERN statement first. If it runs out of patterns, it uses the default behavior for map and plot patterns. See the "PATTERN Statement" on page 455 for details.

Pattern definitions are assigned to the areas below the plot lines in the order the plots are drawn. The first area is that between the horizontal axis and the plot line that is drawn first. The second area is that above the first plot line and below the plot line that is drawn second, and so on. If the line that is drawn second lies below the line that is drawn first, the second area is hidden when the first is filled. The plots with the lower line values must be drawn first to prevent one area fill from overlaying another. If the lines cross, only the part of an area that is above the previous line is visible.

Therefore, you must order your PLOT statements when creating multiple plots in combination with the OVERLAY option. Order them so that the plot request that produces the lowest line value is first (leftmost). The plot request that produces the next lowest line value is the second plot request, and so on.

If you produce multiple plots with a *y-variable*x-variable=third-variable* plot request, the lines are plotted in order of increasing third variable values. In that case, you must recode your data so that the lowest value of the third variable produces the lowest plot line. Likewise, the next lowest value produces the next lowest plot line, and so on.

If you use the VALUE= option in the SYMBOL statement, some symbols might be hidden. If you also specify reference lines with the AREAS= option, they are drawn behind the pattern fill.

Restriction Partially supported by Java

Notes The AREAS= option works only if all plot lines are generated by the same PLOT or PLOT2 statement.

If you have specified the NOGSTYLE option and the first color in your device's default color list is black, color rotation begins with the second color in the list. That is, there are no solid black patterns. See "How Default Patterns and Outlines Are Generated" on page 465 for more information.

Example "Example 7: Filling Areas in an Overlay Plot" on page 1355

AUTOVREF
draws reference lines at all of the major tick marks on the vertical axis. If you also use the AREAS= option, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option in either the PROC GPLOT statement or the PLOT statement. LAUTOVREF=, CAUTOVREF=, and WAUTOVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

CAUTOVREF=*reference-line-color*
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option. If you do not specify the CAUTOVREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

CAXIS=*axis-color*
specifies the color for the axis line and all major and minor tick marks. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Alias CA=

CBASELINE=*baseline-color*
specifies the color of the horizontal baseline for reference lines of a plot. This option compensates for a reference line obscuring the axis. This can occur when using a combination of the AUTOVREF option, the CAUTOVREF= option, or the CVREF= option. They must be specifying any color other than black, and a vertical axis with no offset. For example, the light gray reference line drawn by the AUTOVREF option and a CVREF= graydd specification obscures the default black color of the axis line. This option enables you to set the color of the baseline to match the axis color.

Specify the *baseline-color* as a name or a quoted string. Two examples are presented. The first indicates a name. The second specifies an RGB color code:

```
cbaseline=black
```

```
cb='CX0BB5FF'
```

Alias CB=

Default The value of the CAXIS= option is the default color if you do not specify an alternate color for the baseline. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Restriction This option is not supported by the ACTIVEX, ACTXIMG, and JAVAIMG devices.

Note The baseline is drawn after the plot elements are drawn, and either a default or specified color is applied. This means that the reference

lines are drawn behind plot elements such as bars. Use the FRONTREF option if you want reference lines drawn in front of the bars.

See "Specifying Colors in SAS/GRAPH Programs" on page 364

"FRONTREF" on page 1326

CFRAME=*background-color*
fills the axis area with the specified color.

Style reference Color attribute of the GraphWalls element

Restriction CFRAME= is overridden by the NOFRAME option and the PLOT statement's IFRAME= option. If the FRAME= option is in effect, an image appears in the axis area instead of the color.

Interaction If the FRAME option is also in effect, the procedure behavior is modified. The procedure determines the color of the frame around the axis area according to the precedence list given within the FRAME option description.

COUTLINE=*outline-color*
specifies the color of the outline that is drawn around filled areas. The filled areas are generated when the SYMBOL statement specifies " INTERPOL=*map/plot-pattern* " on page 480 or the GOPTIONS statement specifies "INTERPOL Graphics Option" on page 655. The default outline color is specified in the current style. However, the NOGSTYLE option might be specified. In that case, the default color is the first color in the device's color list (the foreground color). Also, the default slice outline color is determined as follows:

- If you do not specify a PATTERN statement, the default outline color is the color defined in the current style.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

Alias CO=

Style reference Color attribute of the GraphOutlines element

Restriction Not supported by Java

Interaction The COUTLINE= option overrides the SYMBOL statement option CO=.

CTEXT=*text-color*
specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions. The GPLOT procedure searches for a color specification in this order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements. They override the CTEXT= option specified in the PLOT statement. This also includes colors specified by the POINTLABEL= option in the SYMBOL statement.
2. the color specified by the CTEXT= option in the PLOT statement

3 the color specified by the CTEXT= option in the GOPTIONS statement

4 the color specified in the current style, or a NOGSTYLE option specification. With NOGSTYLE specified, the default color is black for the Java and ActiveX devices and the first color in the color list for all other devices

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. The COLOR= color is still used for all other axis colors, such as tick marks.

Alias C=

Style reference Color attributes of the GraphValueText and the GraphLabelText elements

Note If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)
specifies the color of reference lines drawn perpendicular to the vertical axis. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the VREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the VREF= option. The syntax of the color list requires parentheses and colors separated by spaces (*color1 color2 ... colorN*). If you do not specify the CVREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias CV=

Style reference Color attribute of the GraphGridLines element

FRAME
specifies that a line is drawn around the axis area. For the frame color, a specification is searched for in this order:

1 the CAXIS= option

2 the COLOR= option in the AXIS definition assigned to the vertical axis

3 the COLOR= option in the AXIS definition assigned to the horizontal axis

4 the default, which is the color defined by the current style

Alias FR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1324 option.

To fill the axis area with an image, use the PLOT statement's "IFRAME=fileref | 'external-file'" on page 1324 option.

See To override the default and specify that no line is drawn around the axis area, use the "NOFRAME" on page 1328option.

FRONTREF
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars. By default, reference lines are drawn on the back plane of the axis.

GRID
draws reference lines at all major tick marks on both axes. The line color is the color of the axis. When specified in a PLOT2 statement, the reference lines are drawn on the vertical axis on the right side of the plot.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

For Java and ActiveX, the PLOT statement is partially supported by the ACTIVEX device. It is fully supported by the JAVAIMG and ACTXIMG devices. The PLOT2 statement is supported only by the ACTXIMG device.

Restriction Partially supported by Java and ActiveX

See "Data Tips for Web Presentations" on page 253

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java and ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

Example "Example 10: Creating Plots with Drill-down Functionality for the Web" on page 1367

LAUTOVREF=*reference-line-type*

specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

Style reference LineStyle attribute of the GraphGridLines element

LEGEND | LEGEND=LEGEND*1–99*

generates a legend or specifies the defined legend to use for the plot. Use LEGEND to produce a default legend.

Use the LEGEND option with an assigned number to first define the legend characteristics, and then assign the defined LEGEND*1–99* statement to the plot. Use any number in the range of 1–99, for example, LEGEND1.

Interaction A PLOT statement that includes the OVERLAY option does not automatically generate a legend. In these plot types, use LEGEND to produce a default legend, or LEGEND=LEGEND*1–99* to assign a defined LEGEND statement to the plot. The default legend is centered below the axis frame and identifies which colors and plot symbols represent the *y-variables* that you specify for the plots. To control the order of the legend entries for overlaid plots, use the ORDER= option in the LEGEND statement. Specify the list of variables in quotation marks in the preferred order. For example, the following causes the legend entry for y3 to be displayed first, y1 next, and y2 last:

```
legend1 order=('y3' 'y1' 'y2');
proc gplot data=mydata2;
plot (y1 y2 y3)*x / overlay  legend=legend1;
run;
```

Note If you use the SHAPE= option in a LEGEND statement, the value SYMBOL is valid. If you use the PLOT statement's AREAS= option, SHAPE=BAR is also valid.

Tip A plot request of the form *y-variable*x-variable=third-variable* automatically generates a default legend that identifies which colors and plot symbols represent each value of the classification variable. In these plot types, override the default by using LEGEND=LEGEND*1–99* to assign a defined LEGEND statement to the plot.

See " ORDER=(*value-list*) | DESCENDING" on page 438 for more information.

"LEGEND Statement" on page 432

Example "Example 6: Generating an Overlay Plot" on page 1351

LVREF=*reference-line-type* | (*reference-line-type*) | (*reference-line-type-list*)

specifies the type of reference lines drawn with the AUTOVREF, VREF, and GRID options. The *reference-line-type* value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single line type applies that line type

to all reference lines. The LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single line type applies that line type only to the first line drawn with the VREF= option. Specifying a line-type list applies line types in sequence to successive reference lines drawn with the VREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (*type1 type2 ... typeN*). The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these references lines, use the CVREF= option. To specify labels for these reference lines, use the VAXIS= option.

You might generate a needle plot with a Java or ActiveX device driver. In this case, the value of the LVREF= option is not applied to the default reference line that is drawn at zero. This occurs when the minimum value of the vertical axis is less than zero. This line is solid (not dashed).

Alias LV=

Style reference LineStyle attribute of the GraphGridLines element

Restriction Partially supported by Java and ActiveX

Example "Example 5: Connecting Plot Data Points" on page 1348

NOAXIS

suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias NOAXES

NOFRAME

specifies that a line is not drawn around the axis area.

To override the default and specify that no line is to be drawn, use the NOFRAME option.

Alias NOFR

Default FRAME

Interaction If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

Tips To fill the axis area with color, use the "CFRAME=*background-color*" on page 1324 option.

To fill the axis area with an image, use the PLOT statement's "IFRAME=*fileref* | '*external-file*'" on page 1309 option.

See To specify that a line is drawn around the axis area, use the "FRAME" on page 1325option.

NOLASTAREA

removes the last area of a plot when the value specified by the AREAS= option exceeds the number of bounded areas in the plot. The last area is the area between the last plot line and the axis frame. You can use the NOLASTAREA option in PLOT and PLOT2 statements. However, when both PLOT and PLOT2 statements exist, the NOLASTAREA option is ignored in the PLOT statement. The NOLASTAREA option is also ignored in both PLOT and PLOT2 statements

when the AREAS= option is not specified. It is also ignored when the number of bounded areas in the plot is less than or equal to the value specified by the AREAS= option.

Alias NOLASTFILL

Restriction Not supported by Java and ActiveX.

Note Certain conditions related to the PLOT2 statement might result in variations in the behavior of the NOLASTAREA option.

See "AREAS=*n*" on page 1321

NOLEGEND

suppresses the legend that is generated by a plot request of the type *y-variable*x-variable=third-variable*.

OVERLAY

places all the plots that are generated by the PLOT statement on one set of axes. The axes are scaled to include the minimum and maximum values of all of the variables. They also include the variable names or labels associated with the first pair of variables label the axes.

The OVERLAY option produces a legend if you include the LEGEND option or the LEGEND=*n* option in the PLOT statement.

OVERLAY is not enabled with plot requests of the form *y-variable*x-variable=third-variable*. However, you can achieve an overlay effect by using a PLOT and PLOT2 statement.

When generating output for the web with the JAVAIMG device driver, the following condition applies. The OVERLAY option cannot be used in the PLOT or PLOT2 statement if the PLOT or PLOT2 statement is combined with the global SYMBOL statement when the SYMBOL statement uses the INTERPOL= BOX, HILO, or STD.

Restriction Partially supported by Java

Examples "Example 6: Generating an Overlay Plot" on page 1351

"Example 7: Filling Areas in an Overlay Plot" on page 1355

REGEQN

displays the regression equation that is specified in the INTERPOL= option of the SYMBOL statement in the lower left corner of the plot. You cannot modify the format that is used for the equation.

The GPLOT regression equation is computed from the screen coordinates of the markers. Therefore, a graph might not be displayed if the chart area for the plot becomes so small that markers cannot be drawn. This is because there are no coordinates from which to build the regression equation. In such cases, the regression equation is no longer meaningful.

Restriction Not supported by ActiveX

Example "Example 4: Plotting Two Variables" on page 1345

SKIPMISS

breaks a plot line or an area fill at occurrences of missing values of the Y variable. By default, plot lines and area fills are not broken at missing values. The SKIPMISS option is available only with JOIN interpolation. If the SKIPMISS

option is used, observations should be sorted by the independent (horizontal axis) variable. If the plot request is *y-variable*x-variable=third-variable*, observations should also be sorted by the values of the third variable.

See "About the Input Data Set" on page 1264

URL=*character-variable*
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only

Not supported by the JAVAIMG and ACTXIMG devices when using the PLOT2 statement

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

Note You can specify this option starting with SAS 9.4M1. The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

VAXIS=*value-list* | AXIS<1 – 99>
specifies the major tick mark values for the vertical axis or assigns an axis definition. For a description of *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on page 1306 for the PLOT statement. To assign labels to reference lines, use an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOVREF and VREF= options.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the "AXIS Statement" on page 399.

Restriction Partially supported by Java and ActiveX

Examples "Example 4: Plotting Two Variables" on page 1345

"Example 5: Connecting Plot Data Points" on page 1348

VMINOR=*number-of-minor-ticks*
specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis. Minor tick marks are not labeled. The VMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias VM=

Example "Example 2: Labeling and Sizing Plot Bubbles" on page 1339

VREF=*value* | (*value*) | (*value-list*)
draws one or more reference lines perpendicular to the vertical axis at specified points. For a description of *value-list* see "HAXIS=*value-list* | AXIS<1 – 99>" on

page 1306 for the PLOT statement. If the AREAS= option is also used, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option. LVREF=, CVREF=, and WVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

Example "Example 5: Connecting Plot Data Points" on page 1348

VREVERSE
specifies that the order of the values on the vertical axis be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values. When all vertical variable values are positive, the first tick mark on the vertical axis has a value of zero. When all vertical variable values are negative, the last tick mark on the vertical axis has a value of zero.

The VZERO request is ignored if the vertical variable contains a mix of positive and negative values. It is ignored when the vertical variable is ordered with the VAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the vertical variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the VZERO option is ignored, the last tick mark on the vertical axis has a value of zero.

WAUTOVREF=*reference-line-width*
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option "CAUTOVREF=*reference-line-color*" on page 1323.

Style reference LineThickness attribute of the GraphGridLines element

WVREF=*reference-line-width* | (*reference-line-width*) | (*reference-line-width-list*)
specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOVREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=*reference-line-width* for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the VREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (*width1 width2 ...widthN*). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the option "CVREF=*reference-line-color* | (*reference-line-color*) | (*reference-line-color-list*)" on page 1325.

Style reference LineThickness attribute of the GraphGridLines element

Details

Options

Options for the PLOT2 statement are identical to the options for the PLOT statement except for these options, which are ignored if you specify them:

- AUTOHREF
- CAUTOHREF=
- CHREF=
- DESCRIPTION=
- HAXIS=
- HMINOR=
- HREF=
- HREVERSE=
- HZERO=
- IFRAME=
- IMAGESTYLE =
- LAUTOHREF=
- LHREF=
- NAME=
- WHREF=
- WAUTOHREF=

See "PLOT Statement" on page 1296 for descriptions of options that you can use with the PLOT2 statement.

Description

The PLOT2 statement specifies one or more plot requests that name the horizontal and right vertical axis variables. This statement automatically does the following:

- plots data points within the axes
- scales the axes to include the maximum and minimum data values
- labels each axis with the name of its variable and displays each major tick mark value

You can use statement options to manipulate the axes and modify the appearance of your graph. You can use SYMBOL definitions to modify plot symbols for the data points, join data points, draw regression lines, plot confidence limits, or specify other types of interpolation. For more information about the SYMBOL statement see "About SYMBOL Definitions" on page 1317.

Note: You might be using the PLOT2 statement to generate output with the Java or ACTIVEX device drivers. If you are also using the global statement SYMBOL, the value of the SYMBOL statement option INTERPOL= cannot be BOX, STD, or HILO.

In addition, you can use global statements to modify the axes; to add titles, footnotes, and notes to the plot; or to modify the legend if one is generated by the plot. You can also use an Annotate data set to enhance the plot.

Matching Plot Requests

Matching Requests between PLOT and PLOT2 Statements

The plot requests in both the PLOT and PLOT2 statements must be evenly matched as in this example:

```
plot  y*x  b*a;
    plot2 y2*x b2*a;
```

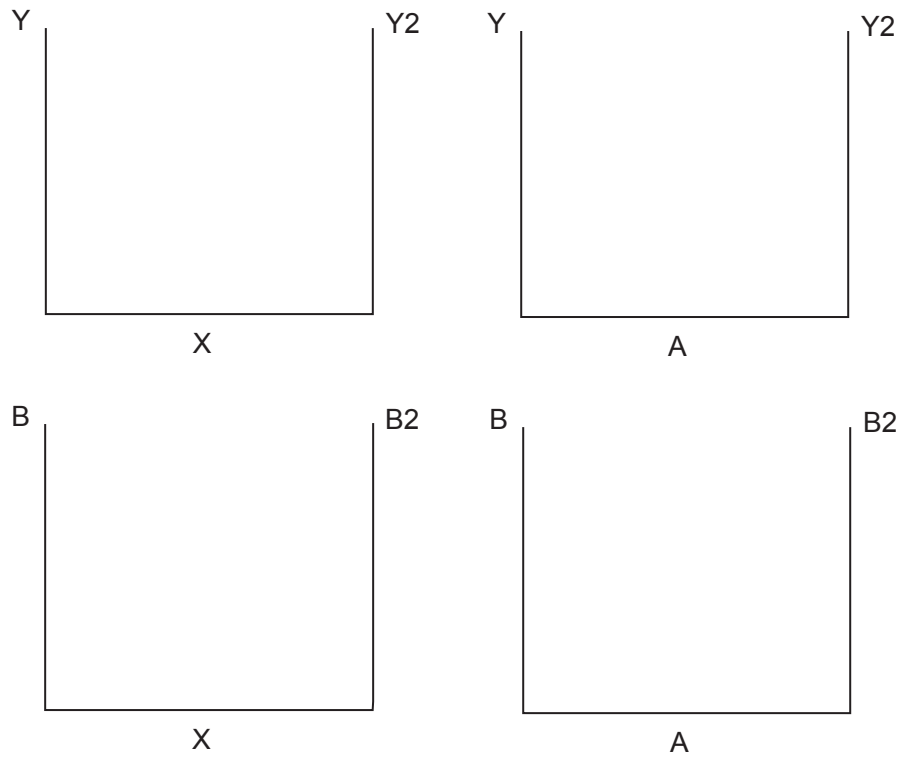
These statements produce two graphs, and each one has two vertical axes. The first pair of plot requests (Y*X and Y2*X) produce one graph. In this graph X is plotted on the horizontal axis, Y is plotted on the left axis, and Y2 is plotted on the right axis. The second pair of plot requests (B*A and B2*A) produce another graph. In that graph A is plotted on the horizontal axis, B is plotted on the left axis, and B2 is plotted on the right axis.

Using Multiple Plot Requests

Plot requests of the form (*y-variable(s)*)*(*x-variable(s)*). Both the PLOT and PLOT2 statements generate multiple graphs (the actual plots are produced on separate pages). See Figure 42.162 on page 1334.

```
plot (y b)*(x a);
    plot2 (y2 b2)*(x a);
```

Figure 42.8 Graphs Produced by Multiple Plot Requests in PLOT and PLOT2 Statements

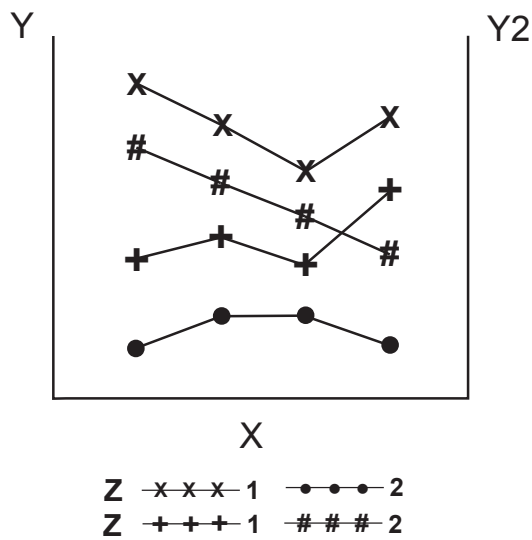


Requesting Plots of Three Variables with a Legend

When both the PLOT and PLOT2 statements use plot requests of the form *y-variable*x-variable=third-variable*, each statement generates a separate legend. If the third variable has two values, these statements produce one graph with four sets of data points. See Figure 42.163 on page 1335. The figure assumes that SYMBOL statements are used to specify the plot symbols that are shown and to connect the data points with straight lines.

```
plot y*x=z;
    plot2 y2*x=z;
```

Figure 42.9 *Multiple Plots on One Graph*



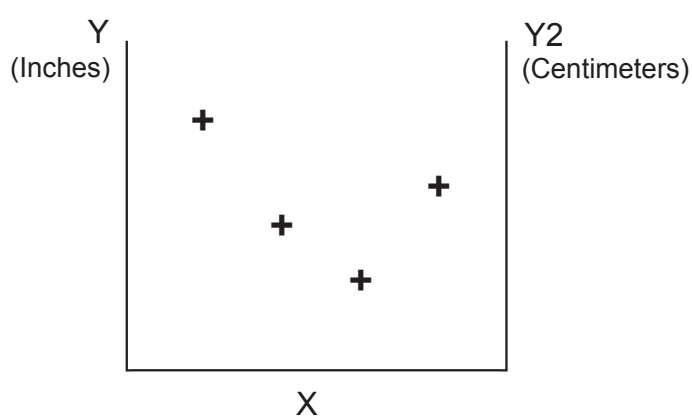
Using a Second Vertical Axis

Displaying the Same Values in a Different Scale

If your data contain the same variable values in two different scales, such as height in inches and height in centimeters, you can display one scale of values on the left axis and the other scale of values on the right axis. Both vertical axes might be calibrated so that they represent the same range of values. In this case, for each observation of X, the data points for Y and Y2 are the same.

For example, Y is height in inches and Y2 is height in centimeters. If the Y axis values range from 0 to 84 inches and the Y2 axis values range from 0 to 213.36 centimeters, the plot is displayed like Figure 42.164 on page 1335.

Figure 42.10 *Right Axis with Different Scale of Values*

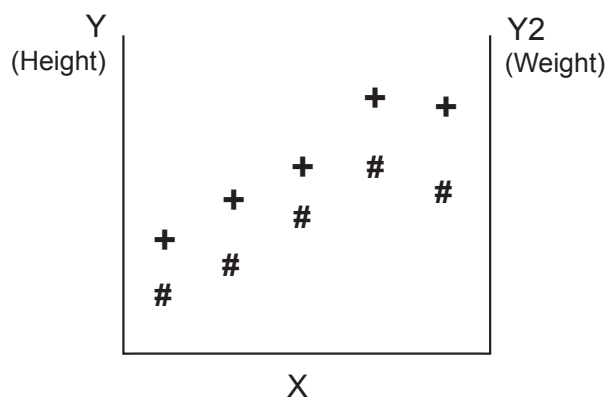


For these types of plots, the PLOT2 statement should use a SYMBOL statement that specifies INTERPOL=NONE and VALUE=NONE.

Displaying Different Values

Your data might contain variables with different data values (such as height and weight). In this case you can display one type of data on the left axis and another type of data on the right axis. Because the Y variable and the Y2 variable contain different data, two sets of data points are displayed on the graph. For example, if Y is height and Y2 is weight, the plot is like Figure 42.165 on page 1336.

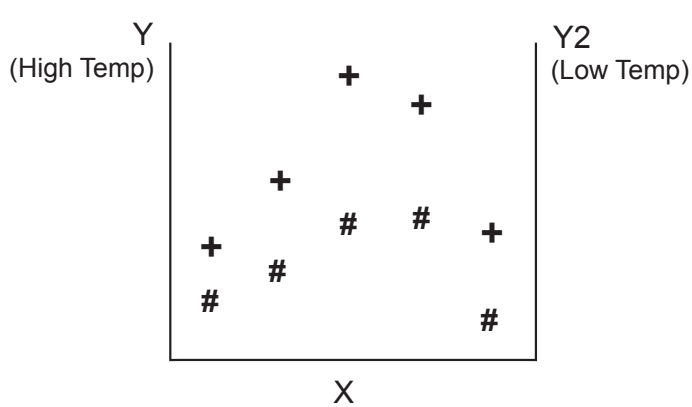
Figure 42.11 *Right Axis with Different Values and Different Scale*



Displaying the Same Scale on Both Axes

Your data might contain two sets of values for the same type of data. In this case you can use the PLOT2 statement to generate a right axis that is calibrated the same as the left axis. This makes the data points on the right of the graph easier to read. For example, if Y is high temperatures and Y2 is low temperatures, you can create a graph like Figure 42.166 on page 1336.

Figure 42.12 *Right Axis with Same Scale of Values*



To scale both axes the same, specify the same range of values either with the VAXIS= option in both the PLOT and PLOT2 statements, or with AXIS statements.

Using PATTERN and SYMBOL Definitions

The PLOT2 statement uses PATTERN and SYMBOL definitions in the same way the PLOT statement does. These definitions are assigned in order first to the PLOT statement and then to the PLOT2 statement.

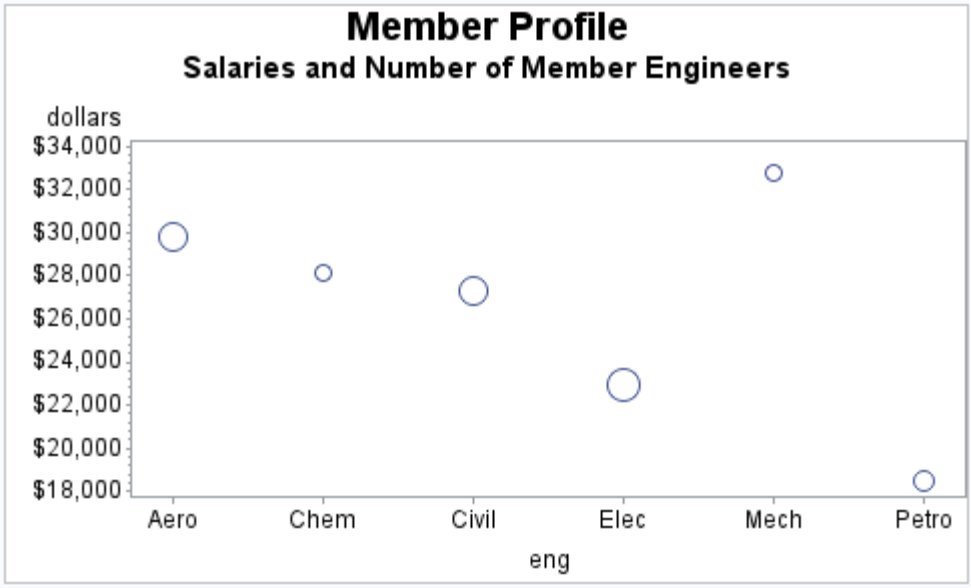
For more information, see “About SYMBOL Definitions” on page 1317.

Examples: GPLOT Procedure

Example 1: Generating a Simple Bubble Plot

Features:	BUBBLE statement option HAXIS= GOPTIONS statement option BORDER AXIS statement FORMAT statement
Sample library member:	GPLBUBL1
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example shows a bubble plot in which each bubble represents a category of engineer. The plot shows engineers on the horizontal axis and average salaries on the vertical axis. Each bubble's vertical location is determined by the average salary for the category. Each bubble's size is determined by the number of engineers in the category: the more engineers, the larger the bubble.



Program

```
goptions reset=all border;

data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;

title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";

axis1 offset=(5,5);

proc gplot data=jobs;
   format dollars dollar9.;
   bubble dollars*eng=num / haxis=axis1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set. The data set JOBS contains average salary data for several categories of engineer. It also indicates the number of engineers in each category.

```
data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;
```

Define titles.

```
title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";
```

Define axis characteristics. The OFFSET= option specifies an offset for the tick marks so that bubbles near an axis are not clipped.

```
axis1 offset=(5,5);
```

Generate bubble plot. The HAXIS= option assigns the AXIS1 statement to the horizontal axis. The salary averages are assigned a dollar format.

```
proc gplot data=jobs;
   format dollars dollar9.;
   bubble dollars*eng=num / haxis=axis1;
run;
quit;
```

Example 2: Labeling and Sizing Plot Bubbles

Features:

BUBBLE statement options
- BCOLOR
- BLABEL
- BSIZE
- HAXIS=
- VAXIS=
- VMINOR

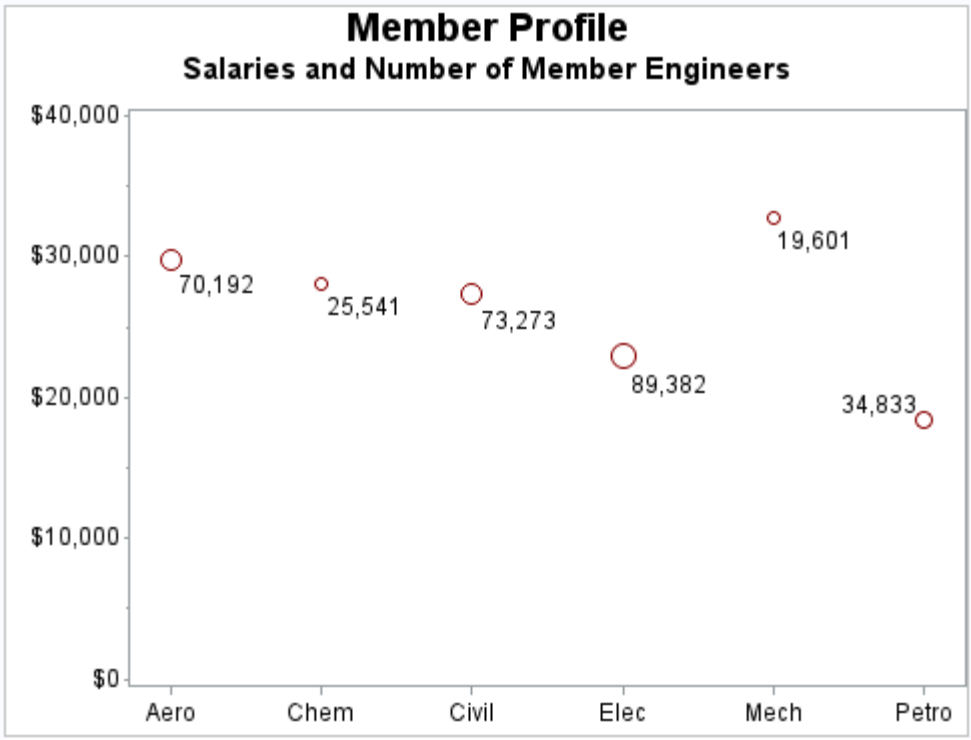
GOPTIONS statement option
- BORDER

AXIS statement

Sample library member: GPLBUBL2

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example modifies the code in Example 1. It shows how BUBBLE statement options control the appearance of bubbles and their labels. It also shows how AXIS statements can modify the plot axes.



Program

```
goptions reset=all border;

data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;

title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";

axis1 label=none
      offset=(5,5);
axis2 order=(0 to 40000 by 10000)
      label=none;

proc gplot data=jobs;
   format dollars dollar9. num comma7.0;
   bubble dollars*eng=num / haxis=axis1
```

```
                              vaxis=axis2
                              vminor=1
                              bcolor=darkred
                              blabel
                              bsize=3;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set. The data set JOBS contains average salary data for several categories of engineer. It also indicates the number of engineers in each category.

```
data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;
```

Define titles.

```
title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";
```

Define axis characteristics. AXIS1 suppresses the horizontal axis label and uses the OFFSET= option to move the first and last major tick mark values away from the vertical axes. This ensures that the bubbles are not clipped. AXIS2 uses the ORDER= option to set major tick mark intervals. This could be done with the VAXIS= option in the BUBBLE statement, but then you could not suppress the axis label and alter other axis characteristics.

```
axis1 label=none
      offset=(5,5);
axis2 order=(0 to 40000 by 10000)
      label=none;
```

Generate bubble plot. The VMINOR= option specifies one minor tick mark for the vertical axis. The BLABEL option labels each bubble with the value of variable NUM. Then BCOLOR= option specifies the color for the bubbles. The BLABEL option labels the bubbles with the value of the third variable, which in this case is the number of engineers in the job category. The BSIZE option specifies the size of the bubbles.

```
proc gplot data=jobs;
   format dollars dollar9. num comma7.0;
   bubble dollars*eng=num / haxis=axis1
                            vaxis=axis2
                            vminor=1
```

```
                                    bcolor=darkred
                                    blabel
                                    bsize=3;
run;
quit;
```

Example 3: Adding a Right Vertical Axis

Features:

BUBBLE statement options
- VAXIS=
- HAXIS=
- HMINOR=
- VMINOR=
- BLABEL

AXIS statement

FORMAT statement

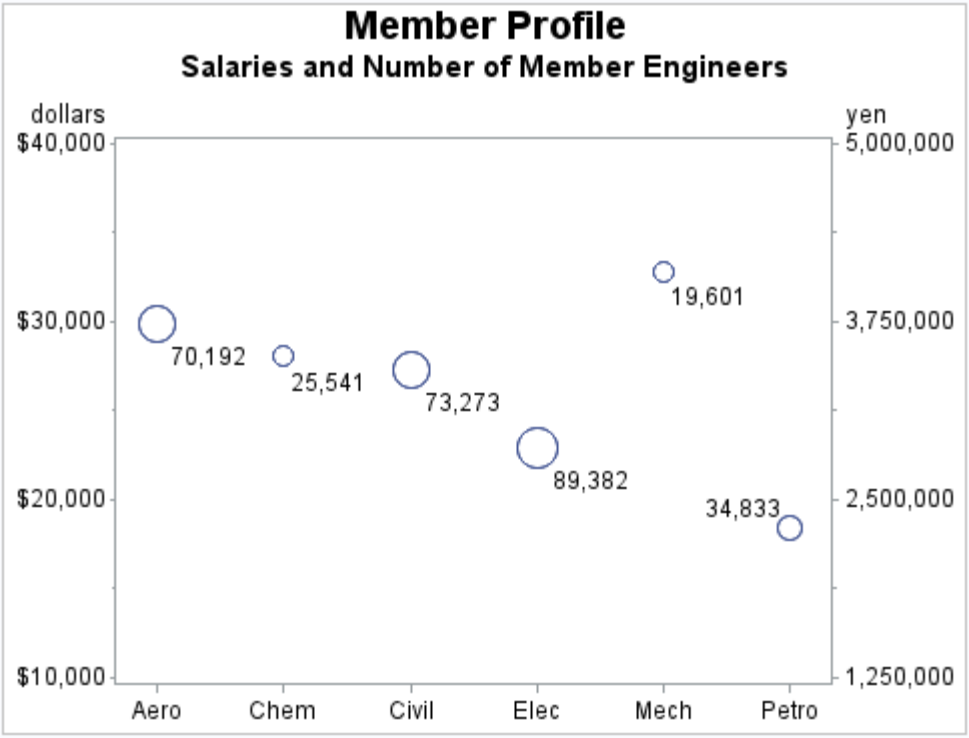
GOPTIONS statement option
- BORDER

Sample library member: GPLAXIS1

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example modifies "Example 2: Labeling and Sizing Plot Bubbles" on page 1339 to show how a BUBBLE2 statement generates a right vertical axis. This axis displays the values of the vertical coordinates in a different scale from the scale that is used for the left vertical axis. Salary values are scaled by dollars on the left vertical axis and by yen on the right vertical axis. BUBBLE and BUBBLE2 statement options control the appearance of the graph. In particular, the VAXIS options calibrate the axes so that the data points are identical and only one set of bubbles appears.

Note: If the data points are not identical, two sets of bubbles are displayed.



Program

```
goptions reset=all border;

data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;

data jobs2;
   set jobs;
   yen=dollars*125;
run;

title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";

axis1 label=none
       offset=(5,5);

proc gplot data=jobs2;
   format dollars dollar7. num yen comma9.0;
   bubble dollars*eng=num / haxis=axis1
                            vaxis=10000 to 40000 by 10000
                            hminor=0
```

```
                                vminor=1
                                blabel;

   bubble2 yen*eng=num / vaxis=1250000 to 5000000 by 1250000
                         vminor=1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the JOBS data set. The data set JOBS contains average salary data for several categories of engineer. It also indicates the number of engineers in each category.

```
data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;
```

Create the data set JOBS2 and calculate variable YEN. The DATA step uses a SET statement to read the JOBS data set.

```
data jobs2;
   set jobs;
   yen=dollars*125;
run;
```

Define titles.

```
title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";
```

Define horizontal-axis characteristics.

```
axis1 label=none
       offset=(5,5);
```

Generate bubble plot with second vertical axis. In the BUBBLE statement, the HAXIS= option specifies the AXIS1 definition and the VAXIS= option scales the left axis. In the BUBBLE2 statement, the VAXIS= option scales the right axis. Both axes represent the same range of monetary values. The BUBBLE and BUBBLE2 statements ensure that the bubbles generated by each statement are identical by coordinating specifications on any options in these statements.

```
proc gplot data=jobs2;
   format dollars dollar7. num yen comma9.0;
   bubble dollars*eng=num / haxis=axis1
                            vaxis=10000 to 40000 by 10000
```

```
                                hminor=0
                                vminor=1
                                blabel;

   bubble2 yen*eng=num / vaxis=1250000 to 5000000 by 1250000
                         vminor=1;
run;
quit;
```

Example 4: Plotting Two Variables

Features:

PLOT statement options
- HAXIS=
- HMINOR=
- REGEQN
- VAXIS=

GOPTIONS statement option
- BORDER

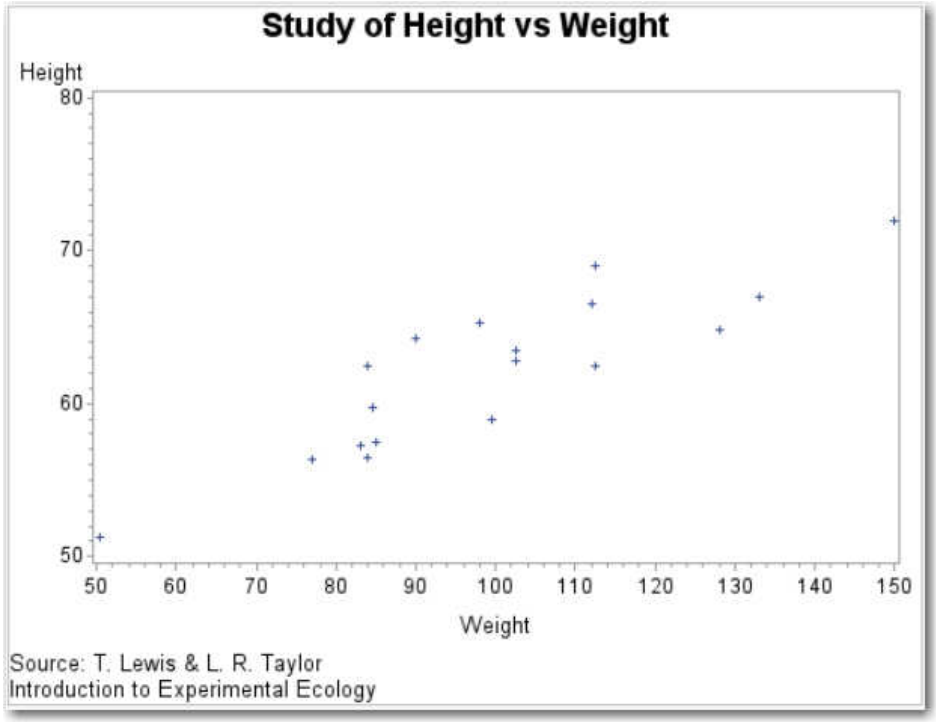
SYMBOL statement

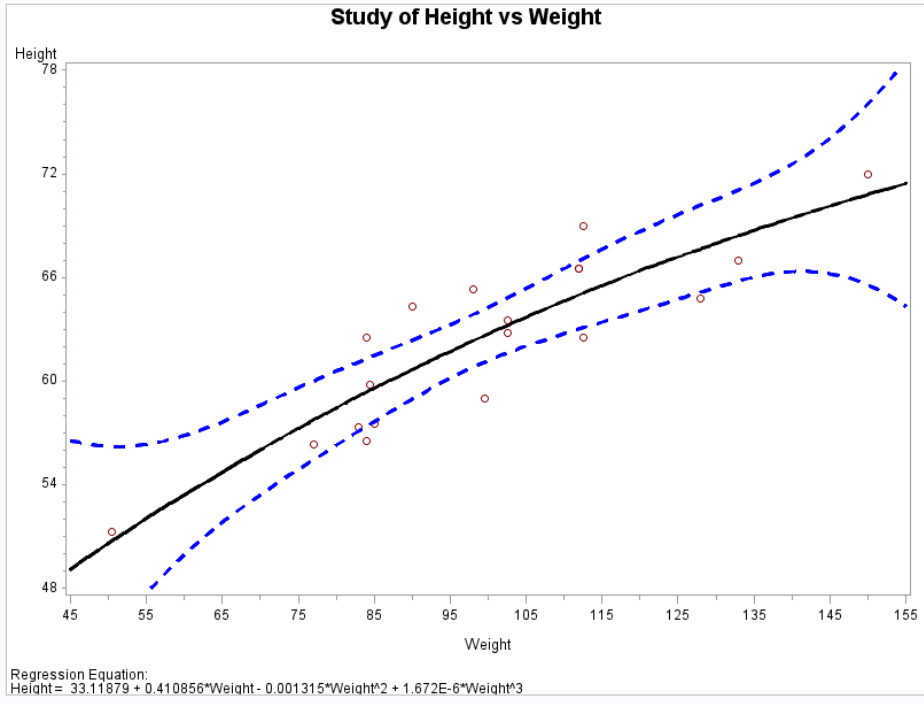
Sample library member: GPLVRBL1

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

In this example, the PLOT statement uses a plot request of the type *y-variable*x-variable* to plot the variable HEIGHT against the variable WEIGHT. The plot shows that weight generally increases with size.

The second example then requests the same plot with some modifications. As shown by the second output display, the second plot request specifies a regression analysis with confidence limits, and scales the range of values along the vertical and horizontal axes. It also displays the regression equation specified for the SYMBOL statement. Because the procedure supports RUN-group processing, you do not have to repeat the PROC GPLOT statement to generate the second plot.





Program

```
goptions reset=all border;

title "Study of Height vs Weight";
footnote1 j=l "Source: T. Lewis & L. R. Taylor";
footnote2 j=l "Introduction to Experimental Ecology";
```

```
proc gplot data=sashelp.class;
   plot height*weight;
run;

footnote1; /* this clears footnote1 and footnote2 */

symbol1 interpol=rcclm95
       value=circle
       cv=darkred
       ci=black
       co=blue
       width=2;

   plot height*weight / haxis=45 to 155 by 10
                        vaxis=48 to 78 by 6
                        hminor=1
                        regeqn;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define title and footnotes.

```
title "Study of Height vs Weight";
footnote1 j=l "Source: T. Lewis & L. R. Taylor";
footnote2 j=l "Introduction to Experimental Ecology";
```

Generate a default scatter plot.

```
proc gplot data=sashelp.class;
   plot height*weight;
run;
```

Clear the footnotes to make room for the regression equation.

```
footnote1; /* this clears footnote1 and footnote2 */
```

Define symbol characteristics. The INTERPOL= option specifies a cubic regression analysis with confidence limits for mean predicted values. The VALUE=and CV= options specify a plot symbol and color. The CI=, CO=, and WIDTH= options specify colors and a thickness for the interpolation and confidence-limits lines.

```
symbol1 interpol=rcclm95
       value=circle
       cv=darkred
       ci=black
       co=blue
       width=2;
```

Generate scatter plot with regression line. The HAXIS= and VAXIS= options define the range of axes values. The HMINOR= option specifies one minor tick mark between major tick marks. The REGEQN option displays the regression equation specified on the SYMBOL1 statement.

```
   plot height*weight / haxis=45 to 155 by 10
```

```
                                vaxis=48 to 78 by 6
                                hminor=1
                                regeqn;
run;
quit;
```

Example 5: Connecting Plot Data Points

Features:

PLOT statement options
- HMINOR=
- LVREF=
- VAXIS=
- VMINOR=
- VREF=

GOPTIONS statement option
- BORDER

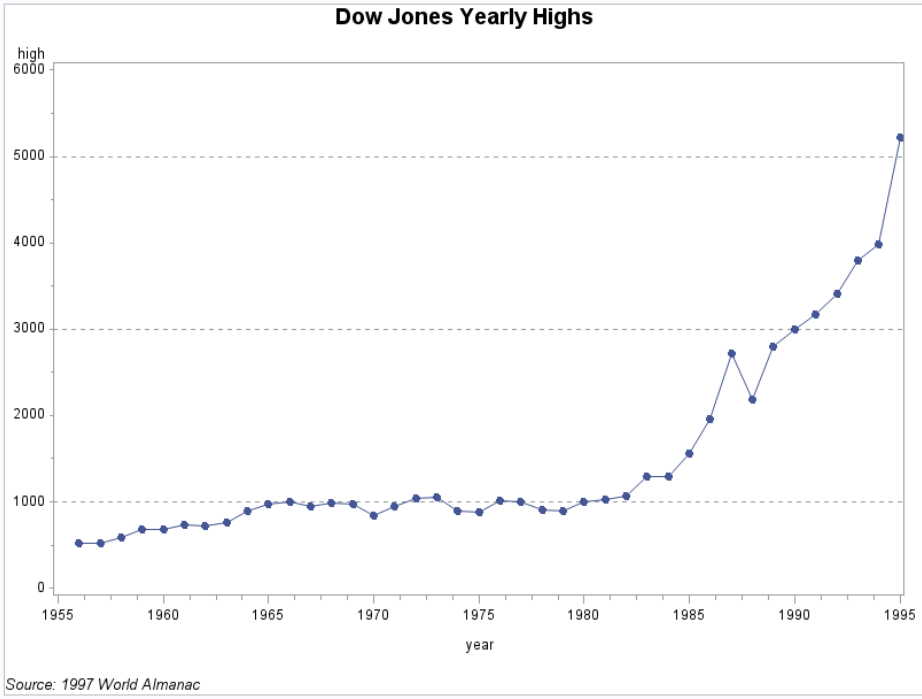
SYMBOL statement

Sample library member: GPLDTPT1

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

In this example, the PLOT statement uses a plot request of the type *y-variable*x-variable* to plot the variable HIGH against the variable YEAR to show the annual highs of the Dow Jones Industrial Average over several decades.

This example uses a SYMBOL statement to specify a plot symbol and connect data points with a straight line. In addition, the example shows how PLOT statement options can add reference lines and modify the axes (AXIS statements are not used).



Program

```
goptions reset=all border;

data stocks;
   input year high low @@;

   datalines;
1956  521.05  462.35 1957  520.77  419.79
1958  583.65  436.89 1959  679.36  574.46
1960  685.47  568.05 1961  734.91  610.25
1962  726.01  535.76 1963  767.21  646.79
1964  891.71  768.08 1965  969.26  840.59
1966  995.15  744.32 1967  943.08  786.41
1968  985.21  825.13 1969  968.85  769.93
1970  842.00  631.16 1971  950.82  797.97
1972 1036.27  889.15 1973 1051.70  788.31
1974  891.66  577.60 1975  881.81  632.04
1976 1014.79  858.71 1977  999.75  800.85
1978  907.74  742.12 1979  897.61  796.67
1980 1000.17  759.13 1981 1024.05  824.01
1982 1070.55  776.92 1983 1287.20 1027.04
1984 1286.64 1086.57 1985 1553.10 1184.96
1986 1955.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
1994 3978.36 3593.35 1995 5216.47 3832.08
;

title1 "Dow Jones Yearly Highs";
footnote1 j=l "Source: 1997 World Almanac";
```

```
symbol1 interpol=join
        value=dot;

proc gplot data=stocks;
   plot high*year / haxis=1955 to 1995 by 5
                    vaxis=0 to 6000 by 1000
                    hminor=3
                    vminor=1

                    vref=1000 3000 5000
                    lvref=2;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set. STOCKS contains yearly highs and lows for the Dow Jones Industrial Average and the dates of the high and low values each year.

```
data stocks;
   input year high low @@;

   datalines;
1956  521.05  462.35 1957  520.77  419.79
1958  583.65  436.89 1959  679.36  574.46
1960  685.47  568.05 1961  734.91  610.25
1962  726.01  535.76 1963  767.21  646.79
1964  891.71  768.08 1965  969.26  840.59
1966  995.15  744.32 1967  943.08  786.41
1968  985.21  825.13 1969  968.85  769.93
1970  842.00  631.16 1971  950.82  797.97
1972 1036.27  889.15 1973 1051.70  788.31
1974  891.66  577.60 1975  881.81  632.04
1976 1014.79  858.71 1977  999.75  800.85
1978  907.74  742.12 1979  897.61  796.67
1980 1000.17  759.13 1981 1024.05  824.01
1982 1070.55  776.92 1983 1287.20 1027.04
1984 1286.64 1086.57 1985 1553.10 1184.96
1986 1955.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
1994 3978.36 3593.35 1995 5216.47 3832.08
;
```

Define title and footnote.

```
title1 "Dow Jones Yearly Highs";
footnote1 j=l "Source: 1997 World Almanac";
```

Define symbol characteristics. Specifying INTERPOL=JOIN joins the data points with straight lines and the VALUE= option specifies the type of symbol used.

```
symbol1 interpol=join
```

```
          value=dot;
```

Generate the plot and modify the axis values. The VAXIS= option sets major tick marks for the vertical axis. The HMINOR= and VMINOR= options specify the number of tick marks between major tick marks.

```
proc gplot data=stocks;
   plot high*year / haxis=1955 to 1995 by 5
                    vaxis=0 to 6000 by 1000
                    hminor=3
                    vminor=1
```

Add reference lines. The VREF= option draws reference lines on the vertical axis at three marks. TheLVREF= option specifies the line style (dashed) for the lines.

```
                    vref=1000 3000 5000
                    lvref=2;
run;
quit;
```

Example 6: Generating an Overlay Plot

Features:

PLOT statement options
- COLOR=
- HAXIS=
- HMINOR=
- LEGEND=
- LVREF=
- OVERLAY
- VAXIS=
- VMINOR=
- VREF=

GOPTIONS statement options
- BORDER
- RESET=

LEGEND statement

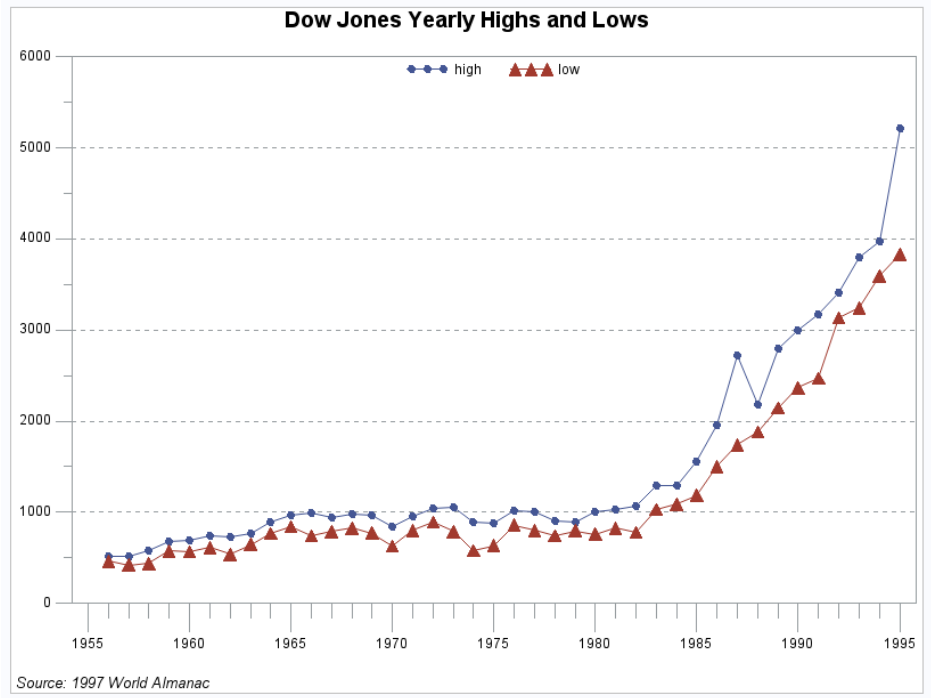
SYMBOL statement

Sample library member: GPLOVRL1

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

In this example, one PLOT statement plots both the HIGH and LOW variables against the variable YEAR using two plot requests. The OVERLAY option in the PLOT statement determines that both plot lines appear on the same graph. The other PLOT options scale the vertical axis, add a reference line to the plot, and specify the number of minor tick marks on the axes. The SYMBOL, AXIS, and LEGEND statements modify the plot symbols, axes, and legend.

Note: If the OVERLAY option is not specified, each plot request generates a separate graph.



Program

```
goptions reset=all border;

data stocks;
   input year high low @@;

   datalines;
1956  521.05  462.35 1957  520.77  419.79
1958  583.65  436.89 1959  679.36  574.46
1960  685.47  568.05 1961  734.91  610.25
1962  726.01  535.76 1963  767.21  646.79
1964  891.71  768.08 1965  969.26  840.59
1966  995.15  744.32 1967  943.08  786.41
1968  985.21  825.13 1969  968.85  769.93
1970  842.00  631.16 1971  950.82  797.97
1972 1036.27 889.15 1973 1051.70  788.31
1974  891.66  577.60 1975  881.81  632.04
1976 1014.79  858.71 1977  999.75  800.85
1978  907.74  742.12 1979   897.61  796.67
1980 1000.17  759.13 1981 1024.05  824.01
1982 1070.55  776.92 1983 1287.20 1027.04
1984 1286.64 1086.57 1985 1553.10 1184.96
1986 1955.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
```

```
1994 3978.36 3593.35 1995 5216.47 3832.08
;

title1 "Dow Jones Yearly Highs and Lows";
footnote1 j=l " Source: 1997 World Almanac"
          ;

symbol1 interpol=join
        value=dot
        color=_style_;
symbol2 interpol=join
        value=C
        font=marker
        color=_style_ ;

axis1 order=(1955 to 1995 by 5) offset=(2,2)
      label=none
      major=(height=2)
     minor=(height=1)
      ;

axis2 order=(0 to 6000 by 1000) offset=(0,0)
      label=none
      major=(height=2)
     minor=(height=1)
      ;

legend1 label=none
        position=(top center inside)
        mode=share;

proc gplot data=stocks;
   plot high*year low*year / overlay legend=legend1
                             vref=1000 to 5000 by 1000
                             lvref=2
                             haxis=axis1 hminor=4
                             vaxis=axis2 vminor=1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set. STOCKS contains yearly highs and lows for the Dow Jones Industrial Average and the dates of the high and low values each year.

```
data stocks;
   input year high low @@;

   datalines;
1956  521.05  462.35 1957  520.77  419.79
1958  583.65  436.89 1959  679.36  574.46
1960  685.47  568.05 1961  734.91  610.25
1962  726.01  535.76 1963  767.21  646.79
1964  891.71  768.08 1965  969.26  840.59
```

```
1966  995.15  744.32 1967  943.08  786.41
1968  985.21  825.13 1969  968.85  769.93
1970  842.00  631.16 1971  950.82  797.97
1972 1036.27 889.15 1973 1051.70  788.31
1974  891.66  577.60 1975  881.81  632.04
1976 1014.79  858.71 1977  999.75  800.85
1978  907.74  742.12 1979   897.61  796.67
1980 1000.17  759.13 1981 1024.05  824.01
1982 1070.55  776.92 1983 1287.20 1027.04
1984 1286.64 1086.57 1985 1553.10 1184.96
1986 1955.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
1994 3978.36 3593.35 1995 5216.47 3832.08
;
```

Define title and footnote.

```
title1 "Dow Jones Yearly Highs and Lows";
footnote1 j=l " Source: 1997 World Almanac"
          ;
```

Define symbol characteristics. Each SYMBOL statement specifies a symbol type for the plot symbols, and connects the data points with a straight line.

```
symbol1 interpol=join
        value=dot
        color=_style_;
symbol2 interpol=join
        value=C
        font=marker
        color=_style_ ;
```

Define axis characteristics.

```
axis1 order=(1955 to 1995 by 5) offset=(2,2)
      label=none
      major=(height=2)
     minor=(height=1)
      ;

axis2 order=(0 to 6000 by 1000) offset=(0,0)
      label=none
      major=(height=2)
     minor=(height=1)
      ;
```

Define legend characteristics. The LABEL= option suppresses the legend label. The POSITION= option centers the legend inside the top of the axis frame. The MODE= option shares the legend area with other graphics elements.

```
legend1 label=none
        position=(top center inside)
        mode=share;
```

Generate two plots and display them on the same set of axes. The OVERLAY option specifies that both plot lines appear on the same graph. The LEGEND= option assigns the LEGEND1 definition to the graph. The VAXIS= option sets major tick marks for the vertical axis. The HMINOR= and VMINOR= options specify the number of tick marks between major tick marks.

```
proc gplot data=stocks;
   plot high*year low*year / overlay legend=legend1
                             vref=1000 to 5000 by 1000
                             lvref=2
                             haxis=axis1 hminor=4
                             vaxis=axis2 vminor=1;
run;
quit;
```

Example 7: Filling Areas in an Overlay Plot

Features:

PLOT statement options
- AREAS=
- HAXIS=
- HMINOR=
- VAXIS=
- VMINOR=
- CAXIS=
- OVERLAY

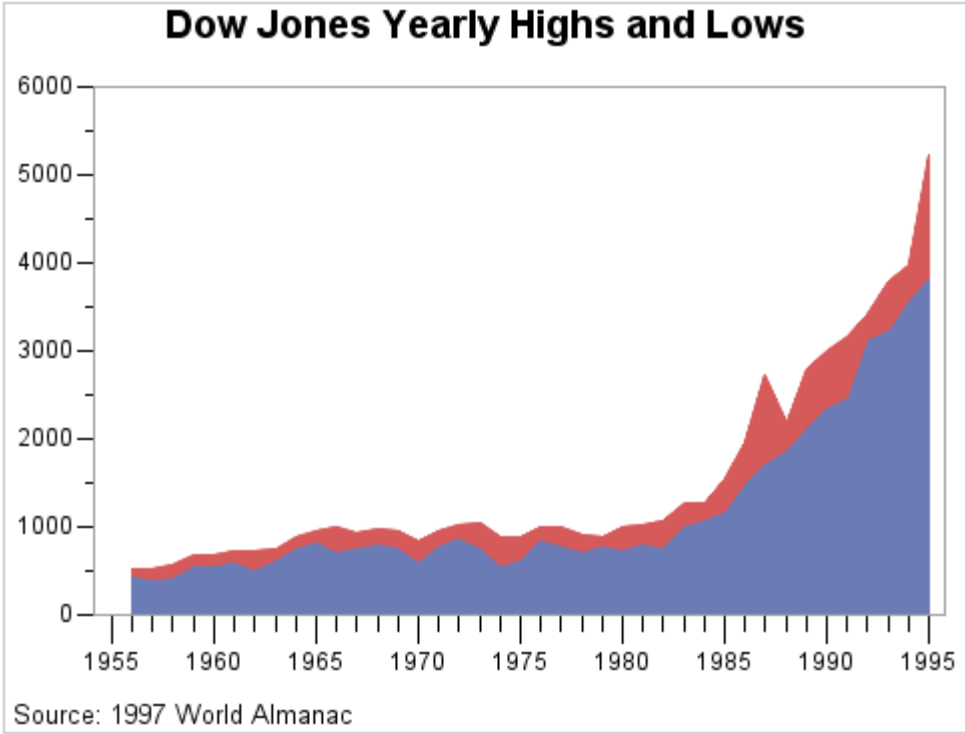
GOPTIONS statement option
- BORDER

SYMBOL statement

Sample library member: GPLFILL1

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses the AREAS= option in the PLOT statement to fill the areas that are under the plot lines. As in the previous example, two plots are overlaid on the same graph.



Program

```
goptions reset=all border;

data stocks;
   input year high low @@;
   datalines;
1956  521.05  462.35 1957  520.77  419.79
1958  583.65  436.89 1959  679.36  574.46
1960  685.47  568.05 1961  734.91  610.25
1962  726.01  535.76 1963  767.21  646.79
1964  891.71  768.08 1965  969.26  840.59
1966  995.15  744.32 1967  943.08  786.41
1968  985.21  825.13 1969  968.85  769.93
1970  842.00  631.16 1971  950.82  797.97
1972 1036.27  889.15 1973 1051.70  788.31
1974  891.66  577.60 1975  881.81  632.04
1976 1014.79  858.71 1977  999.75  800.85
1978  907.74  742.12 1979  897.61  796.67
1980 1000.17  759.13 1981 1024.05  824.01
1982 1070.55  776.92 1983 1287.20 1027.04
1984 1286.64 1086.57 1985 1553.10 1184.96
1986 1955.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
1994 3978.36 3593.35 1995 5216.47 3832.08
;
run;
```

```
title1 "Dow Jones Yearly Highs and Lows";
footnote1 j=l " Source: 1997 World Almanac";

symbol1 interpol=join;

axis1 order=(1955 to 1995 by 5) offset=(2,2)
      label=none
      major=(height=2)
      minor=(height=1);
axis2 order=(0 to 6000 by 1000) offset=(0,0)
      label=none
      major=(height=2)
      minor=(height=1);

proc gplot data=stocks;
   plot low*year high*year / overlay
                             haxis=axis1
                             hminor=4
                             vaxis=axis2
                             vminor=1
                             caxis=black
                             areas=2;
run;
quit;
```

Program Description

Set the graphics environment. BORDER draws a border around the graph.

```
goptions reset=all border;
```

Create the data set. STOCKS contains yearly highs and lows for the Dow Jones Industrial Average and the dates of the high and low values each year.

```
data stocks;
   input year high low @@;
   datalines;
1956  521.05  462.35 1957  520.77  419.79
1958  583.65  436.89 1959  679.36  574.46
1960  685.47  568.05 1961  734.91  610.25
1962  726.01  535.76 1963  767.21  646.79
1964  891.71  768.08 1965  969.26  840.59
1966  995.15  744.32 1967  943.08  786.41
1968  985.21  825.13 1969  968.85  769.93
1970  842.00  631.16 1971  950.82  797.97
1972 1036.27  889.15 1973 1051.70  788.31
1974  891.66  577.60 1975  881.81  632.04
1976 1014.79  858.71 1977  999.75  800.85
1978  907.74  742.12 1979  897.61  796.67
1980 1000.17  759.13 1981 1024.05  824.01
1982 1070.55  776.92 1983 1287.20 1027.04
1984 1286.64 1086.57 1985 1553.10 1184.96
1986 1955.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
1994 3978.36 3593.35 1995 5216.47 3832.08
```

```
;
run;
```

Define the title and the footnote.

```
title1 "Dow Jones Yearly Highs and Lows";
footnote1 j=l " Source: 1997 World Almanac";
```

Define the symbol characteristics. The INTERPOL= option specifies a line to connect data points. The line creates the fill boundary.

```
symbol1 interpol=join;
```

Define the axis characteristics.

```
axis1 order=(1955 to 1995 by 5) offset=(2,2)
      label=none
      major=(height=2)
      minor=(height=1);
axis2 order=(0 to 6000 by 1000) offset=(0,0)
      label=none
      major=(height=2)
      minor=(height=1);
```

Generate a plot with filled areas. The plot requests are ordered to draw the lowest plot first. Area 1 occupies the space between the lowest (first) plot line and the horizontal axis, and area 2 is below the highest (second) plot line. This arrangement prevents the pattern for area 1 from overlaying the pattern for area 2. AREAS=2 fills all the areas below the second plot line.

```
proc gplot data=stocks;
   plot low*year high*year / overlay
                             haxis=axis1
                             hminor=4
                             vaxis=axis2
                             vminor=1
                             caxis=black
                             areas=2;
run;
quit;
```

Example 8: Plotting Three Variables

Features:

PLOT classification variable

PLOT statement options
- HAXIS=
- HMINOR=
- LEGEND=
- VAXIS=
- VMINOR=

GOPTIONS statement option
- BORDER

AXIS statement

SYMBOL statement

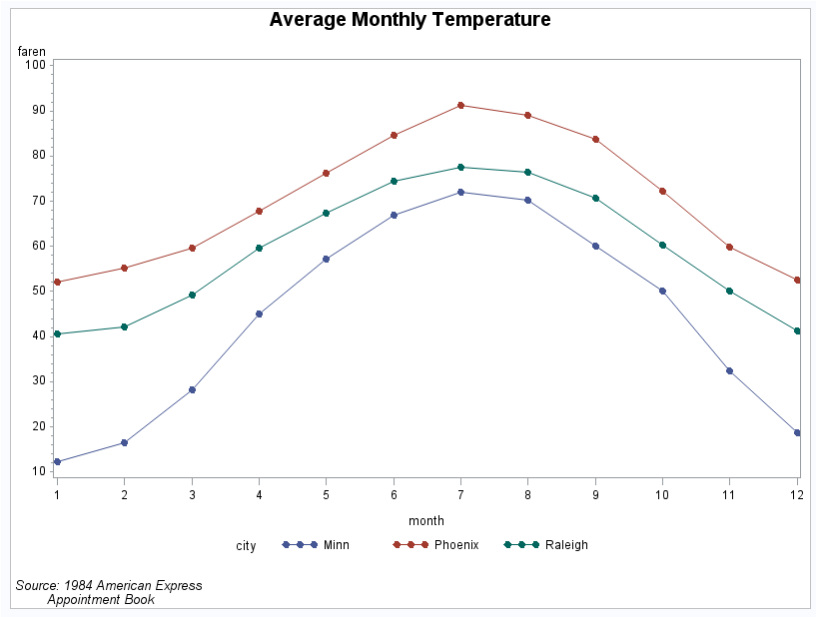
RUN-group processing

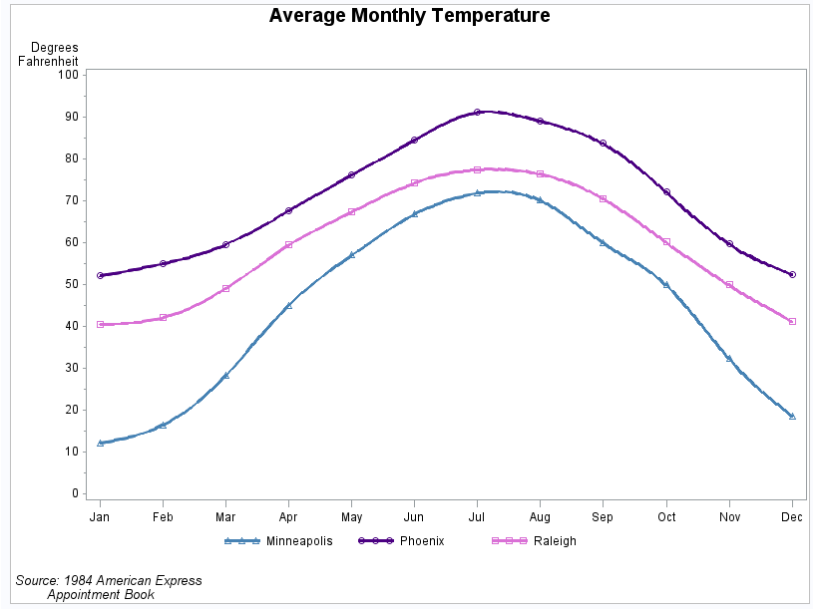
Sample library member: GPLVRBL2

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example shows that when your data contain a classification variable that groups the data, you can use a plot request of the form *y-variable*x-variable=third-variable* to generate a separate plot for every value of the classification variable, which in this case is CITY. With this type of request, all plots are drawn on the same graph and a legend is automatically produced. This legend identifies the values of *third-variable*. The default legend uses the variable name CITY for the legend label and the variable values for the legend value descriptions.

This example then modifies the plot request. As shown in the second output display, the plot is enhanced by using different symbol definitions and colors for each plot line. It is also enhanced by changing axes labels and scaling the vertical axes differently.





Program

```
goptions reset=all border;

proc format;
   value mmm_fmt
   1='Jan'
   2='Feb'
   3='Mar'
   4='Apr'
   5='May'
   6='Jun'
   7='Jul'
   8='Aug'
   9='Sep'
   10='Oct'
   11='Nov'
   12='Dec'
   ;
run;

data citytemp;
   input  month faren city $ @@;
   datalines;
   1      40.5    Raleigh     1      12.2    Minn
   1      52.1    Phoenix     2      42.2    Raleigh
   2      16.5    Minn        2      55.1    Phoenix
   3      49.2    Raleigh     3      28.3    Minn
   3      59.7    Phoenix     4      59.5    Raleigh
   4      45.1    Minn        4      67.7    Phoenix
   5      67.4    Raleigh     5      57.1    Minn
   5      76.3    Phoenix     6      74.4    Raleigh
   6      66.9    Minn        6      84.6    Phoenix
   7      77.5    Raleigh     7      71.9    Minn
```

```
   7      91.2    Phoenix     8      76.5    Raleigh
   8      70.2    Minn        8      89.1    Phoenix
   9      70.6    Raleigh     9      60.0    Minn
   9      83.8    Phoenix    10      60.2    Raleigh
  10      50.0    Minn       10      72.2    Phoenix
  11      50.0    Raleigh    11      32.4    Minn
  11      59.8    Phoenix    12      41.2    Raleigh
  12      18.6    Minn       12      52.5    Phoenix
;

title1 "Average Monthly Temperature";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l "         Appointment Book";

symbol1 interpol=join  value=dot ;

proc gplot data= citytemp;
   plot faren*month=city / hminor=0;
run;

symbol1 interpol=spline width=2 value=triangle c=steelblue;
symbol2 interpol=spline width=2 value=circle c=indigo;
symbol3 interpol=spline width=2 value=square c=orchid;

axis1 label=none
      order = 1 to 12 by 1
      offset=(2);

axis2 label=("Degrees" justify=right  "Fahrenheit")
      order=(0 to 100 by 10);

legend1 label=none value=(tick=1 "Minneapolis");

format month mmm_fmt.;

plot faren*month=city /
        haxis=axis1 hminor=0
        vaxis=axis2 vminor=1
        legend=legend1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create a format for the month values. Format mmm_fmt formats the numeric month values into three-character month names.

```
proc format;
   value mmm_fmt
   1='Jan'
   2='Feb'
   3='Mar'
   4='Apr'
   5='May'
   6='Jun'
```

```
   7='Jul'
   8='Aug'
   9='Sep'
   10='Oct'
   11='Nov'
   12='Dec'
   ;
run;
```

Create the data set. CITYTEMP contains the average monthly temperatures of three cities: Raleigh, Minneapolis, and Phoenix.

```
data citytemp;
   input  month faren city $ @@;
   datalines;
   1      40.5    Raleigh     1      12.2    Minn
   1      52.1    Phoenix     2      42.2    Raleigh
   2      16.5    Minn        2      55.1    Phoenix
   3      49.2    Raleigh     3      28.3    Minn
   3      59.7    Phoenix     4      59.5    Raleigh
   4      45.1    Minn        4      67.7    Phoenix
   5      67.4    Raleigh     5      57.1    Minn
   5      76.3    Phoenix     6      74.4    Raleigh
   6      66.9    Minn        6      84.6    Phoenix
   7      77.5    Raleigh     7      71.9    Minn
   7      91.2    Phoenix     8      76.5    Raleigh
   8      70.2    Minn        8      89.1    Phoenix
   9      70.6    Raleigh     9      60.0    Minn
   9      83.8    Phoenix    10      60.2    Raleigh
  10      50.0    Minn       10      72.2    Phoenix
  11      50.0    Raleigh    11      32.4    Minn
  11      59.8    Phoenix    12      41.2    Raleigh
  12      18.6    Minn       12      52.5    Phoenix
;
```

Define title and footnotes.

```
title1 "Average Monthly Temperature";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l "         Appointment Book";
```

Define symbol for the first plot. This statement specifies that a straight line connect data point. Because no color is specified, the default color behavior is used and each line is a different color.

```
symbol1 interpol=join  value=dot ;
```

Generate a plot of three variables that produces a legend. The plot request draws one plot on the graph for each value of CITY and produces a legend that defines CITY values.

```
proc gplot data= citytemp;
   plot faren*month=city / hminor=0;
run;

symbol1 interpol=spline width=2 value=triangle c=steelblue;
symbol2 interpol=spline width=2 value=circle c=indigo;
symbol3 interpol=spline width=2 value=square c=orchid;
```

Define new axis characteristics. AXIS1 suppresses the axis label and specifies month abbreviations for the major tick mark labels. AXIS2 specifies a two-line axis

label and scales the axis to show major tick marks at every 10 degrees from 0 to 100 degrees.

```
axis1 label=none
      order = 1 to 12 by 1
      offset=(2);

axis2 label=("Degrees" justify=right  "Fahrenheit")
      order=(0 to 100 by 10);
```

Enhance the legend.

```
legend1 label=none value=(tick=1 "Minneapolis");
```

Format variable month. The previously defined Format mmm_fmt formats the numeric month values to three-character month names.

```
format month mmm_fmt.;
```

Generate the enhanced plot. Because the procedure supports RUN-group processing, you do not have to repeat the PROC GPLOT statement to generate the second plot.

```
plot faren*month=city /
        haxis=axis1 hminor=0
        vaxis=axis2 vminor=1
        legend=legend1;
run;
quit;
```

Example 9: Plotting with Different Scales of Values

Features:

PLOT statement options
- HAXIS=
- HMINOR=

PLOT and PLOT2 statement options
- VAXIS=
- VMINOR=

GOPTIONS statement option
- BORDER

AXIS statement

SYMBOL statement

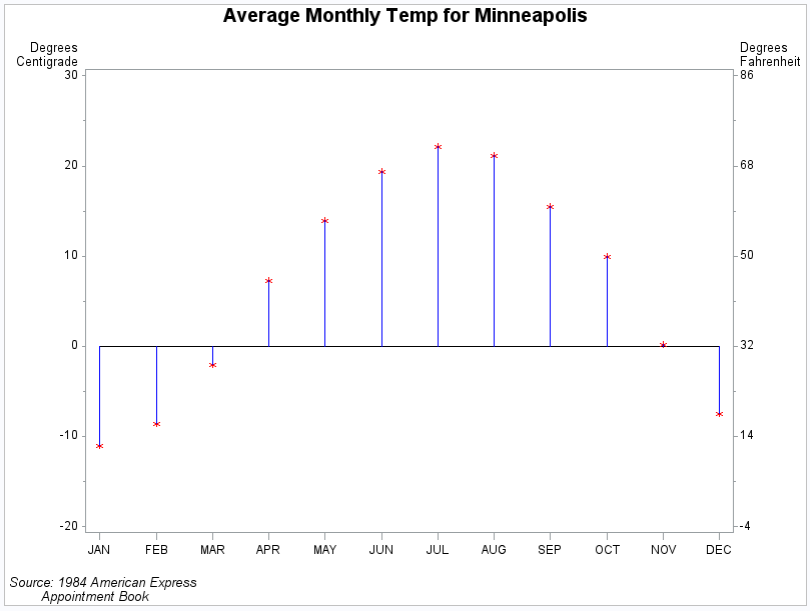
Sample library member: GPLSCVL1

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example shows how a PLOT2 statement generates a right axis that displays the values of the vertical coordinates in a scale different from that used for the left axis. This plot shows the average monthly temperature for Minneapolis. Temperature variables represent degrees centigrade (displayed on the left axis) and degrees Fahrenheit (displayed on the right axis). These temperature variables are

plotted against the variable MONTH. Although the procedure produces two sets of data points, it calibrates the axes so that the data points are identical and it displays only one plot.

This example uses SYMBOL statements to define symbol definitions. By default, the SYMBOL1 statement is assigned to the plot that is generated by the PLOT statement, and SYMBOL2 is assigned to the plot generated by the PLOT2 statement.



Program

```
goptions reset=all border;

proc format;
   value mmm_fmt
   1='JAN'
   2='FEB'
   3='MAR'
   4='APR'
   5='MAY'
   6='JUN'
   7='JUL'
   8='AUG'
   9='SEP'
   10='OCT'
   11='NOV'
   12='DEC'
   ;
run;

data minntemp;
   input @10 month
         @23 f2;     /* fahrenheit temperature for Minneapolis */
         c2=(f2-32)/1.8; /* calculate centigrade temperature */
                         /* for Minneapolis                  */
```

```
   output;
   datalines;
01JAN83  1    1   40.5  12.2  52.1
01FEB83  2    1   42.2  16.5  55.1
01MAR83  3    2   49.2  28.3  59.7
01APR83  4    2   59.5  45.1  67.7
01MAY83  5    2   67.4  57.1  76.3
01JUN83  6    3   74.4  66.9  84.6
01JUL83  7    3   77.5  71.9  91.2
01AUG83  8    3   76.5  70.2  89.1
01SEP83  9    4   70.6  60.0  83.8
01OCT83  10   4   60.2  50.0  72.2
01NOV83  11   4   50.0  32.4  59.8
01DEC83  12   1   41.2  18.6  52.5
;

title1 "Average Monthly Temp for Minneapolis";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l "         Appointment Book";

symbol1 interpol=needle ci=blue cv=red  value=star;

symbol2 interpol=none value=none;

axis1 label=none order=(1 to 12 by 1) offset=(2);

axis2 label=( "Degrees" justify=right " Centigrade")
      order=(-20 to 30 by 10);

axis3 label=( "Degrees" justify=left "Fahrenheit")
      order=(-4 to 86 by 18);

proc gplot data= minntemp;
   format month mmm_fmt.;

   plot c2*month / haxis=axis1 hminor=0
                   vaxis=axis2 vminor=1;

   plot2 f2*month / vaxis=axis3 vminor=1;

run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create a format for the month values. Format mmm_fmt formats the numeric month values into three-character month names.

```
proc format;
   value mmm_fmt
   1='JAN'
   2='FEB'
   3='MAR'
   4='APR'
```

```
   5='MAY'
   6='JUN'
   7='JUL'
   8='AUG'
   9='SEP'
   10='OCT'
   11='NOV'
   12='DEC'
   ;
run;
```

Create the data set and calculate centigrade temperatures. MINNTEMP contains average monthly temperatures for Minneapolis.

```
data minntemp;
   input @10 month
         @23 f2;     /* fahrenheit temperature for Minneapolis */
         c2=(f2-32)/1.8; /* calculate centigrade temperature */
                         /* for Minneapolis                  */
   output;
   datalines;
01JAN83  1    1   40.5  12.2  52.1
01FEB83  2    1   42.2  16.5  55.1
01MAR83  3    2   49.2  28.3  59.7
01APR83  4    2   59.5  45.1  67.7
01MAY83  5    2   67.4  57.1  76.3
01JUN83  6    3   74.4  66.9  84.6
01JUL83  7    3   77.5  71.9  91.2
01AUG83  8    3   76.5  70.2  89.1
01SEP83  9    4   70.6  60.0  83.8
01OCT83  10   4   60.2  50.0  72.2
01NOV83  11   4   50.0  32.4  59.8
01DEC83  12   1   41.2  18.6  52.5
;
```

Define title and footnotes.

```
title1 "Average Monthly Temp for Minneapolis";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l "         Appointment Book";
```

Define symbol characteristics. INTERPOL=NEEDLE generates a horizontal reference line at zero on the left axis and draws vertical lines from the data points to the reference line. CI= specifies the color of the interpolation line and CV= specifies the color of the plot symbol.

```
symbol1 interpol=needle ci=blue cv=red  value=star;
```

Define symbol characteristics for PLOT2. SYMBOL2 suppresses interpolation lines and plotting symbols. Otherwise, they would overlay the lines or symbols displayed by SYMBOL1.

```
symbol2 interpol=none value=none;
```

Define axis characteristics. In the AXIS2 and AXIS3 statements, the ORDER= option controls the scaling of the axes. Both axes represent exactly the same range of temperature. The distance between the major tick marks on both axes represent an equivalent quantity of degrees (10 for centigrade and 18 for Fahrenheit).

```
axis1 label=none order=(1 to 12 by 1) offset=(2);
```

```
axis2 label=( "Degrees" justify=right " Centigrade")
      order=(-20 to 30 by 10);

axis3 label=( "Degrees" justify=left "Fahrenheit")
      order=(-4 to 86 by 18);
```

Generate a plot with a second vertical axis. The previously defined mmm_fmt format is applied to the MONTH variable to format the month numbers into three-character month names. The HAXIS= option specifies the AXIS1 definition. The VAXIS= option specifies AXIS2 and AXIS3 definitions in the PLOT and PLOT2 statements. Axis labels and major tick mark values use the default color. The VMINOR= option specifies the number of minor tick marks for each axis.

```
proc gplot data= minntemp;
   format month mmm_fmt.;

   plot c2*month / haxis=axis1 hminor=0
                   vaxis=axis2 vminor=1;

   plot2 f2*month / vaxis=axis3 vminor=1;

run;
quit;
```

Example 10: Creating Plots with Drill-down Functionality for the Web

Features:

PLOT statement options
- HTML=
- HTML_LEGEND=

ODS HTML statement
- BODY=
- NOGTITLE
- PATH=

GOPTIONS statement option
- BORDER

BY statement

GOPTIONS statement

Sample library member: GPLDRIL1

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

Details

This example shows how to create a plot with simple drill-down functionality for the web. If you display the plot in a web browser, you can select any plot point or legend symbol to display a report on monthly temperatures for the selected city.

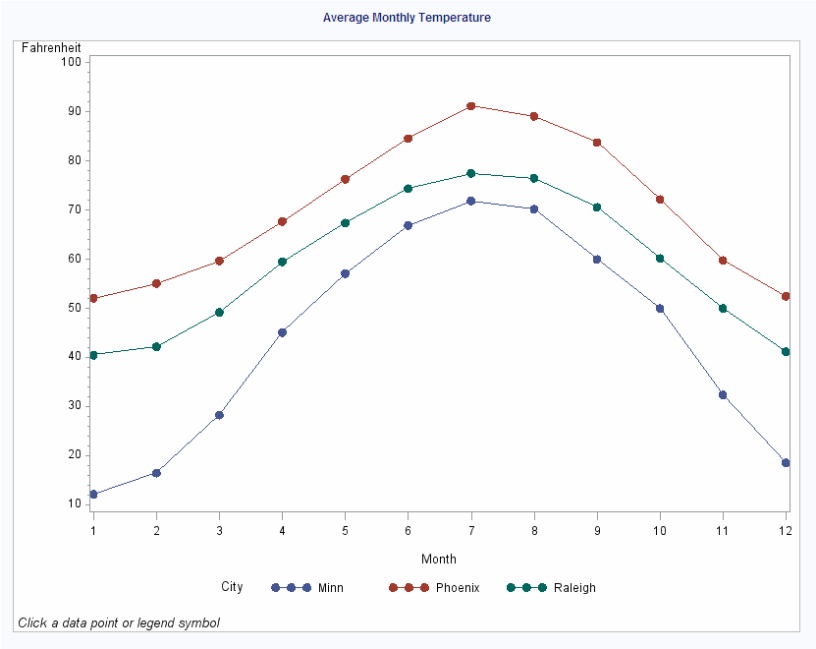
The example explains how to use an ODS statement such as ODS HTML to generate a graph with drill-down links. It shows how to do the following actions:

- explicitly name the HTML files and direct the different types of output to different files
- use BY-group processing with ODS, and determine the anchor names for the different pieces of output
- use the PATH= option to specify the destination for the HTML and GIF files created by the ODS statement
- add an HTML HREF string to a data set to define a link target
- assign link targets with the HTML= and HTML_LEGEND= procedure options
- suppress the titles in the GIF files and display them in the HTML file

For more information about drill-down graphs, see “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 255.

This program modifies the code from sample GPLVRBL2, which shows how to generate separate plots for the formatted values of a classification variable. In this example, the code implements drill-down capability for the plot. This enables you to select any plot point or legend symbol to drill down to a report on the yearly temperatures for the corresponding city. The first figure shows the drill-down plot as it is viewed in a browser.

The second figure shows the report that appears when you select any plot point or legend symbol that corresponds to the data for Raleigh.



Monthly Temperatures in Raleigh

Month	Fahrenheit
1	40.5
2	42.2
3	49.2
4	59.5
5	67.4
6	74.4
7	77.5
8	76.5
9	70.6
10	60.2
11	50
12	41.2

Program

```
filename odsout  ".";

goptions reset=all border device=gif;

ods _all_ close;

ods html path=odsout
         body="city_plots.html"
         nogtitle;

data citytemp;
   input  Month Fahrenheit City $ @@;
   datalines;
   1      40.5    Raleigh     1      12.2    Minn
   1      52.1    Phoenix     2      42.2    Raleigh
   2      16.5    Minn        2      55.1    Phoenix
   3      49.2    Raleigh     3      28.3    Minn
   3      59.7    Phoenix     4      59.5    Raleigh
   4      45.1    Minn        4      67.7    Phoenix
   5      67.4    Raleigh     5      57.1    Minn
   5      76.3    Phoenix     6      74.4    Raleigh
   6      66.9    Minn        6      84.6    Phoenix
   7      77.5    Raleigh     7      71.9    Minn
   7      91.2    Phoenix     8      76.5    Raleigh
   8      70.2    Minn        8      89.1    Phoenix
   9      70.6    Raleigh     9      60.0    Minn
   9      83.8    Phoenix    10      60.2    Raleigh
  10      50.0    Minn       10      72.2    Phoenix
  11      50.0    Raleigh    11      32.4    Minn
  11      59.8    Phoenix    12      41.2    Raleigh
  12      18.6    Minn       12      52.5    Phoenix
;

data newtemp;
   set citytemp;
   length citydrill $ 40;
        if city="Minn" then
      citydrill="HREF='city_reports.html#IDX1'";
   else if city="Phoenix" then
      citydrill="HREF='city_reports.html#IDX2'";
   else if city="Raleigh" then
      citydrill="HREF='city_reports.html#IDX3'";

title1 "Average Monthly Temperature";
footnote1 j=l " Click a data point or legend symbol";

symbol1 interpol=join
        value=dot;

proc gplot data=newtemp;
   plot fahrenheit*month=city / hminor=0
        html=citydrill
        html_legend=citydrill;
run;
quit;

 ods html path=odsout
```

```
body="city_reports.html";

proc sort data=newtemp;
   by city month;
run;

goptions reset=footnote;
option nobyline;

title1 "Monthly Temperatures in #byval(city)";
proc report data=newtemp nowindows;
  by city;
  column city month fahrenheit;
  define city       / noprint group;
  define month      / group;
  define Fahrenheit / group;
run;

ods html close;

ods html; /* Not required in SAS Studio */./
```

Program Description

Define graphics output location.

```
filename odsout  ".";
```

Set the graphics environment.

```
goptions reset=all border device=gif;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Open an HTML output file in ODS.

```
ods html path=odsout
         body="city_plots.html"
         nogtitle;
```

Create the data set CITYTEMP. CITYTEMP contains the average monthly temperatures for three cities.

```
data citytemp;
   input  Month Fahrenheit City $ @@;
   datalines;
   1      40.5    Raleigh     1      12.2    Minn
   1      52.1    Phoenix     2      42.2    Raleigh
   2      16.5    Minn        2      55.1    Phoenix
   3      49.2    Raleigh     3      28.3    Minn
   3      59.7    Phoenix     4      59.5    Raleigh
   4      45.1    Minn        4      67.7    Phoenix
   5      67.4    Raleigh     5      57.1    Minn
   5      76.3    Phoenix     6      74.4    Raleigh
   6      66.9    Minn        6      84.6    Phoenix
   7      77.5    Raleigh     7      71.9    Minn
   7      91.2    Phoenix     8      76.5    Raleigh
   8      70.2    Minn        8      89.1    Phoenix
   9      70.6    Raleigh     9      60.0    Minn
```

```
   9      83.8    Phoenix    10      60.2    Raleigh
  10      50.0    Minn       10      72.2    Phoenix
  11      50.0    Raleigh    11      32.4    Minn
  11      59.8    Phoenix    12      41.2    Raleigh
  12      18.6    Minn       12      52.5    Phoenix
;
```

Add the HTML variable to CITYTEMP and create the NEWTEMP data set. The HTML variable CITYDRILL contains the target locations to associate with the different values of the variable CITY. Each location for CITYDRILL references the file city_reports.html, which this program creates. Each location ends with the default anchor name (IDX1, IDX2, and IDX3) that ODS assigns to the target output when it creates that output in file city_reports.html.

```
data newtemp;
   set citytemp;
   length citydrill $ 40;
        if city="Minn" then
      citydrill="HREF='city_reports.html#IDX1'";
   else if city="Phoenix" then
      citydrill="HREF='city_reports.html#IDX2'";
   else if city="Raleigh" then
      citydrill="HREF='city_reports.html#IDX3'";
```

Define a title and footnote and a symbol definition for the plots.

```
title1 "Average Monthly Temperature";
footnote1 j=l " Click a data point or legend symbol";

symbol1 interpol=join
        value=dot;
```

Generate the plot. Both HTML= and HTML_LEGEND= specify CITYDRILL as the variable that contains the targets for the drill-down links. The HTML= option determines that each plot point is a hot zone that links to target output. The HTML_LEGEND= option determines that the legend symbols are hot zones that link to target output. This GPLOT procedure generates the first piece of output in this program. Thus, the plot receives the first default anchor name, which is IDX.

```
proc gplot data=newtemp;
   plot fahrenheit*month=city / hminor=0
        html=citydrill
        html_legend=citydrill;
run;
quit;
```

Change the HTML file. The BODY= option opens a new HTML file for storing the reports for city temperatures. The new file is assigned the name city_reports.html, which is the filename assigned above to variable CITYDRILL as part of its target-link locations. The reports that are generated later in this program are all written to this one HTML file.

```
 ods html path=odsout
body="city_reports.html";
```

Sort data set NEWTEMP in order by city.

```
proc sort data=newtemp;
   by city month;
run;
```

Clear the footnotes, and suppress the default BY line.

```
goptions reset=footnote;
option nobyline;
```

Print a report of monthly temperatures for each city. The BY statement determines that a separate report is generated for each city. Thus, the REPORT procedure generates three pieces of output. To assign anchor locations to this new output, ODS increments the last anchor name that was used (IDX). Therefore, ODS assigns the anchor names IDX1, IDX2, and IDX3 to the output. These are the anchor locations that were specified above as the anchor locations for variable CITYDRILL.

```
title1 "Monthly Temperatures in #byval(city)";
proc report data=newtemp nowindows;
  by city;
  column city month fahrenheit;
  define city       / noprint group;
  define month      / group;
  define Fahrenheit / group;
run;
```

Close ODS HTML. This closes the HTML output file.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```
ods html; /* Not required in SAS Studio */
```

Chapter 43

GRADAR Procedure

Overview: GRADAR Procedure

About the GRADAR Procedure

The GRADAR procedure creates radar charts, wind rose charts, and calendar charts. Note that the GRADAR procedure is not supported by the Java device drivers.

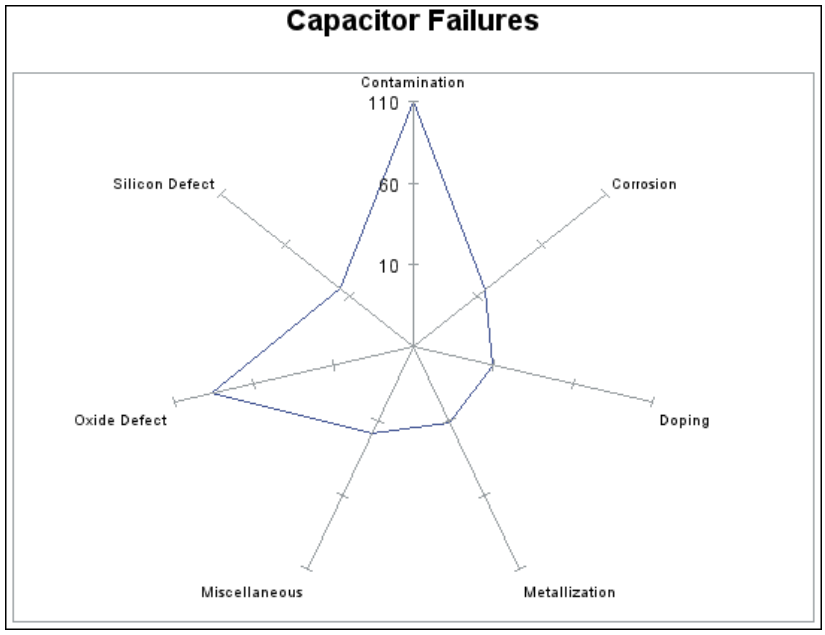
About Radar Charts

Radar charts show the relative frequency of data measures in quality control or market research problems. The chart statistics are displayed along spokes that radiate from the center of the chart. The charts are often stacked on top of one another with reference circles, which gives them the look of a radar screen. By default, the chart vertices—the points where the statistical values intersect the spokes—are based on the frequencies associated with the levels of a single numeric variable. Non-integer values of the chart variable are truncated to integers. The measures can be displayed in decreasing order, the order in which they appear in the input data, increasing order of internal values, or lexicographic order of variable names. Radar charts are sometimes also called *star charts* or *spider charts*. The terms vary depending on the appearance of the chart and the variable types that the chart is depicting.

Note: A calendar chart, circular in format, is generated with the GRADAR procedure.

TIP Radar charts are most effective when the input data set is limited to a few hundred points.

Figure 43.1 *Basic Radar Chart (GRRBASIC)*



If the radar chart includes all of the axes and the reference lines, it is commonly referred to as a spider chart. The program for this chart is in "Example 1: Producing a Basic Radar Chart" on page 1400. For more information about producing a radar chart, see the "CHART Statement" on page 1379.

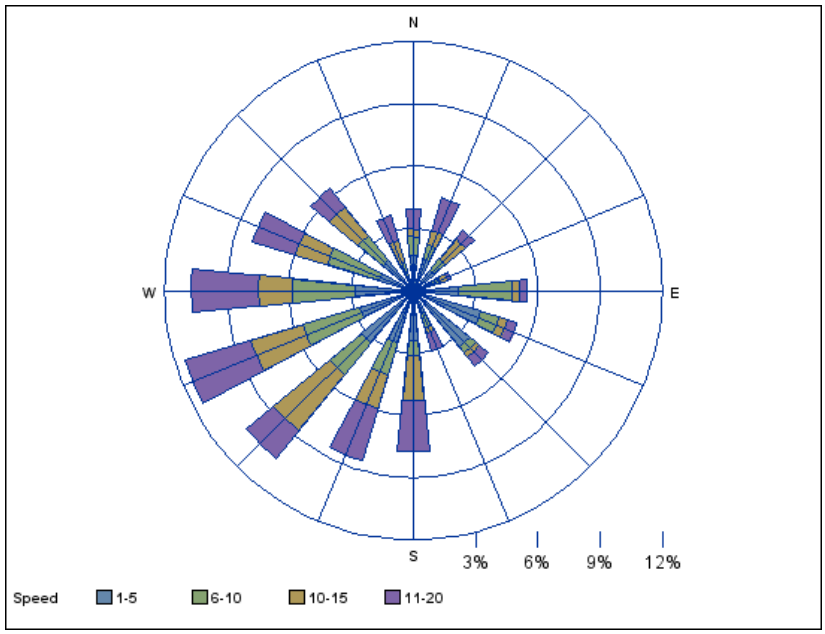
About Wind Rose Charts

A wind rose chart displays wind direction and speed over a period of time at a specific location. The segments in a wind rose chart represent the percentage of time that the wind blew from each direction.

The wind data must be sorted by direction. You can enter data for four, eight, or sixteen compass points.

The angular direction of the axes, as in this example, can be significant.

Figure 43.2 *Wind Rose Chart (GRRWNDRS)*

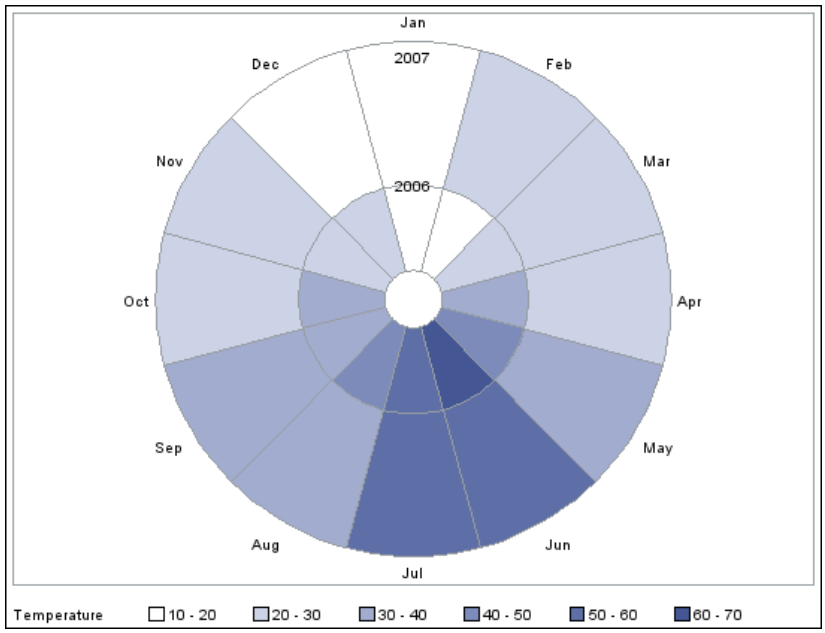


The program for this chart is in "Example 8: Creating a Wind Rose Chart" on page 1411. For more information about producing a radar chart, see the "CHART Statement" on page 1379.

About Calendar Charts

Calendar charts display twelve equal-sized segments, one for each month of the year. The color shading of each segment represents the magnitude of the frequency variable. This calendar chart is overlaid with climate data showing temperatures for each month of the years 2006 and 2007.

Figure 43.3 Calendar Charts (GRRCALEN)



The program for this chart is in "Example 9: Creating a Calendar Chart" on page 1412. For more information about producing a radar chart, see the "CHART Statement" on page 1379.

Syntax: GRADAR Procedure

Restrictions: Not supported by Java

The SAS 'DOLLAR8.2' FORMAT is not supported by the ActiveX device.

Requirement: At least one CHART statement is required.

Supported Statements: AXIS, FOOTNOTE, GOPTIONS, TITLE

Supports: RUN-group processing

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, "Plotting a Cloud Analytic Services (CAS) In-Memory Table," on page 153.

The procedure can include the SAS/GRAPH statements BY on page 424 and NOTE on page 507, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See "SAS/GRAPH Statements" on page 397 and *SAS DATA Step Statements: Reference* for more information.

Tip: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, `ODS PDF; procedure-code; ODS PDF CLOSE;`). See Chapter 7, "Using Run-Group Processing," on page 75 for more information.

PROC GRADAR <DATA=*input-data-set*>

<GOUT=<*libref.*>*output-catalog*>
<ANNOTATE=*Annotate-data-set*>;
CHART *chart-variable* <*/ options*>;
QUIT;

PROC GRADAR Statement

Identifies the data set that contains the plot variables. Specifies an output catalog (optional).

Requirement: An input data set is required.

Syntax

PROC GRADAR <DATA=*input-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<ANNOTATE=*Annotate-data-set*>;

Optional Arguments

PROC GRADAR statement options affect all graphs produced by the procedure.

ANNOTATE=*Annotate-data-set*
specifies a data set to add annotate elements to all graphs that are produced by the GRADAR procedure. To add annotate elements to individual graphs, use ANNOTATE= in the CHART statement.

Alias ANNO=

Restriction The GRADAR procedure does not support coordinate systems 2 or 8. See "Coordinate Systems" on page 734.

See Chapter 28, "Using Annotate Data Sets," on page 721

DATA=*input-data-set*
specifies the SAS data set that contains the variable or variables to chart. By default, the procedure uses the most recently created SAS data set.

GOUT=<*libref.*>*output-catalog*
specifies the SAS catalog in which to save the graphics output produced by the GRADAR procedure.

CHART Statement

Creates the radar charts in which the length of the vertices along the spines represent the values of the chart statistic for the data categories.

Requirement: At least one chart variable is required.

Syntax

CHART *chart-variable* <*/ options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to add annotate elements to charts produced by the CHART statement.

CAXIS=*color*
: specifies a color for the chart frame and the spokes or grid lines of the chart.

CFRAME=*color* | (*variable*)
: fills the area inside the frame or the individual tiles with the specified color.

CFRAMESIDE=*color* | (*variable*)
: specifies the color for filling the area for the row labels displayed along the left side of a chart.

CFRAMETOP=*color* | (*variable*)
: specifies the color for filling the area for the column labels that are displayed across the top of a chart.

CSPOKES=*spoke-color* | (*variable*)
: specifies a color to use for the spokes in a chart.

CSTARCIRCLES=*color* | (*colors-list*)
: specifies a color or list of colors for the circles that are requested with the STARCIRCLES= option.

CSTARFILL=*color* | (*color-list*)
: specifies a color or colors for filling the interior of stars when STARFILL= is set to SOLID.

CSTARS=*color* | (*color-list*)
: specifies a color or list of colors for the outlines of stars.

CTEXT=*color*
: specifies a color for all text on the chart.

CTILES=(*variable*) | color
: specifies one or more colors for the tiles.

FONT=*font*
: specifies the font for all text strings in the radar chart.

FRAME

NOFRAME
: specifies whether a frame is drawn around the chart.

HEIGHT=*height*
: specifies the height in cells for labels and legends.

IFRAME=*fileref* | "*external-image-file*"
: specifies an image file to display inside the chart's frame.

IMAGESTYLE=TILE | FIT
: specifies how to display the image file that is specified on the IFRAME= option.

INBORDER

generates an inside border around the chart.

INHEIGHT=*value*
specifies the height for spoke labels.

INTERTILE=*value*
specifies the distance (in cells) between tiles in a chart.

LSPOKES=*linetype*
specifies a line type for the spokes in a radar chart.

LSTARCIRCLES=*linetypes* | (*linetypes*)
specifies one or more line types for the circles requested with the STARCIRCLES= option.

LSTARS=(*linetypes*)
specifies the line types for the outlines of stars that are produced for a radar chart.

MAXNVERT=*n*
specifies the maximum number of vertices, from 1 to 360, in the radar chart.

NOLEGEND
suppresses the legend that is otherwise automatically displayed.

SPIDERWEB
displays lines connecting the points where tick marks would be instead of displaying the tick marks.

SPKLABEL=CATEGORY | ALL | NONE
labels the chart spokes with the category of the variable that is being charted.

SPOKESCALE=CATEGORY | VERTEX
specifies whether every spoke is drawn to the same scale, or whether each spoke is drawn to a different scale.

STARCIRCLES=(*values*)
specifies reference circles that are superimposed on the stars that are produced for a radar chart.

STARFILL= lists of SOLID | EMPTY, one for each star
determines whether the stars in the radar chart are empty or filled with a solid color.

STARINRADIUS=*percent*
specifies the inner radius of stars as a percent from 0 to 100.

STARLEGEND=CLOCK | CLOCK0 | NUMBER | DEGREES | NONE
specifies the style of the legend used to identify the vertices of stars that are produced for a radar chart.

STARLEGENDLAB="*legend-label*"
specifies the label displayed to the left of the legend for stars requested with the STARLEGEND= option.

STAROUTRADIUS=*value*
specifies the outer radius of stars as a percent up to 100.

STARSTART=*value*
specifies the vertex angle for the first variable that is specified in the CHART statement.

TILELEGEND=(*variable*)
specifies a variable used to add a legend for CTILES= colors.

TILELEGLABEL="*label*"
specifies a label displayed to the left of the legend that is created when you specify a TILELEGEND= variable.

WFRAME=*n*
specifies the width in pixels for the frame lines.

WINDROSECIRCLES=*n*
specifies the number of reference circles.

WSPOKES=*line-width*
specifies the width in pixels of the spokes in a radar chart.

WSTARCIRCLES=(*line-widths*)
specifies the width in pixels of the outline of circles requested by the STARCIRCLES= and WINDROSECIRCLES= options.

WSTARS=*line-width* | (*line-widths*)
specifies the width in pixels of the outline of stars that are produced for a radar chart.

Axis options

STARAXIS= (AXIS<1...99><, . . . ,AXIS<1...99>>)
assigns one or more axis definitions to the axis spokes in the radar chart.

WAXIS=
See WFRAME=.

Catalog entry description options

DESCRIPTION="*description*"
specifies a description of the output.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Chart options

ACROSSVAR=*variable*
generates a radar chart for each value (including missing values) of the specified variable and displays the charts from left-to-right across the output area

CALENDAR
produces a chart displaying 12 equal-sized segments, one for each month of the year January through December.

DOWNVAR=*variable*
generates a radar chart for each value of the specified variable, and displays the charts from top-to-bottom.

FREQ=*numeric-variable*
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

MISSING
accepts a missing value as a valid midpoint for the chart variable.

MODE=SHARE | PROTECT | RESERVE
specifies the display mode for a radar chart.

NCOLS=*n*
specifies the number of columns in a chart.

NLEVELS=*n*
specifies the number of colors used in the calendar chart to represent the magnitude of the frequency variable.

NOZEROREF
turns off the zero reference circle when negative values are plotted.

NROWS=*n*

specifies the number of rows in a chart.

OTHER="*category*"
specifies a new category that merges all categories not selected because of the MAXNVERT= option.

OVERLAYVAR=*overlay-variable*
creates a comparative radar chart using the levels of the overlay variable.

SPEED=*speed-variable*
specifies the wind speed in wind rose charts.

STARTYPE=CORONA | POLYGON | RADIAL | SPOKE | WEDGE
specifies the style of the stars that are produced for a radar chart.

SUMVAR=*summary-variable*
specifies a numeric variable to be used to construct weighted radar charts.

WINDROSE
specifies a wind rose chart.

ODS options

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Required Argument

chart-variable(s)
specifies one or more variables that define the categories of data to be charted. The values of the chart variable determine the spokes in the corresponding radar chart. These values are the observations in the input data set for the chart variable. You must have at least three observations in the data set as it takes three points to define a plane. Technically, you can create a GRADAR chart with only one or two observations, but a true radar chart is not displayed.

Optional Arguments

Options in a CHART statement affect all graphs produced by that statement. You can specify as many options as needed and list them in any order.

ACROSSVAR=*variable*
generates a radar chart for each value (including missing values) of the specified variable and displays the charts from left-to-right across the output area . If used with the DOWN= option, the charts are drawn in left-to-right and top-to-bottom order. To limit the number of columns or rows that are displayed, specify the NCOLS= and NROWS= options.

Alias ACROSS=

Restriction This option is ignored if you specify the WINDROSE, CALENDAR, or OVERLAYVAR= options.

See DOWNVAR= on page 1388, NCOLS= on page 1392, and NROWS= on page 1393

Examples "Example 3: Tiling Radar Charts" on page 1402

"Example 4: Using Multiple Classification Variables in Radar Charts" on page 1403

ANNOTATE=*Annotate-data-set*

specifies a data set to add annotate elements to charts produced by the CHART statement.

Alias ANNO=

Restriction The GRADAR procedure does not support coordinate systems 2 or 8. See "Coordinate Systems" on page 734.

See Chapter 28, "Using Annotate Data Sets," on page 721

CALENDAR

produces a chart displaying 12 equal-sized segments, one for each month of the year January through December. The color shading of each segment represents the magnitude of the frequency variable. Use the OVERLAY variable to subdivide each segment (for example, by year).

Restriction When you specify the CALENDAR option, you must also specify the OVERLAYVAR= option.

Tip Missing values are treated as zeros. For any month in which the frequency variable is missing, the color used for that chart segment is the color for zero.

See FREQ= on page 1389, NLEVELS= on page 1393, and OVERLAYVAR= on page 1393

Example "Example 9: Creating a Calendar Chart" on page 1412

CAXIS=*color*

specifies a color for the chart frame and the spokes or grid lines of the chart. The specified color must be a valid SAS/GRAPH color name.

Alias CAXES=, CA=

Default For ActiveX, the default color is black, and is unchangeable. For other devices, when the NOGYSTYLE system option is not specified, the default color is retrieved from the current style. If NOGSTYLE is specified, the default color is the first color in the color list.

Style reference Color attribute of the GraphAxisLines element

Restriction Not supported by ActiveX

Interactions For the spokes of a chart, the CSPOKES= option overrides the CAXIS= option.

The NOGSTYLE option suppresses use of the current style's color.

Note Avoid using a style with a dark background when specifying an ActiveX device with its default color of black for its chart frame and spokes.

See CFRAME= on page 1385 CSPOKES= on page 1385

CFRAME=*color* | (*variable*)
fills the area inside the frame or the individual tiles with the specified color. You can specify a valid SAS/GRAPH color name, or a character variable whose value is the background color.

Alias CFR=

Interaction The CTILES= option overrides the CFRAME= option.

See CAXIS= on page 1384

CFRAMESIDE=*color* | (*variable*)
specifies the color for filling the area for the row labels displayed along the left side of a chart. The specified color must be a valid SAS/GRAPH color name or a character variable whose value is a valid SAS/GRAPH color name. If a label is associated with the classification variable, the specified color is also used to fill the area for this label. By default, these areas are not filled.

Restriction The CFRAMESIDE= option is ignored unless you also specify the DOWNVAR= option.

CFRAMETOP=*color* | (*variable*)
specifies the color for filling the area for the column labels that are displayed across the top of a chart. The specified color must be a valid SAS/GRAPH color name, or a character variable whose value is a valid SAS/GRAPH color name. If a label is associated with the classification variable, the specified color is also used to fill the area for this label. By default, these areas are not filled.

Restriction The CFRAMESIDE= option is ignored unless you also specify the ACROSSVAR= option.

CSPOKES=*spoke-color* | (*variable*)
specifies a color to use for the spokes in a chart. The specified color must be a valid SAS/GRAPH color name, or a character variable whose value is the color. The default color is specified by the current style or is the first color in the color list if the NOGSTYLE option is specified.

Alias CSPOKE=

Restriction The CSPOKES= option is ignored if you specify the CALENDAR or WINDROSE options.

CSTARCIRCLES=*color* | (*colors-list*)
specifies a color or list of colors for the circles that are requested with the STARCIRCLES= option. All specified colors must be valid SAS/GRAPH color names. By default, the color specified with the CSTARS= option is used. If the CSTARS= option is omitted, the default color is specified by the current style or is the first color in the color list if the NOGSTYLE option is specified.

Alias CSTARCIRCLE=

Example “Example 2: Overlaying Radar Charts” on page 1401

CSTARFILL=*color* | (*color-list*)

specifies a color or colors for filling the interior of stars when STARFILL= is set to SOLID. All specified colors must be valid SAS/GRAPH color names.

If STARFILL is set to SOLID, the GRADAR procedure fills the stars with the first set of colors that it finds from the following list:

- the color or colors specified on the CSTARFILL= option
- the color or colors specified on the CSTARS= option
- the color or colors specified by the current style or, if the NOGSTYLE option is specified, the colors in the device color list.

The number of colors that you specify depends on the number of stars in the chart.

- If the OVERLAY= option is not used, all stars are filled with the same color. Specify a single fill color. If the ACROSSVAR= option or the DOWNVAR= option are used, the specified color is applied to each star in the tiled display.
- If the OVERLAY= option is used, the chart contains multiple overlaid stars. In that case, specify a list of colors in parentheses. Make sure that there are at least as many colors in the list as there are stars in the chart. If you do not specify enough colors for each star to have a different color, the GRADAR procedure assigns colors from the current style (or the device color list) to the remaining stars. (If the NOGSTYLE option is specified, the color for the star positioned at subgroup *n* on the chart is the value of the color corresponding to the color at position *n* in the device color list.)

If the CSTARFILL= option is specified and the CSTARS= option is not specified for the outline, then the outline is the same as the CSTARFILL option.

If the STARFILL= option is not set or is set to EMPTY, then the CSTARFILL= option sets only the outline color. You can also use the CSTARS= option to set the outline color.

See CSTARS= on page 1386

CSTARS=*color* | (*color-list*)

specifies a color or list of colors for the outlines of stars. All specified colors must be valid SAS/GRAPH color names. The GRADAR procedure uses the first set of colors that it finds from the following list:

- the color or colors specified on the CSTARS= option
- the color or colors specified on the CSTARFILL= option
- the color or colors specified by the current style or, if the NOGSTYLE option is specified, the colors in the device color list, starting with the second color.

The number of colors that you specify depends on the number of stars in the chart.

- If the OVERLAY= option is not used, all stars are filled with the same color. Specify a single fill color. If the ACROSSVAR= option or the DOWNVAR= option are used, the specified color is applied to each star in the tiled display.
- If the OVERLAY= option is used, the chart contains multiple overlaid stars. In that case, specify a list of colors in parentheses. Make sure that there are at least as many colors in the list as there are stars in the chart. If you do not specify enough colors for each star to have a different color, the GRADAR procedure assigns colors from the current style (or the device color list) to the remaining stars. (If the NOGSTYLE option is specified, the color for the star

positioned at subgroup *n* on the chart is the value of the color corresponding to the color at position *n* in the device color list.)

Alias CSTAR=

See CSTARFILL= on page 1386

Example "Example 5: Modifying the Appearance of Radar Charts" on page 1405

CTEXT=*color*

specifies a color for all text on the chart. The specified color must be a valid SAS/GRAPH color name.

Defaults For ActiveX, the default color is black. For other devices, the GRADAR procedure uses the first color that it finds in the following list.

1. the CTEXT= option in a GOPTIONS statement
2. the color specified by the current style
3. if the NOGSTYLE option is specified, the second color in the color list

Style reference Color attribute of the GraphValueText element

Note Use the CTEXT= option to specify a more visible color if you are using a style with a dark background, and an ActiveX device with its default text color of black.

CTILES=(*variable*) | color

specifies either a character variable whose values are the fill colors for the tiles or a single color that is the fill color for all tiles. By default, the tiles are not filled.

If you specify a variable, the values of the specified variable must be identical for all observations with the same level of the classification variables. The same color can be used to fill more than one tile. Use the special value, EMPTY, to indicate that a tile is not to be filled.

The CTILES= option cannot be used in conjunction with the NOFRAME option or the CFRAME= option. You can use the TILELEGEND= option in conjunction with the CTILES= option to add an explanatory legend for the CTILES= option colors at the bottom of the chart.

Alias CTILE=

Restriction Not supported by ActiveX

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" on page 251 for more information.

- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the entry-description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default RADAR CHART OF *chart-variable*

DOWNVAR=*variable*

generates a radar chart for each value of the specified variable, and displays the charts from top-to-bottom. If used with the ACROSS= option, the charts are drawn in left-to-right and top-to-bottom order. To limit the number of columns or rows that are displayed, use the NCOLS= option or the NROWS= option.

Alias DOWN=

Restrictions This option is ignored if you specify the WINDROSE, CALENDAR, or OVERLAYVAR= options.

The DOWNVAR= option ignores missing values of *variable*.

Example "Example 4: Using Multiple Classification Variables in Radar Charts" on page 1403

FONT=*font*

specifies the font for all text strings in the radar chart. If you omit the FONT= option, the font that is specified by the FTEXT= graphics option is used. If neither option is specified, the default font is specified by the current style or, if the NOGSTYLE option is specified, by the current device.

Style reference Font attribute of the GraphValueText element

FRAME | NOFRAME

specifies whether a frame is drawn around the chart. FRAME draws a frame inside the border specified by INBORDER (if INBORDER is specified). NOFRAME suppresses the frame.

By default, the frame color is specified by the current style or, if the NOGSTYLE option is specified, is the first color in the color list. If you want to specify a different color, use the CFRAME= option to fill the frame with a specified color or the CAXIS= option to specify the color of the frame.

Default FRAME

Restriction The NOFRAME option cannot be specified with the CFRAME= option or the CTILES= option. This option is not supported by ActiveX.

See CAXIS= on page 1384 and CFRAME= on page 1385

Example "Example 8: Creating a Wind Rose Chart" on page 1411

FREQ=*numeric-variable*

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times that are specified by the value of *numeric-variable* for that observation. If the value of *numeric-variable* is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of *numeric-variable* are truncated to integers.

The statistics are not affected by applying a format to *numeric-variable*.

See "Calculating Weighted Statistics" on page 1399

Examples "Example 1: Producing a Basic Radar Chart" on page 1400

"Example 2: Overlaying Radar Charts" on page 1401

"Example 3: Tiling Radar Charts" on page 1402

HEIGHT=*height*

specifies the height in cells for labels and legends.

Alias HLABEL=

Interaction The HEIGHT= option overrides the HTEXT= option in a GOPTIONS statement. This does not change the size of titles or footnotes.

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend..

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

IFRAME=*fileref* | "*external-image-file*"

specifies an image file to display inside the chart's frame. *Fileref* must be a valid SAS fileref up to eight characters long and must have been previously assigned with a FILENAME statement. *External-image-file* must specify the complete filename of the image file that you want to use. The format of *external-image-file* varies across operating environments. For more information, see "Displaying an Image in Graph Frame" on page 387 .

Restriction Not supported by ActiveX

Interactions This option is overridden by the NOIMAGEPRINT graphics option.

This option is ignored if you specify the NOFRAME option or if you specify the STYLE=0 option in the AXIS statement.

IMAGESTYLE=TILE | FIT
specifies how to display the image file that is specified on the IFRAME= option. TILE copies the image as many times as needed to fit inside the frame. FIT stretches the image so that a single copy fits within the frame.

Note When used with the IFRAME option, the IMAGESTYLE option must be within the PROC statement. When used with the IBACK option, the IMAGESTYLE option must be specified in the GOPTIONS statement.

INBORDER
generates an inside border around the chart. This border is inside the border created by the BORDER option in the GOPTIONS statement, if it is specified.

Restriction Not supported by ActiveX

INHEIGHT=*value*
specifies the height for spoke labels. The default unit is PCT, which is percentage of graphics output area. The INHEIGHT= option overrides the HTEXT= option in a GOPTIONS statement. This option does not change the size of titles or footnotes.

Restriction Not supported by ActiveX

See HEIGHT=

Example "Example 7: Modifying Chart Axes" on page 1408

INTERTILE=*value*
specifies the distance (in cells) between tiles in a chart, and is used only with the ACROSSVAR= option and the DOWNVAR= option. By default, the tiles are contiguous (*value*=0).

Alias INTERCHART=

Default 0

Example "Example 4: Using Multiple Classification Variables in Radar Charts" on page 1403

LSPOKES=*linetype*
specifies a line type for the spokes in a radar chart.

Default 1 (solid line)

LSTARCIRCLES=*linetypes* | (*linetypes*)
specifies one or more line types for the circles requested with the STARCIRCLES= option. If the number of line types specified with LSTARCIRCLES= matches the number of circles requested with STARCIRCLES=, then the line types are paired with the circles in the order specified. If you request more circles than you specify lines types for, SAS/GRAPH uses the line types that you specify and defaults to 1 (solid) for the remaining circles.

Alias LSTARCIRCLE=

Default 1 (solid line)

LSTARS=(*linetypes*)
specifies the line types for the outlines of stars that are produced for a radar chart. By default, the outlines rotate through the list of line types. The default line type for the star positioned at subgroup *n* is the value of the line type corresponding to the position *n* in the list of line types.

The number of line types that you specify depends on the number of stars in the chart.

- If the OVERLAY= option is not used, all stars use the same line type. Specify a single fill line type. If the ACROSSVAR= option or the DOWNVAR= option are used, the specified line type is applied to each star in the tiled display.
- If the OVERLAY= option is used, the chart contains multiple overlaid stars. In that case, specify a list of line types in parentheses. Make sure that there are at least as many line types in the list as there are stars in the chart.

To specify line colors, use the CSTARS= option.

Alias LSTAR=

Example "Example 5: Modifying the Appearance of Radar Charts" on page 1405

MAXNVERT=*n*
specifies the maximum number of vertices, from 1 to 360, in the radar chart.

Alias MAXVERT=

MISSING
accepts a missing value as a valid midpoint for the chart variable. By default, observations with missing values are ignored.

MODE=SHARE | PROTECT | RESERVE
specifies the display mode for a radar chart.

SHARE
shares the drawing space between the text and the graph.

PROTECT
shares the drawing space but maintains a solid rectangle (using the background color) behind the text. This is useful when the text is illegible because of the image specified with the IFRAME= option or the color specified with the CFRAME= option.

RESERVE
reduces the size of the text and graph in order to accommodate both.

Default RESERVE

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**

- The name is truncated to eight characters.
- The first character is always represented in uppercase, and all other characters are represented in lowercase.
- If the name begins with a number, an underscore is prepended to the name.
- If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

For the graphics output file name:

- The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
- All characters are represented in lowercase.
- If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
- If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

  Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default RADAR

NCOLS=*n*

specifies the number of columns in a chart. You can use the NCOLS= option in conjunction with the NROWS= option. NCOLS=2 and NROWS=2 if two classification variables are specified. If used with the ACROSSVAR= or DOWNVAR= options, the default number of columns or rows is calculated from the number of classifications for the variables that are listed on the ACROSSVAR= or DOWNVAR= options. In that case, you can use the NCOLS= option and NROWS= option to limit the number of columns and rows that are specified.

Alias NCOL=

Default 1 if one classification variable is specified

Restriction Not supported by ActiveX

See NROWS= on page 1393

Example "Example 4: Using Multiple Classification Variables in Radar Charts" on page 1403

NLEVELS=*n*
specifies the number of colors used in the calendar chart to represent the magnitude of the frequency variable. The colors are shown in the legend as a color ramp ranging from white to the full intensity of one color.

Default 6

NOLEGEND
suppresses the legend that is otherwise automatically displayed.

NOZEROREF
turns off the zero reference circle when negative values are plotted. When a negative value is plotted, a dashed circle indicates the zero position. You cannot change the appearance of this zero reference circle, but you can turn it off with the NOZEROREF option. The zero reference circle does not appear if there are no negative values plotted.

Restriction Not supported by ActiveX

NROWS=*n*
specifies the number of rows in a chart. You can use the NROWS= option in conjunction with the NCOLS= option. See NCOLS= for details.

Alias NROW=

Default 1

Restriction Not supported by ActiveX

See NCOLS= on page 1392

Example "Example 4: Using Multiple Classification Variables in Radar Charts" on page 1403

OTHER="*category*"
specifies a new category that merges all categories not selected because of the MAXNVERT= option. The category should be specified as a formatted value of the chart variable.

Restriction The OTHER= option is ignored unless you also specify the MAXNVERT= option.

OVERLAYVAR=*overlay-variable*
creates a comparative radar chart using the levels of the overlay variable. All charts are displayed in the same set of spokes. A maximum of 24 overlays can be displayed in a single radar chart.

Alias OVERLAY=

Restriction This option cannot be used with the ACROSSVAR= option or the DOWNVAR= options.

Examples "Example 2: Overlaying Radar Charts" on page 1401

"Example 9: Creating a Calendar Chart" on page 1412

"Example 7: Modifying Chart Axes" on page 1408

SPEED=*speed-variable*
specifies the wind speed in wind rose charts.

SPIDERWEB
displays lines connecting the points where tick marks would be instead of displaying the tick marks, using the same number of points for all axes as for the first axis. The default number of web lines is three.

If there is an AXIS statement in effect, then the web gets its values (such as number, thickness, and color) from the MAJOR= values for the axis drawn at the first position. (The default first position is 12 o'clock.)

Alias SPIDER

Restriction This option is ignored if you also specify the CALENDAR or WINDROSE option.

SPKLABEL=CATEGORY | ALL | NONE
labels the chart spokes with the category of the variable that is being charted.

You can specify this option starting with SAS 9.4M2. When you specify the SPKLABEL=ALL option, the GRADAR procedure generates a spoke (axis) label for each graph displayed in the graphics output area. Without the ALL specification, only the first graph displayed has the spoke (axis) labels, and labels for subsequent graphs are suppressed.

NONE suppresses the labels.

Default CATEGORY However, if the STARLEGEND= option is specified, the default is NONE.

Restriction Not supported by ActiveX device

SPOKESCALE=CATEGORY | VERTEX
specifies whether every spoke is drawn to the same scale, or whether each spoke is drawn to a different scale. When you specify the SPOKESCALE=CATEGORY option (or you do not specify the SPOKESCALE option), the GRADAR procedure determines the minimum and maximum value of all the spokes. Then, the vertices of all the spokes have that same maximum value and minimum value. When you specify the SPOKESCALE=VERTEX option, each vertex has its own maximum value and minimum value, and each vertex is labeled at the tick marks.

Restriction If you specify SPOKESCALE=VERTEX, you must also specify the OVERLAYVAR=, ACROSSVAR=, or DOWNVAR= option.

STARAXIS= (AXIS<1...99><, . . . ,AXIS<1...99>>)
assigns one or more axis definitions to the axis spokes in the radar chart. GRADAR displays axis spokes clockwise, starting at the 12 o'clock position. The axis definitions that are specified using the STARAXIS= option are assigned consecutively to the spokes, starting from the first spoke. AXIS statements are assigned in clockwise order. For example, the STARAXIS=(AXIS3, AXIS1, AXIS2) option assigns the AXIS3 statement's definition to the first axis spoke (at the 12 o'clock position), the AXIS1 statement's definition to the second axis spoke, and the AXIS2 statement's definition to the third axis spoke.

The axis definitions are assigned consecutively, and you cannot skip a spoke. For example, to assign a definition to the seventh spoke, you must also assign

definitions to the first six spokes. However, you do not have to assign definitions to all of the spokes. Any remaining axis spokes on the GRADAR chart are displayed with the default settings. For example, if the STARAXIS= option specifies three definitions and the chart has more than three axis spokes, the fourth and remaining spokes are displayed with the default settings. If there are more definitions specified than there are axis spokes in the chart, the excess definitions are ignored.

Alias STARAXES=

Example "Example 7: Modifying Chart Axes" on page 1408

STARCIRCLES=(*values*)

specifies reference circles that are superimposed on the stars that are produced for a radar chart. All of the circles are displayed and centered at each point plotted on the primary chart. The *value* determines the diameter of the circle. A value of 0.0 specifies a circle with the *inner radius*, which displays a circle at the minimum data value. A value of 1.0 specifies a circle with the *outer radius*, which is the length of the spokes in the chart. In general, a value of *h* specifies a circle with a radius equal to *inradius* + *h*×(*outradius* - *inradius*).

For example, the values 0.0 and 1.0 correspond to an *inner circle* and an *outer circle*. The value 0.5 specifies a circle with a radius of *inradius* + *0.5×(outradius - inradius)*, or a circle halfway between the inner circle and the outer circle. Likewise, the value 0.25 specifies a circle one-fourth of the way from the inner circle to the outer circle.

To specify the line types for the circles, use the LSTARCIRCLES= option. To specify colors for the circles, use the CSTARCIRCLES= option.

Alias STARCIRCLE=

Example "Example 2: Overlaying Radar Charts" on page 1401

STARFILL= lists of SOLID | EMPTY, one for each star

determines whether the stars in the radar chart are empty or filled with a solid color. Valid values are EMPTY (the default) and SOLID. If there are multiple stars in the chart, specify, in parentheses, a separate value for each star.

If the STARFILL=(SOLID) option and the CSTARFILL= option are not specified, then each star is filled with the colors specified on the CSTARS= option. If the CSTARS= option is not specified, then the fill colors are determined by the current style or, if the NOGSTYLE option is specified, by the device color list.

If STARFILL= is not set or is set to EMPTY, then CSTARFILL= is ignored.

STARINRADIUS=*percent*

specifies the inner radius of stars as a percent from 0 to 100. The inner radius of a star is the distance from the center of the star to the circle that represents the lower limit of the standardized vertex variables. The lower limit can correspond to the minimum value, a multiple of standard deviations below the mean, or a lower specification limit. The value must be less than the value that is specified with the STAROUTRADIUS= option. The default value is one-third of the outer radius.

Restriction Not supported by ActiveX

See STAROUTRADIUS= on page 1396

Example "Example 7: Modifying Chart Axes" on page 1408

STARLEGEND=CLOCK | CLOCK0 | NUMBER | DEGREES | NONE
specifies the style of the legend used to identify the vertices of stars that are produced for a radar chart.

CLOCK
identifies the vertex variables by their positions on the clock (starting with 12:00).

CLOCK0
identifies the vertex variables by their positions on the clock (starting with 0:00 corresponding to 12:00).

NUMBER
identifies the vertex variables by numbers. The number 1 corresponds to 12 o'clock. Legend entries are assigned in clockwise order.

DEGREES
identifies the vertex variables by angles in degrees. Zero degrees corresponds to 12 o'clock.

NONE
suppresses the legend. This is the default.

Example "Example 3: Tiling Radar Charts" on page 1402

STARLEGENDLAB="*legend-label*"
specifies the label displayed to the left of the legend for stars requested with the STARLEGEND= option. The label can be up to 16 characters and must be enclosed in quotation marks. The default label is *Vertices:*.

Example "Example 3: Tiling Radar Charts" on page 1402

STAROUTRADIUS=*value*
specifies the outer radius of stars as a percent up to 100. The value must be greater than the value of the STARINRADIUS= option. The inner radius of a star is the distance from the center of the star to the circle that represents the lower limit of the standardized vertex variables. The lower limit can correspond to the minimum value, a multiple of standard deviations below the mean, or a lower specification limit.

Restriction Not supported by ActiveX

See STARINRADIUS= on page 1395

STARSTART=*value*
specifies the vertex angle for the first variable that is specified in the CHART statement. Vertex angles for the remaining variables are equally spaced clockwise and assigned in the order listed. You can specify the value in the following ways:

- **Clock position**: If you specify the value as a time literal (between '0:00'T and '12:00'T), the corresponding clock position is used for the first vertex variable. For example, '12:00'T indicates the 12 o'clock position, '03:00'T the 3 o'clock position (90 degrees), and '09:00'T the 9 o'clock position (270 degrees).
- **Degrees**: To specify a value in degrees, you must specify a negative number. (This is to distinguish degrees from clock values, which are stored internally as positive numbers.) If you specify a negative number, the absolute value is used for the first vertex angle in degrees. Here, 0 degrees corresponds to 12:00, −90 degrees to 3:00, and −270 degrees to 9:00. Always specify the value in degrees as a negative number.

The default value is zero, so the first vertex variable is positioned at 12:00.

STARTYPE=CORONA | POLYGON | RADIAL | SPOKE | WEDGE
specifies the style of the stars that are produced for a radar chart. The following keywords are available:

CORONA
polygon with star-vertices emanating from the inner circle

POLYGON
closed polygon

RADIAL
rays emanating from the center

SPOKE
rays emanating from the inner circle

WEDGE
closed polygon with rays from the center to the full spoke length.

Default WEDGE

SUMVAR=*summary-variable*
specifies a numeric variable to be used to construct weighted radar charts. The values of *summary-variable* can be positive, negative, zero, or missing. If SUMVAR= is not specified, the weights applied to the chart variable are assumed to be one. The chart is not affected by applying a format to *numeric-variable*.

See "Calculating Weighted Statistics" on page 1399

Example "Example 7: Modifying Chart Axes" on page 1408

TILELEGEND=(*variable*)
specifies a variable used to add a legend for CTILES= colors. The variable can have a formatted length less than or equal to 32. If a format is associated with the variable, then the formatted value is displayed. The TILELEGEND= option must be used in conjunction with the CTILES= option for filling the tiles in a chart. If CTILES= is specified and TILELEGEND= is not specified, a color legend is not displayed.

The values of the CTILES= and TILELEGEND= variables should be consistent for all observations with the same level of the classification variables. The value of the TILELEGEND= variable is used to identify the corresponding color value of the CTILES= variable in the legend.

Restriction Not supported by ActiveX

TILELEGLABEL="*label*"
specifies a label displayed to the left of the legend that is created when you specify a TILELEGEND= variable. The label can be up to 16 characters and must be enclosed in quotation marks. The default label is *Tiles:*.

Restriction Not supported by ActiveX

URL=*character-variable*
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only.

Not supported by the GIF, PNG, and ACTXIMG devices

See "Overview of Enhancing Web Presentations" on page 250

"Example: GIF Output with Drill-Down Links" on page 187

WAXIS=
See WFRAME=.

WFRAME=*n*
specifies the width in pixels for the frame lines.

Alias WAXIS=

Default 1

Restriction Not supported by ActiveX

WINDROSE
The input data set can contain data for four, eight, or 16 compass points. The data must be sorted by direction.

See "About Wind Rose Charts" on page 1377

Example "Example 8: Creating a Wind Rose Chart" on page 1411

WINDROSECIRCLES=*n*
specifies the number of reference circles.

WSPOKES=*line-width*
specifies the width in pixels of the spokes in a radar chart.

Alias WSPOKE=

Default 1

WSTARCIRCLES=(*line-widths*)
specifies the width in pixels of the outline of circles requested by the STARCIRCLES= and WINDROSECIRCLES= options.

Alias WSTARCIRCLE=

Default 1

Restriction This option is ignored unless the STARCIRCLES= or WINDROSECIRCLES= option is specified.

WSTARS=*line-width* | (*line-widths*)
specifies the width in pixels of the outline of stars that are produced for a radar chart.

Alias WSTAR=

Default 1

Example "Example 5: Modifying the Appearance of Radar Charts" on page 1405

Usage: GRADAR Procedure

Calculating Weighted Statistics

By default, each observation is counted only once in the calculation of the chart statistics. To calculate weighted statistics in which an observation can be counted more than once, use the FREQ= option. This option identifies a variable whose values are used as a multiplier for the observation in the calculation of the statistic. If the value of the FREQ= variable is missing, 0, or negative, the observation is excluded from the calculation.

If you use the SUMVAR= option, then for each observation, the value of the SUMVAR= variable is multiplied by the value of the FREQ= variable in calculating the chart statistic.

For example, to use a variable called COUNT to produce weighted statistics, assign FREQ=COUNT. If you also assign the variable HEIGHT to the SUMVAR= option, then the following table shows how the values of COUNT and HEIGHT would affect the statistic calculation:

Value of COUNT	Value of HEIGHT	Number of Times the Observation is Used	Value Used for HEIGHT
1	55	1	55
5	65	5	325
.	63	0	-
-3	60	0	-

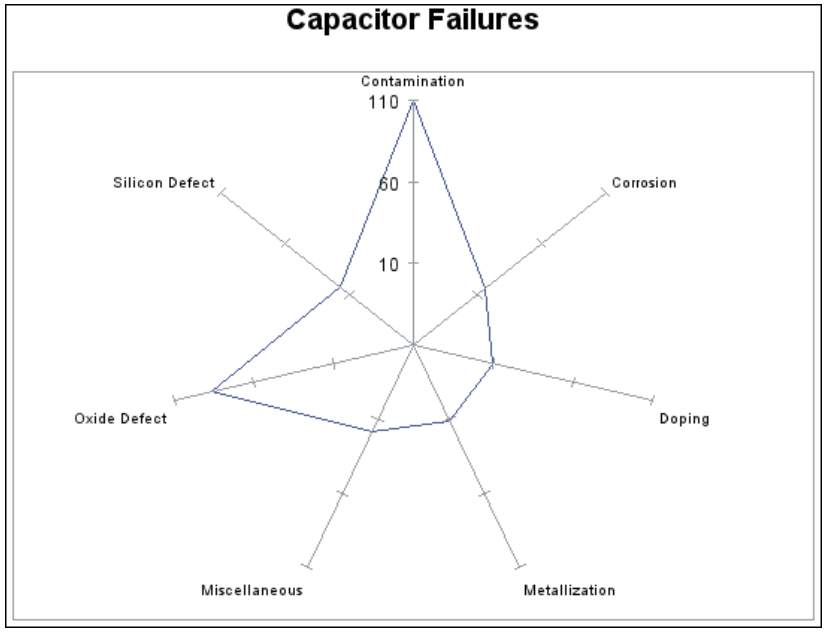
Examples: GRADAR Procedure

Example 1: Producing a Basic Radar Chart

Features:	CHART statement options FREQ=
Sample library member:	GRRBASIC
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

During the manufacture of a metal-oxide semiconductor (MOS) capacitor, different cleaning processes were used by two manufacturing systems that were operating in parallel. Process A used a standard cleaning solution. Process B used a different cleaning mixture that contained less particulate matter. For five consecutive days the causes of failure with each process were observed, recorded, and saved in the SAS data set called FAILURE.

In a radar chart, the vertices are determined by the levels of a single variable, which is specified in the CHART statement. In this example, the variable CAUSE is specified as the chart variable. The spokes in the chart start at the twelve o'clock position and go in a clockwise order. The output shows that Contamination and Oxide Defects are the most frequently occurring problems. The FREQ= option specifies variable COUNT to score vertex lengths. Thus, the values of COUNT weigh the contribution of each observation in the computation of the chart statistic.




```
goptions reset=all border;
title "Capacitor Failures";

proc gradar data=sashelp.failure;
    chart cause / freq=count;
run;
quit;
```

Example 2: Overlaying Radar Charts

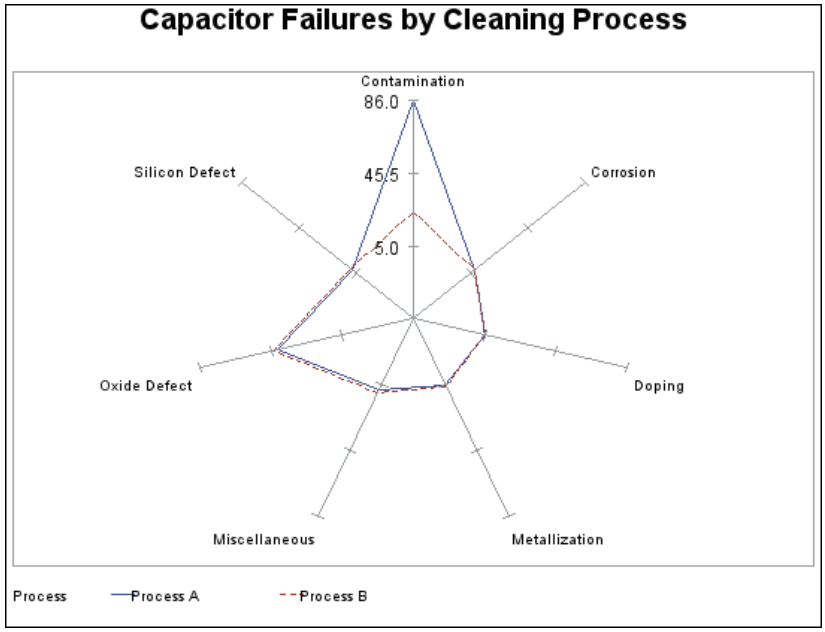
Features: CHART statement options
- FREQ=
- OVERLAYVAR=

Sample library member: GRROVER

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

The most typical way that radar charts are displayed is to overlay the charts on top of each other. To produce an overlay chart, use the OVERLAY= option in the CHART statement. On the OVERLAY= option, specify the classification variable whose values determine the charts to be overlaid. This example shows two blocks of code. For overlay charts with multiple stars, the lines for the stars are rotated through different line styles and colors so that the different stars can be easily seen.

In this example, the OVERLAY= option specifies variable DAY as the overlay variable.




```
goptions reset=all border;
title "Capacitor Failures by Cleaning Process";

proc gradar data=sashelp.failure;
    chart cause / freq=count
    overlayvar=process;
run;
quit;
```

Example 3: Tiling Radar Charts

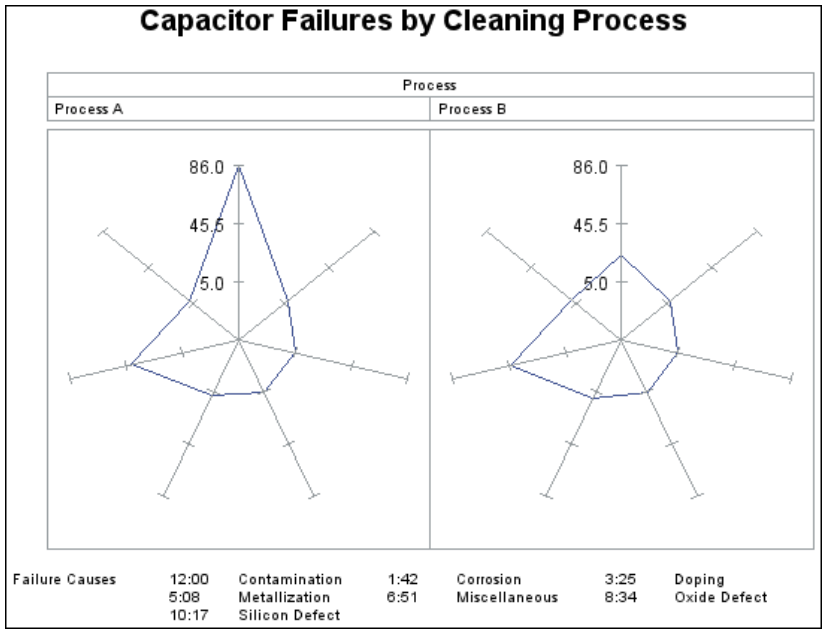
Features: CHART statement options
- ACROSSVAR=
- FREQ=
- STARLEGEND=
- STARLEGENDLAB=

Sample library member: GRRTILE

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

As an alternative to overlaying multiple radar charts (see "Example 2: Overlaying Radar Charts" on page 1401), you can *tile* charts horizontally, vertically, or in both directions (see "Example 4: Using Multiple Classification Variables in Radar Charts" on page 1403) using the ACROSSVAR= or DOWNVAR= options. Each cell in the output corresponds to a level of the classification variable. By default, the cells are arranged in alphabetical order of the values of the variable from top to bottom. The *key cell* is the left cell (corresponding to *PROCESS* = *Process A* in this example).

The output in this example shows that the main difference in the frequencies for Process A and Process B is a drop in contamination using Process B.



ACROSSVAR= specifies variable PROCESS as the categorical variable whose values determine the number of charts that are tiled. STARLEGEND=CLOCK generates a legend that identifies spoke positions. Value CLOCK determines that the positions are identified using a clock metaphor. STARLEGENDLAB= specifies the category-legend label *Failure Causes*.

```
goptions reset=all border;
title "Capacitor Failures by Cleaning Process";

proc gradar data=sashelp.failure;
    chart cause / acrossvar=process
    freq=count
    starlegend=clock
    starlegendlab="Failure Causes";
run;
quit;
```

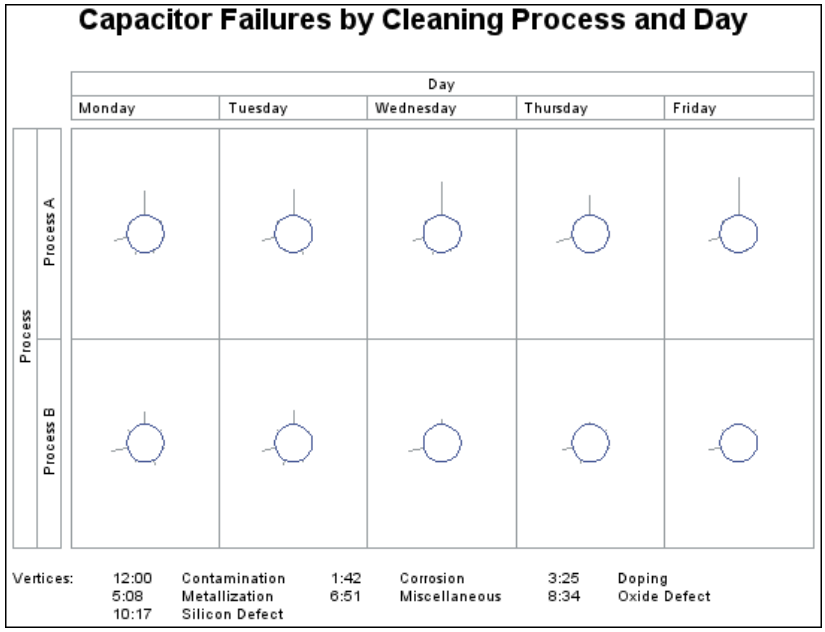
Example 4: Using Multiple Classification Variables in Radar Charts

Features: CHART statement options
- ACROSSVAR=
- DOWNVAR=
- FREQ=
- STARTYPE=
- NCOLS=
- NROWS=
- STARLEGEND=

Sample library member: GRRTWOWY

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

You can study the effects of two classifications simultaneously with a two-way comparative radar chart. This arrangement provides the opportunity to discover both one-way marginal effects and interaction effects. To produce the chart, use both the ACROSSVAR= and DOWNVAR= options.



The ACROSSVAR= option specifies variable DAY as the variable whose values determine the rows in the chart matrix. The DOWNVAR= option specifies variable PROCESS as the variable whose values determine the columns in the chart matrix. The STARTYPE= option determines that the stars are displayed with rays emanating from the inner circle. The NROWS= and NCOLS= options specify the number of rows and columns in the chart. The STARLEGEND=CLOCK option generates a legend that identifies spoke positions. Value CLOCK determines that the positions are identified using a clock metaphor.

```
goptions reset=all border;
title "Capacitor Failures by Cleaning Process and Day";

proc gradar data=sashelp.failure;
    chart cause / acrossvar=day
    downvar=process
    freq=count
    startype=spoke
    nrows=2 ncols=5
    starlegend=clock;
run;
quit;
```

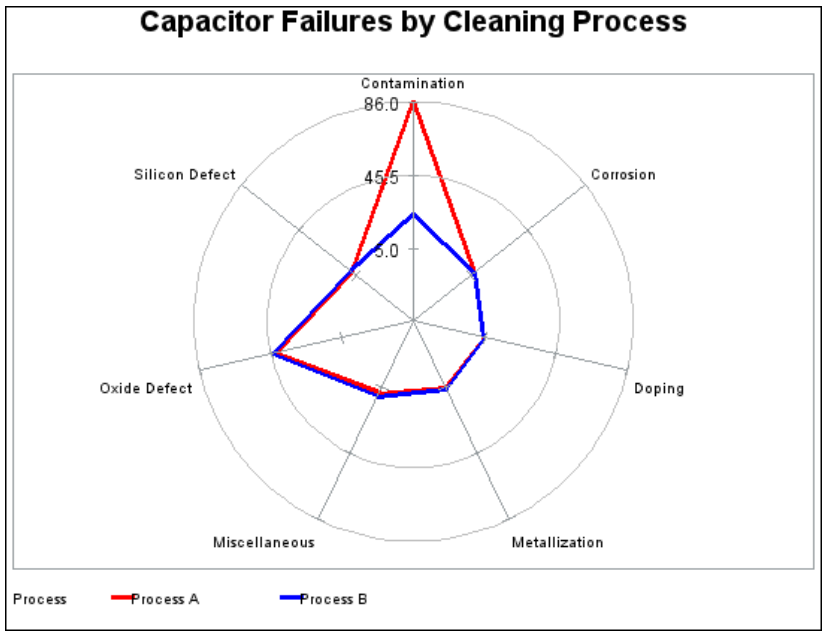
Example 5: Modifying the Appearance of Radar Charts

Features: CHART statement options
- CSTARS=
- FREQ=
- LSTARS=
- OVERLAYVAR=
- STARCIRCLES=
- WSTARS=

Sample library member: GRRAPEAR

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

For overlay charts with multiple stars, the lines for the stars are rotated through different line styles and colors so that the different stars can be easily seen. Rather than relying on the default rotation patterns, you can control the line colors, widths, and styles with the CSTARS=, LSTARS=, and WSTARS= options.



The CSTARS= option specifies a different color for each of the star outlines in the chart. The WSTARS= option specifies the width of the line for each star outline. The LSTARS= option specifies a solid line as the line style for each star outline. The STARCIRCLES= option determines that two reference circles are superimposed on the star charts. The value 1.0 determines that a circle with a radius equal to the spoke length is displayed. The value 0.5 determines that a circle is displayed half way between the outer circle and the smallest circle (value 0.0) that could be drawn for the chart. The value 0.0 would display a circle at the minimum data value, which

does not mean that it is actually 0. For example, for data values of 4, 8, 10, and 12, STARCIRCLES=(0.0 1.0) would draw circles at 4 and 12.

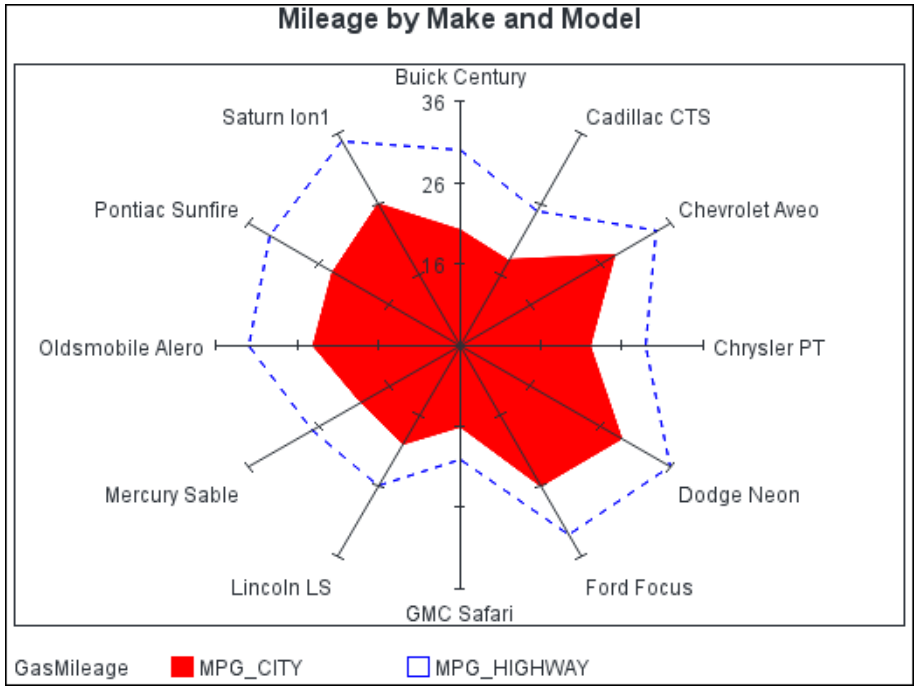
```
goptions reset=all border;
title "Capacitor Failures by Cleaning Process";

proc gradar data=sashelp.failure;
    chart cause /  freq=count
    overlayvar=process
    cstars=(red, blue)
    wstars=2 2
    lstars=1 1
    starcircles=(0.5 1.0)
    cstarcircles=ltgray;
run;
quit;
```

Example 6: Specifying Fill for Multiple Stars

Features: CHART statement options
- FREQ=
- OVERLAY=
- CSTARFILL=
- STARFILL=

Sample library member: GRRFILLD

For overlay charts with multiple stars, you can use fill to represent each unique value of the classification variable that is specified in the OVERLAY= option. Use the STARFILL= and CSTARFILL= options to specify the fill type and color for each unique classification value. This example shows you how to specify the fill for a star chart of the make and model of sedans that are made in the USA with an overlay of city and highway mileage for each.



Get the data for sedans made in the USA from data set Sashelp.Cars.

```
data cars;
  set sashelp.cars;
  where type='Sedan' and origin='USA';
run;
```

Sort data set Cars by variable Make.

```
proc sort;
   by make;
run;
```

Add variable Make_Model to data set Cars. This variable is an amalgamation of variables Make and Model by Make.

```
data cars;
   set cars;
   by make;
   length make_model $25;
   make_model=strip(make)||' '||strip(scan(model,1,' '));
   if first.make then output;
run;
```

Add variables MPG and GasMileage to data set Cars. Store the value of Mpg_Highway and Mpg_City in variable MPG, and store the mileage type (highway or city) in variable GasMileage. Drop the unused variables from data set Cars.

```
data cars;
   set cars;
   mpg=mpg_highway; GasMileage='MPG_HIGHWAY'; output;
   mpg=mpg_city; GasMileage='MPG_CITY'; output;
   keep make_model mpg GasMileage;
run;
```

Set the graphics environment.

```
goptions border htext=10pt;
```

Create the radar chart. Specify variable GasMileage as the classification variable in the OVERLAY= option. Specify the fill type for each of the stars in option STARFILL=. In this example, the first star has a solid fill, and the second star is empty. Specify the colors for the two stars in the CSTARFILL= option. In this example, the fill for the first star is red, and the outline for the second star is blue.

```
title h=12pt 'Mileage by Make and Model';
proc gradar data=cars;
  chart make_model / freq=mpg
                     overlay=GasMileage
                     starfill=(solid empty)
                     cstarfill=(red blue)
                     name="Radar";
run;
quit;
```

Reset the graphics environment.

```
goptions reset=all;
```

Example 7: Modifying Chart Axes

Features:

AXIS statement options
- ORDER=
- VALUE=

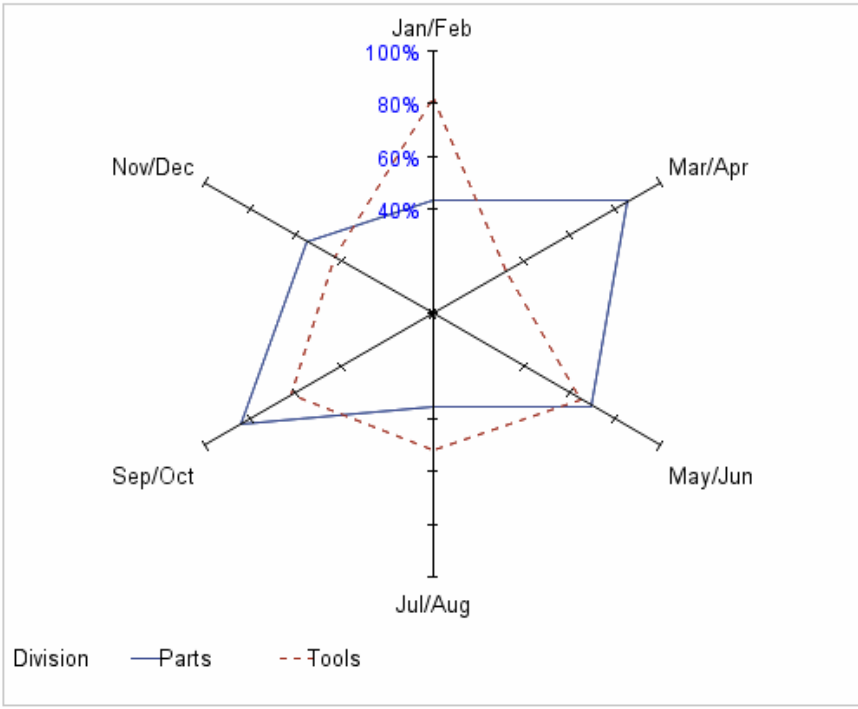
CHART statement options
- HEIGHT=
- NOFRAME
- STARAXIS=
- STARINRADIUS=
- SUMVAR=

FORMAT procedure

Sample library member: GRRAXIS

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

You can use AXIS statements to modify the tick marks that appear on the chart spokes and the order in which the tick mark labels appear. This example uses AXIS statements, in addition to formats, to modify the tick marks, tick mark labels, and axis labels on the radar chart spokes.



Program

```
libname library "library-specification";
proc format library=library;
value mnth
   1="Jan/Feb" 2="Mar/Apr" 3="May/Jun"
   4="Jul/Aug" 5="Sep/Oct" 6="Nov/Dec";
run;

data goals;
   input month Division $ value @@;
   format month mnth.;
   format value percentn7.0;
datalines;
1 Parts .43 1 Tools .82
2 Parts .86 2 Tools .32
3 Parts .70 3 Tools .65
4 Parts .35 4 Tools .52
5 Parts .84 5 Tools .62
6 Parts .55 6 Tools .43
;
run;

goptions reset=all border hsize=5.15in vsize=4.2in;

axis1 order=(0 to .4 by .4, .6 to 1 by .2) value=(height=3pct c=blue
tick=1 "");
axis2 order=(0 to .4 by .4, .6 to 1 by .2) value=none;

proc gradar data=goals;
   chart month / sumvar=value
      staraxis=(axis1 axis2 axis2 axis2 axis2 axis2)
```

```
      noframe height=3.25
      starinradius=0
      overlayvar=division;
run;
quit;
```

Program Description

Create a user-defined format, MNTH, for the axis labels. Replace *library-specification* with the path to your library.

```
libname library "library-specification";
proc format library=library;
value mnth
   1="Jan/Feb" 2="Mar/Apr" 3="May/Jun"
   4="Jul/Aug" 5="Sep/Oct" 6="Nov/Dec";
run;
```

Create the input data set WORK.GOALS. The user-defined format is applied to the variable MONTH, and the SAS format PERCENTN7.0 is applied to the variable VALUE.

```
data goals;
   input month Division $ value @@;
   format month mnth.;
   format value percentn7.0;
datalines;
1 Parts .43 1 Tools .82
2 Parts .86 2 Tools .32
3 Parts .70 3 Tools .65
4 Parts .35 4 Tools .52
5 Parts .84 5 Tools .62
6 Parts .55 6 Tools .43
;
run;
```

Set the graphics environment.

```
goptions reset=all border hsize=5.15in vsize=4.2in;
```

Define tick marks for the chart spokes. The AXIS1 statement controls the tick marks for the first (12 o'clock) axis. The ORDER= option specifies the tick mark interval and the actual values that correspond to each tick mark. The format PERCENTn7.0 that was applied to the variable VALUE in the DATA step controls how the values are displayed for the tick mark labels. The VALUE= option specifies the size and color of the tick mark labels. The AXIS2 statement controls all axes except the first. For those axes, the VALUE= option specifies that no tick mark labels are displayed.

```
axis1 order=(0 to .4 by .4, .6 to 1 by .2) value=(height=3pct c=blue
tick=1 "");
axis2 order=(0 to .4 by .4, .6 to 1 by .2) value=none;
```

Create the radar chart. The STARAXIS= option tells the procedure to apply the AXIS1 statement to the first axis and the AXIS2 statement to all other axes in the chart.

```
proc gradar data=goals;
```

```
   chart month / sumvar=value
      staraxis=(axis1 axis2 axis2 axis2 axis2 axis2)
      noframe height=3.25
      starinradius=0
      overlayvar=division;
run;
quit;
```

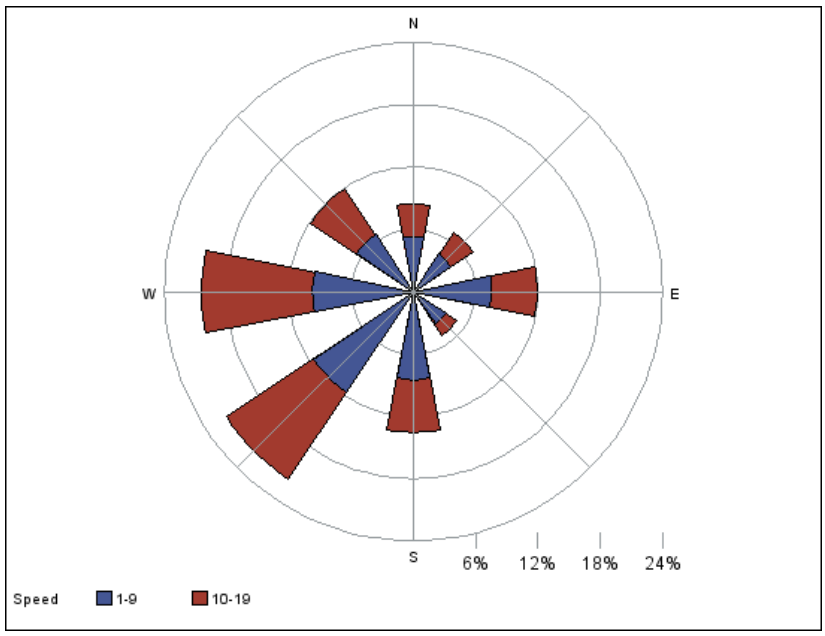
Example 8: Creating a Wind Rose Chart

Features: CHART statement options
- NOFRAME
- SPEED=
- SUMVAR=
- WINDROSE

Sample library member: GRRWNDRS

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

The wind rose chart displays wind direction and speed over a period of time at a specific location. The segments represent the percentage of time that the wind blew from each direction.




```
goptions reset=all border;

data wind;
   input Direction $ Speed $ Percent @@;
datalines;
N    1-9  1.7 N  10-19 1.0 NE   1-9  1.4
```

```
NE  10-19  .8 E    1-9 2.4 E   10-19 1.4
SE   1-9  1.2 SE 10-19  .4 S    1-9  2.7
S   10-19 1.6 SW   1-9 3.7 SW  10-19 3.2
W    1-9  3.1 W  10-19 3.4 NW   1-9  2.1
NW  10-19 1.7
run;

proc gradar data=wind;
    chart direction / sumvar=percent
    windrose
    speed=speed
    noframe;
run;
quit;
```

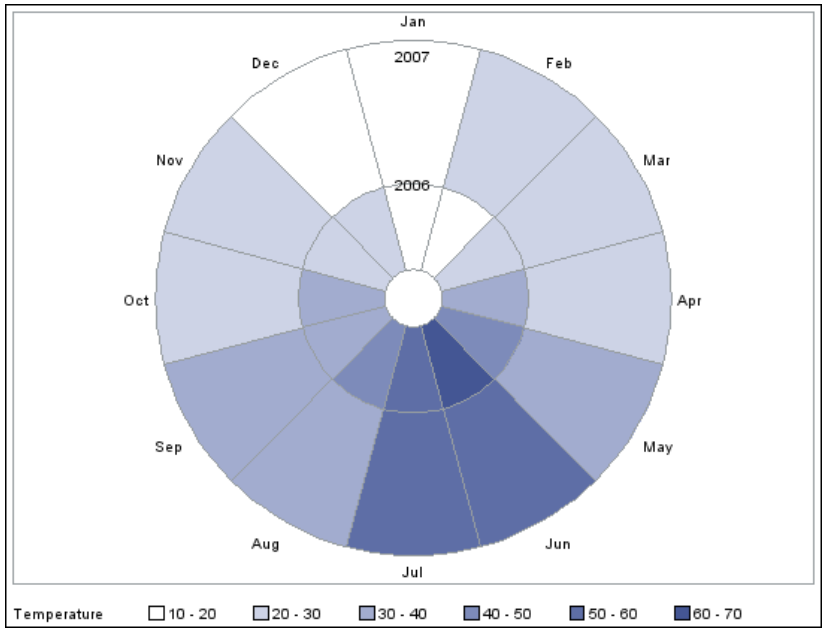
Example 9: Creating a Calendar Chart

Features: CHART statement options
- FREQ=
- CALENDAR
- CSTARS=
- OVERLAYVAR=

Sample library member: GRRCALEN

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

The CALENDAR option produces a chart with 12 equal-sized segments, one for each month of the year. The color shading of each segment represents the magnitude of the frequency variable. Use the OVERLAY variable to subdivide each segment (for example, by year).




```
goptions reset=all border;

data climate;
   input Year Month $ Temperature @@;
datalines;
2006 Jan 16 2006 Feb 19 2006 Mar 22 2006 Apr 33
2006 May 41 2006 Jun 60 2006 Jul 55 2006 Aug 41
2006 Sep 38 2006 Oct 30 2006 Nov 27 2006 Dec 20
2007 Jan 18 2007 Feb 23 2007 Mar 20 2007 Apr 27
2007 May 33 2007 Jun 52 2007 Jul 55 2007 Aug 38
2007 Sep 38 2007 Oct 27 2007 Nov 26 2007 Dec 19
run;

proc gradar data=climate;
    chart month / freq=temperature
    calendar
    overlayvar=year;
    run;
quit;
```

Chapter 44

GREPLAY Procedure

Overview: GREPLAY Procedure

What Does the GREPLAY Procedure Do?

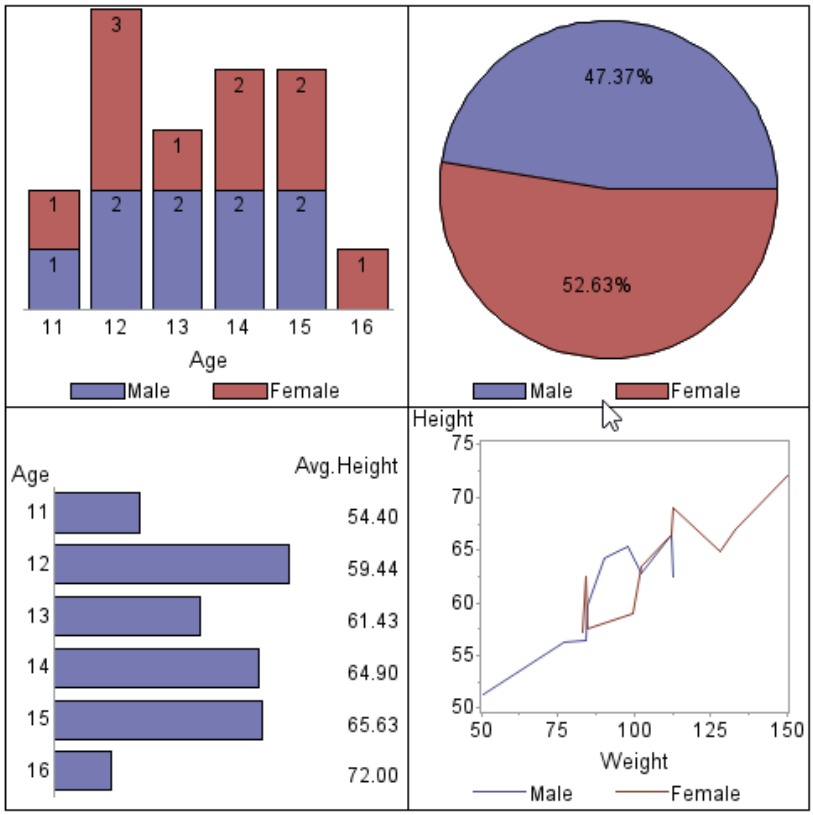
The GREPLAY procedure displays and manages graphics output that is stored in SAS catalogs. The GREPLAY procedure also creates templates and color maps that you can use when you replay your graphics output. The GREPLAY procedure operates in line mode, batch mode, and in the SAS windowing environments.

With the GREPLAY procedure, you can perform any of the following actions:

- Lay out multiple graphs on one page; this output can be used to create dashboards.
- Create an inset graph.
- Select one or more catalog entries from the same catalog for replay, and direct this output to your display or other devices such as plotters and printers.
- Use, create, or modify templates. Use templates to describe positioning on a single display, for graphics output stored in one or more graph catalog entries.
- Create new graphics output by replaying one or more catalog entries into panels within a template.
- Use, create, or modify color maps. Use color maps to map current colors to different colors.
- List templates in SASHELP.TEMPLT.
- Manage GRSEG, TEMPLATE, and CMAP entries in SAS catalogs by doing the following:
  - Rearranging or creating logical groupings of catalog entries that contain graphics output.
  - Renaming, deleting, or copying catalog entries that contain graphics output, templates, and color maps.

Figure 44.170 on page 1417 shows four catalog entries that were replayed into a template and displayed as a single graph.

Figure 44.1 *Graphics Output in a Template*



Concepts: GREPLAY Procedure

Catalog Entries

The GREPLAY procedure can perform actions on three types of catalog entries:

GRSEG entries
: store output from SAS/GRAPH procedures. The GREPLAY procedure uses two types of graphics catalogs: the input-catalog and the output-catalog. The input-catalog is the catalog that contains the graphics output that you want to replay. The output-catalog is the catalog in which graphics output that is produced by the template facility is stored. Both of these catalogs are GSEG catalogs. The same GSEG catalog can be used as the input-catalog and the output-catalog.

TEMPLATE entries
: store templates created with the GREPLAY procedure. The catalog in which template entries are stored is referred to as the template catalog. SAS provides sample templates in SASHELP.TEMPLT. TEMPLATE entries can also be stored in GSEG catalogs.

  You can use templates directly from SASHELP.TEMPLT to replay your graphics output, or you can copy these templates to a different catalog and edit the copied entries. Graphics output replayed using a template creates a new GRSEG entry.

CMAP entries
: store color maps created with the GREPLAY procedure. The catalog in which color map entries are stored is referred to as the color map catalog. CMAP entries can be stored in GSEG catalogs. They can also be stored in other catalogs. You can copy, edit, or use these color maps to replay your graphics output. Graphics output replayed using a color map does not create a GRSEG entry.

You can store all of the previous entry types in a single SAS catalog, or you can store them in separate catalogs and use a different catalog for each type of entry. A single SAS catalog can contain graphics output, color maps, and templates.

Because the GREPLAY procedure operates on catalog entries, you must assign at least one catalog before you perform any tasks. The GREPLAY procedure has several ways to assign catalogs shown in Table 44.92 on page 1418.

Table 44.1 *Assigning Catalogs*

Catalog	How to Assign
input	IGOUT= option in the PROC GREPLAY statement IGOUT statement IGOUT field in the PROC GREPLAY window
output	GOUT= option in the PROC GREPLAY statement GOUT statement GOUT field in the PROC GREPLAY window
template	TC= option in the PROC GREPLAY statement TC statement TC field in the PROC GREPLAY window
color map	CC= option in the PROC GREPLAY statement CC statement CC field in the PROC GREPLAY window

Note: Image entries can exist in catalogs, but are not recognized by the GREPLAY procedure.

Displaying the List of Templates Provided by SAS/GRAPH

To write the list of templates stored in SASHELP.TEMPLT to the SAS log, submit the following code:

```
proc greplay nofs
tc=sashelp.templt;
   list tc;
run;
```

Duplicate Entry Names

The GREPLAY procedure uses the following naming conventions to prevent duplication of names, or overwriting entries:

- For entry names that are less than eight characters long, the procedure adds a numeric suffix to the entry's name. The total number of characters is limited to eight.
- For entry names that are eight characters long or longer, the procedure drops the number of characters that are needed to add a numeric suffix while maintaining the eight-character limit. For example, if you copy an entry TITLEONE to a catalog that already contains an entry with that name, the procedure assigns the name TITLEON1 to the copied entry.

Note: The GREPLAY procedure uses the same naming conventions for entries created by the template facility.

Note: See also "About Filename Indexing" on page 137.

Ways to Use the GREPLAY Procedure

You can view, replay, or manage catalog entries in two ways:

- by submitting code-based GREPLAY procedure statements. The GREPLAY procedure automatically uses code-based statements if you are running in batch mode or in line mode in a non-windowing environment. See "Using Code-based Statements to Run the GREPLAY Procedure" on page 1453.
- by browsing or editing the fields in the GREPLAY procedure windows (if you are running SAS in a windowing environment). For more information, see "Using the GREPLAY Procedure Windows" on page 1448.

If you are in the SAS windowing environment, you can toggle between the windows and code-based statements while you run the GREPLAY procedure.

For more information, see "FS Statement" on page 1433 and "NOFS" on page 1426.

Sizing and Naming Your Graphs for Replay (Best Practice)

To replay your graphics output using the GREPLAY procedure, it is recommended that you do the following:

- Select or create a template to replay your graphs. Determine the size of each panel contained in the template. Define the size of each graph to correspond to the size of a panel contained in the template. Size each graph with GOPTIONS such as the XPIXELS= and YPIXELS= options or the HSIZE= and VSIZE= options. If the graphs that you are replaying are too large for the panels in the template, SAS/GRAPH attempts to resize the images.
- Ensure that the GRSEG entry names that you want to replay match the names in your GREPLAY procedure statements. If you run a procedure multiple times without updating your GREPLAY statements, the original output is replayed, not the most current output. See "Duplicate Entry Names" on page 1419 and "Example 3: Replaying Graphs into a Template" on page 1464.

Considerations for Replaying a Graph

Replaying a Graph with a Border

In SAS 9.1.3, when you request that a border be drawn around your graph, the border is drawn around the graph inside the graph GRSEG. In SAS 9.2 and later versions, when you request that a border be drawn around your graph, if the ODS output destination that you are using supports drawing a border around the graph, the border is drawn outside of the graph GRSEG by ODS. Otherwise, the border is drawn around the graph inside the graph GRSEG as is done in SAS 9.1.3.

Note: When the BORDER option is used with the GSLIDE procedure, a border is drawn around the slide inside the slide GRSEG regardless of whether the ODS destination supports drawing a border around the graph.

The ODS HTML destination supports drawing a border around a graph when a border is requested, while the ODS LISTING, ODS PDF, ODS RTF, and ODS PRINTER destinations do not. When you replay a graph with a border, if you replay the graph to the same ODS destination that was used to generate the original graph, the graph should replay without any problems. However, in the following replay scenarios, you might encounter unexpected results with respect to the border in the replayed graph:

- When you replay a graph with a border that was generated with the ODS LISTING, PDF, RTF, or PRINTER destination to the ODS HTML destination, the replayed graph might have a double border drawn around it. The double border occurs when the original graph GRSEG already contains a border and the ODS HTML destination draws a second border around it.
- When you replay a graph with a border that was generated with the ODS HTML destination to the ODS LISTING, PDF, RTF, or PRINTER destination, the replayed graph does not have a border. The border is missing because the border in the original graph is not contained in the graph GRSEG.

In the case of a double border, to remove the second border, you can specify the NOBORDER graphics option when you replay the graph. In the case of the missing border, you can create a GREPLAY template that draws a border around the output panel and then replay the GRSEG using that template. In your template code, specify the COLOR=*color* option in the GREPLAY procedure TDEF statement. The COLOR= option draws a border around the graph in the new GRSEG. See "COLOR=*border-color*" on page 1442 and "Example 4: Replaying a Graph to Multiple Destinations" on page 1468.

Replaying a Graph with Titles and Footnotes

If your graph contains titles, footnotes, or both, and you use the NOGTITLE and NOGFOOTNOTE options on your ODS statement, the titles and footnotes are rendered outside of the graph. They are not included in the GRSEG in that case. See "Controlling Titles and Footnotes" on page 115. As a result, when you replay the graphs using the GREPLAY REPLAY or TREPLAY statement, the titles and footnotes do not appear in the replay output. If you include titles and footnotes with your graphs and you intend to replay the graphs using GREPLAY, make sure that you do not use the NOGTITLE and NOGFOOTNOTE options in your ODS statement.

Syntax: GREPLAY Procedure

Restriction: Not supported by Java or ActiveX

Requirement: Use the NOFS option in the PROC GREPLAY statement when running in a non-windowing environment, batch mode, or in line mode in a windowing environment. At least one statement is required.

Supports: RUN-group processing

Note: Write access to a catalog is needed to modify, add, or delete catalog entries. Only GRSEG entry types can be replayed with the GREPLAY procedure.

Tip: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, `ODS PDF; procedure-code; ODS PDF CLOSE;`). See Chapter 7, "Using Run-Group Processing," on page 75 for more information.

```
PROC GREPLAY
   <BYLINE>
   <CC=color-map-catalog>
   <CMAP=color-map-entry-type>
   <FS>
   <GOUT=<libref.>output-catalog>
   <IGOUT=<libref.>input-catalog>
   <IMAGEMAP=output-data-set>
   <NOBYLINE>
   <NOFS>
   <PRESENTATION>
   <TC=template-catalog>
   <TEMPLATE=template-entry>;
      ? CC | CMAP | DEVICE | GOUT | IGOUT | TC | TEMPLATE;
      BYLINE;
      CC color-map-catalog;
      CCOPY <color-map-catalog.>color-map-entry<.CMAP>;
      CDEF color-map-entry
         color-number/from-color:to-color
         <color-number/from-color:to-color>
         <...>
         <DES="description">
        ;
      CDELETE color-map-entry(s) | _ALL_;
      CMAP color-map-entry;
      COPY entry-id(s) | _ALL_;
      DELETE entry-id(s) | _ALL_ ;
      DEVICE device-name;
      FS;
      GOUT <libref.>output-catalog;
      GROUP entry-id(s);
      IGOUT <libref.>input-catalog;
      LIST CC | CMAP | IGOUT | TC | TEMPLATE;
      MODIFY modify-pair(s);
      MOVE entry-id-1 AFTER | BEFORE entry-id-2;
      NOBYLINE;
      PREVIEW template-entry(s) | _ALL_ ;
      QUIT;
      REPLAY entry-id(s) | _FIRST_ | _LAST_ | _ALL_ ;
      TC template-catalog;
      TCOPY <template-catalog.>template-entry<.TEMPLATE>;
      TDEF template-entry
         panel-number/<panel-options>
         <panel-number/<panel-options>>
         <...>
```

<DES="*description*">
;
TDELETE *template-entry(s)* | _ALL_;
TEMPLATE *template-entry*;
TREPLAY *select-pair(s)*;

PROC GREPLAY Statement

Determines whether the procedure starts in a windowing or non-windowing environment. Defines whether the session is used for catalog management or output presentation.

Syntax

PROC GREPLAY

<BYLINE>
<CC=*color-map-catalog*>
<CMAP=*color-map-entry-type*>
<FS>
<GOUT=<*libref*.>*output-catalog*>
<IGOUT=<*libref*.>*input-catalog*>
<IMAGEMAP=*output-data-set*>
<NOBYLINE>
<NOFS>
<PRESENTATION>
<TC=*template-catalog*>
<TEMPLATE=*template-entry*>;

Summary of Optional Arguments

BY-line options

BYLINE
: specifies that the BY statement information for the SAS catalog entries should be displayed.

NOBYLINE
: specifies that the BY statement information for the SAS catalog entries should be suppressed.

Catalog options

CC=*color-map-catalog*
: specifies the color map catalog where the color map entries are stored.

CMAP=*color–map-entry-type*
: specifies the type of catalog entry to use with the GREPLAY procedure.

GOUT=<*libref*.>*output-catalog*

specifies the graphics output catalog.

IGOUT=<*libref.*>*input-catalog*
specifies the input catalog that stores the graphics output that you want to use with the GREPLAY procedure.

TC=*template-catalog*
specifies the template catalog to use with the GREPLAY procedure, and identifies the template catalog where the template entry is stored to replay your graphics.

TEMPLATE=*template-entry*
identifies the template entry to use with the GREPLAY procedure.

Image map options

IMAGEMAP= *output-data-set*
used with the REPLAY statement to create a temporary SAS data set that is used to generate an image map in an SVG file when replaying output to the LISTING destination.

Window options

FS
specifies that the GREPLAY procedure should use full-screen windows.

NOFS
specifies that the GREPLAY procedure should use line mode.

PRESENTATION
specifies that the GREPLAY procedure should open the PRESENTATION window, and use the catalog specified by the IGOUT= option as the input catalog.

Optional Arguments

BYLINE
specifies that the BY statement information for the SAS catalog entries should be displayed. specifies that the BY statement information for the SAS catalog entries should be displayed.

Default BY statement information is displayed

CC=*color-map-catalog*
specifies the color map catalog where the color map entries are stored. specifies the color map catalog where the color map entries are stored.

Note To replay graphics output using a color map, you must specify a color map catalog with the CC= option and a color map entry with the CMAP= option.

Example “Example 5: Creating a Color Map” on page 1472

CMAP=*color–map-entry-type*
specifies the type of catalog entry to use with the GREPLAY procedure. A color map entry option must have a catalog entry type of CMAP.

Note To replay graphics output using a color map entry, you must specify a color map catalog with the CC= option and a color map entry with the CMAP= option.

Example "Example 5: Creating a Color Map" on page 1472

FS

specifies that the GREPLAY procedure should use full-screen windows.

Default If your device supports windows, the GREPLAY procedure uses windows. If your device does not support windows, the procedure begins execution in line mode, and the FS option has no effect.

GOUT=<*libref.*>*output-catalog*

specifies the graphics output catalog. New GRSEG entries or GRSEG entries from other catalogs can be copied to an output catalog. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary WORK library, and creates the GSEG catalog if it does not exist.

Note The output catalog can be the same catalog specified in the IGOUT= option.

See "Catalog Entries" on page 1417

IGOUT=<*libref.*>*input-catalog*

specifies the input catalog that stores the graphics output that you want to use with the GREPLAY procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary WORK library.

Notes The input catalog can be the same catalog specified in the GOUT= option.

If the specified input catalog cannot be found, the GREPLAY procedure stops processing, and the following error message is written to the SAS log:

```
ERROR: Memname input-catalog is unknown.
```

Example "Example 2: Replaying GSLIDE Procedure Output in a Template" on page 1462

IMAGEMAP= *output-data-set*

This option is not necessary when you are replaying output to the HTML destination. The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

Restriction The IMAGEMAP= option is valid with the REPLAY statement only.

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255

"Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" on page 262

"REPLAY Statement" on page 1439

NOBYLINE

specifies that the BY statement information for the SAS catalog entries should be suppressed.

Default BY statement information is displayed

NOFS
specifies that the GREPLAY procedure should use line mode.

Default If your device does not support windows: NOFS

Example "Example 1: Creating a Template" on page 1459

PRESENTATION
specifies that the GREPLAY procedure should open the PRESENTATION window, and use the catalog specified by the IGOUT= option as the input catalog. The PRESENTATION option is often used in applications to prevent the application users from deleting or reordering catalog entries. You can only replay graphics output from the PRESENTATION window.

Note The PRESENTATION option overrides the NOFS option on full-screen devices.

TC=*template-catalog*
specifies the template catalog to use with the GREPLAY procedure, and identifies the template catalog where the template entry is stored to replay your graphics.

Note To replay graphics output using a template catalog, you must also assign the template entry with the TEMPLATE= option.

Example "Example 1: Creating a Template" on page 1459

TEMPLATE=*template-entry*
identifies the template entry to use with the GREPLAY procedure. The template entry must have a catalog entry type of TEMPLATE.

Note To replay graphics output using a template entry, you must also assign a current template catalog with the TC= option. If the template entry is not in the template catalog, an error message is written to the SAS log.

Example "Example 2: Replaying GSLIDE Procedure Output in a Template" on page 1462

Details

Invoking the GREPLAY Procedure

The mode of operation for the PROC GREPLAY statement depends on both the environment in which the statement is submitted and whether the NOFS option is included.

Table 44.2 Ways of Invoking the GREPLAY Procedure

Environment	Statement	Result
windowing	PROC GREPLAY;	GREPLAY procedure windows
windowing	PROC GREPLAY NOFS;	line mode
nonwindowing	PROC GREPLAY;	line mode

You can toggle back and forth between windows and line mode within a session.

? Statement

Writes the current value of certain PROC GREPLAY options, or of the current device driver to the SAS log. If the value is not assigned, the GREPLAY procedure issues a message to the SAS log.

Syntax

? CC | CMAP | DEVICE | GOUT | IGOUT | TC | TEMPLATE

Required Argument

Specify one of the following:

CC
writes the name of the current color map catalog.

CMAP
writes the name of the current color map.

DEVICE
writes the name of the current device driver.

Alias DEV

GOUT
writes the name of the output catalog.

IGOUT
writes the name of the input catalog.

TC
writes the name of the current template catalog.

TEMPLATE
writes the name of the current template.

BYLINE Statement

Displays BY statement information directly beneath the primary description of the catalog entries when you list the input catalog contents.

Note: BY statement information is displayed by default.

See: "NOBYLINE Statement" on page 1438

Syntax

BYLINE;

CC Statement

Specifies a color map catalog, and enables you to change the color map catalog without exiting the procedure.

Syntax

CC *libref.color-map-catalog*;

Required Argument

color-map catalog
identifies the SAS catalog where color map entries are stored.

CCOPY Statement

Copies a color map from one color map catalog to another color map catalog. Creates a duplicate color map within the color map catalog.

Requirement: Assign a color map catalog before using the CCOPY statement.

Syntax

CCOPY <*libref.*><*colormap-catalog.*>*color-map-entry*<.CMAP>;

Required Argument

<libref.><color-map-catalog.>color-map-entry<.CMAP>
identifies the color map entry to be copied.

color-map-catalog
is the color map catalog that contains the color map to be copied.

color-map-entry
is the name of the color map entry.

.CMAP
is the color map entry type.

Note If a color map entry with the same name exists in the color map catalog, duplicate entry names are resolved as described in "Duplicate Entry Names" on page 1419.

See "CC Statement" on page 1428

Details

To copy a color map from one catalog to another catalog, use the CC statement to identify the target catalog. The following statements copy HP.CMAP from the catalog named ONE.CCAT to the catalog named TARGET.CLRMAP:

```
LIBNAME target "SAS library";
LIBNAME one "SAS library";

proc greplay nofs;
   cc target.clrmap;
   ccopy one.ccat.hp.cmap;
quit;
```

To create a duplicate color map, omit the name of the color map catalog from your CCOPY statement. The following statement creates a duplicate of hp.cmap named hp2.cmap:

```
ccopy hp.cmap;
```

CDEF Statement

Defines or modifies one or more color maps in the color-map catalog.

Requirement: Assign a color-map catalog before using the CDEF statement.

Syntax

CDEF *color-map-entry*
color-number|*from-color*:*to-color*

<*color-number*|*from-color*:*to-color*>
<…>
<DES="*description*">;

Required Argument

color-map-entry
identifies a color map entry. If the color map entry is not in the color map catalog, then the procedure creates a color map entry. If the color map entry exists in the color map catalog, then the GREPLAY procedure modifies or adds to that color map entry.

See CC statement

Example "Example 5: Creating a Color Map" on page 1472

Optional Arguments

color-number|from-color:to-color
specifies a color-map entry definition.

color-number
specifies the number of a color-mapping pair.

from-color:to-color
defines the color mapping for the color-map entry.

from-color
specifies the color to be mapped.

to-color
specifies the new color that replaces *from-color* in the replayed graphics output.

DES="*description*"
specifies a catalog entry description for the color map entry. Maximum length for the description is 256 characters.

Default Color map

CDELETE Statement

Deletes one or more color map entries from the current color map catalog.

Note: The GREPLAY procedure does not prompt you to confirm your request to delete color maps entries.

Syntax

CDELETE *color-map-entry(s)* | _ALL_;

Required Arguments

color-map-entry(s)
identifies one or more color map entries to delete from the color map catalog. You can submit one entry, or a space-delimited list of entries in one delete statement.

ALL
deletes all of the color map entries from the color map catalog.

Alias CDEL

CMAP Statement

Assigns a color map entry to be used when replaying graphics output.

Requirement: Assign a color map catalog before using the CMAP statement.

See: CC
"Example 5: Creating a Color Map" on page 1472.

Syntax

CMAP *color-map-entry*;

Required Argument

color-map-entry
identifies the color map entry, contained in the current color map catalog, to use when replaying your graphics output. If the color map entry is not in the current color map catalog, the GREPLAY procedure issues an error message.

COPY Statement

Copies one or more GRSEG catalog entries from the input catalog to the output catalog.

Requirement: Assign an input catalog and an output catalog before using the COPY statement.

Note: The COPY statement cannot create a duplicate catalog entry that contains graphics output in the same catalog.

See: "GOUT Statement" on page 1433 and "IGOUT Statement" on page 1434

Syntax

COPY *entry-id(s)* | _ALL_;

Required Arguments

entry-id(s)
is the number (in the order in which they were created) or name of a catalog entry or group of entries to be copied from the input catalog to the output catalog. Entries must contain graphics output. Multiple catalog entries can contain both numbers and names.

ALL
copies all graphics output entries in the input catalog to the output catalog.

DELETE Statement

Deletes SAS catalog entries containing graphics output from the current input catalog.

Note: The GREPLAY procedure does not prompt you to confirm your request to delete an entry containing graphics output.

Syntax

DELETE *entry-id(s)* | _ALL_;

Required Arguments

entry-id(s)
is the number (in the order in which they were created) or name of a catalog entry or a group of entries to be deleted from the input catalog. Entries must contain graphics output. Multiple catalog entries can contain both numbers and names.

ALL
deletes all graphics output entries in the input catalog.

Alias DEL

DEVICE Statement

Specifies the device driver.

Requirement: You must specify a device driver that your graphics device can support, and is available to your SAS session.

Syntax

DEVICE *device-name*;

Required Argument

device-name
specifies the device driver to use when you replay graphics output. The device driver that you specify becomes the current device. It is used for subsequent replays and the output of other graphics procedures. This device driver remains in effect until you change the device driver.

Alias DEV

FS Statement

Toggles from line mode to the GREPLAY procedure windows.

Requirement: Device must support windows

See: "NOFS" on page 1426

Syntax

FS;

GOUT Statement

Assigns the SAS output catalog used by the GREPLAY procedure.

Note: You can change the output catalog without exiting the procedure by using the GOUT statement.

Syntax

GOUT <*libref.*>*output-catalog*;

Required Argument

<libref.>output-catalog
identifies the SAS catalog to use as an output catalog.

Default WORK.GSEG

GROUP Statement

Creates groups of entries in the current input catalog.

Syntax

GROUP *entry-id(s)*;

Required Argument

entry-id(s)
is the number (in the order in which they were created) or name of a catalog entry. All entries specified in the GROUP statement are included in one group, and identified with a group header. You can submit one catalog entry or a list of catalog entries with one GROUP statement. A list of catalog entries can contain both catalog entry numbers and catalog entry names.

Restriction The number of entries in a group cannot exceed 400.

Details

Only one group can be created per group statement. The default name for a group header is GROUP. The default description for the group header is REPLAY GROUP. Duplicate entry names are resolved as described in “Duplicate Entry Names” on page 1419.

To change the name or description of a group, use the MODIFY statement. To manage and display groups of entries use the DELETE, COPY, and REPLAY statements.

IGOUT Statement

Assigns the SAS input catalog used by the GREPLAY procedure.

Note: You can change the input catalog without exiting the procedure by using the IGOUT statement.

Syntax

IGOUT <*libref.*>*input-catalog*;

Required Argument

<libref.>input-catalog
identifies the SAS catalog with entries that contain graphics output that you want to replay.

Note If the specified input catalog cannot be found, the GREPLAY procedure stops processing, and the following error message is written to the SAS log:

```
ERROR: Memname input-catalog is unknown.
```

LIST Statement

Lists entries in the input, template, and color map catalogs, as well as the contents of templates and color maps in the SAS log.

Note: Entries are listed in creation date order.

See: "Example 5: Creating a Color Map" on page 1472

Syntax

LIST CC | CMAP | IGOUT | TC | TEMPLATE

Required Argument

Specify one of the following:

CC
lists the color maps that are in the current color map catalog.

CMAP
lists the *From* and *To* values in the current color map.

IGOUT
lists the number, name, and description of the entries in the input catalog that contain graphics output. In addition, the type of graphics output (dependent or independent) is shown.

TC
lists the templates in the current template catalog.

TEMPLATE
lists the panel definition values of the current template.

MODIFY Statement

Changes the name, description, and BY statement information of entries or group headers in the input catalog.

Syntax

MODIFY *entry-id*/*entry-options*
  <*entry-id*/*entry-options*>
  <…>
;

Required Argument

entry-id*/*entry-options
specifies the entry to modify.

entry-id
specifies the number (in the order in which they were created) or name of a catalog entry or a group of entries in the input catalog.

Entries must contain graphics output. When multiple pairs are specified, the entry IDs can be a mix of numbers and names.

entry-options
specifies one or more of the following to replace the original option values for the specified entry:

BYLINE="*character-string*"
specifies a character string that can be used for additional information or for BY statement information.

A character string can be up to 40 characters long, and must be enclosed in quotation marks. BY statement information appears directly beneath the primary description of the catalog entry.

NAME="*new-entry-name*"
specifies a new name for the GRSEG catalog entry.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - Duplicate names are not allowed. If the name specified already exists in the GRSEG catalog, the name modification is rejected.

- Special characters are not allowed. You must use letters, numbers, and underscores.

  Default The original entry name

DES="*description*"
specifies a description of the output.

The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver that you are using). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

  Default The original graph description

MOVE Statement

Rearranges entries in the input catalog by moving entries before or after other entries.

Syntax

MOVE *entry-id-1* AFTER | BEFORE *entry-id-2*;

Required Arguments

entry-id-1
specifies the number (in the order in which they were created) or name of a catalog entry or a group header that is to be moved.

entry-id-2
is the number (in the order in which they were created) or name of a catalog entry or a group header.

AFTER | BEFORE
specifies whether *entry-id-1* should be moved after or before *entry-id-2*.

Details

To move an entire group, use the group name for *entry-id-1*. To move an entry into a group, move the entry after a group header, or before or after an entry in the group. This statement moves the entry CHART3 into the group named NEW_SALES:

```
move chart3 after new_sales;
```

NOBYLINE Statement

Suppresses BY statement information.

See: "BYLINE Statement" on page 1428

Syntax

NOBYLINE;

PREVIEW Statement

Displays the panel outlines for one or more templates using the current device. Use the TC statement to specify the template catalog before using the PREVIEW statement.

Note: When a template is previewed, graphics output is produced, and stored in a catalog named WORK.GTEM. The temporary catalog is deleted when you end your session.

Tip: When previewing templates, press End or Enter, to move to the next template in the list.

Syntax

PREVIEW *template-entry(s)* | _ALL_;

Required Arguments

template-entry(s)
identifies one or more template entries contained in the template catalog. You can preview one entry or a list of entries with one PREVIEW statement.

ALL
previews all templates in the current template catalog.

QUIT Statement

Exits the GREPLAY procedure.

Aliases: END
STOP

Syntax

QUIT;

REPLAY Statement

Identifies one or more entries for replay from the input catalog.

Alias: PLAY

Notes: If any entry specified in a REPLAY statement is not found in the input catalog, the GREPLAY procedure issues a message to the SAS log. The GREPLAY procedure continues to replay valid entries.

When you replay your graphs, use the same device and orientation that you used when you generated the original graphs. If you use a different device or orientation, your replayed graphs might be distorted.

For a graph with a border, or a graph that includes titles and footnotes, if the border, titles, and footnotes are rendered outside of the GRSEG, they are not included in the replayed graph. See "Considerations for Replaying a Graph" on page 1420.

Syntax

REPLAY *entry-id(s)* | _FIRST_ | _LAST_ | _ALL_;

Required Arguments

entry-id(s)
is the number (in the order in which they were created) or name of a catalog entry or a group of entries in the input catalog. Entries must contain graphics output. Multiple entries can contain both numbers and names. This statement specifies both the entry named GRAPH, and the third entry in the catalog:
`replay graph 3;`

ALL
replays all entries in the input catalog.

FIRST
replays the first entry in the input catalog.

LAST
replays the last entry in the input catalog.

TC Statement

Specifies the template catalog for the GREPLAY procedure.

Note: SAS supplies several templates in the SASHELP.TEMPLT catalog.

Tip: Use the TC statement to change the template catalog without exiting the procedure.

Syntax

TC *template-catalog*;

Required Argument

template-catalog
identifies the SAS catalog where templates are to be stored or identifies the name of a SAS catalog that contains templates.

TCOPY Statement

Copies templates from a catalog to the template catalog, or creates a duplicate of a template within the template catalog.

Requirement: Assign a template catalog before using the TCOPY statement.

See: "TC Statement" on page 1440 .

Syntax

TCOPY <*template-catalog.*>*template-entry*<.TEMPLATE>;

Required Argument

<template-catalog.>template-entry<.TEMPLATE>
identifies the template entry to be copied.

template-catalog
specifies the SAS catalog that contains the template to be copied.

template-entry
specifies the template entry name.

TEMPLATE
specifies the catalog entry type.

Duplicate entry names are resolved as described in "Duplicate Entry Names" on page 1419 .

Details

To copy a template from one catalog to another catalog, specify *template-catalog* as the source catalog. To copy NEWTEMP.TEMPLATE from the catalog named ONE.TEMPLT to the catalog named TARGET.TEMPLT submit the following statements:

```
LIBNAME target "SAS-data-library";
LIBNAME one "SAS-data-library";

proc greplay nofs;
   tc target.templt;
   tcopy one.templt.newtemp.template;
quit;
```

To create a duplicate of a template, simply omit *template-catalog* from your TCOPY statement. For example, to create a duplicate of a template named NEWTEMP within the TEMPLAT catalog, submit the following:

```
tcopy newtemp.template;
```

TDEF Statement

Defines or modifies templates in the template catalog.

Requirement: Assign a template catalog before using the TDEF statement.

See: TC statement
"Example 1: Creating a Template" on page 1459

Syntax

TDEF *template-entry*
panel-number/*<panel-options>*
<panel-number/*<panel-options>*>
<…>
<DES="*description*">
;

Required Argument

template-entry
identifies an existing or new template. If the template is not in the template catalog, it is created by the GREPLAY procedure. If the *template-entry* is in the template catalog, it is modified by the procedure.

Only one template entry is required, but if you specify only the template name without any option, modifications are not made, and a template is not created.

Optional Argument

DES="*description*"
specifies the template entry description. The maximum length for the template entry description is 256 characters.

Default *** new template ***

Panel Options

Panel-options can be one or more of the following options.

CLIP
specifies that any panel behind this panel should be clipped. Only the graphics output to be placed in the CLIP panel can appear in the space that the panel occupies. If a previous panel occupies all or part of that space, CLIP is ignored.

COLOR=*border-color*
specifies the panel border color. In SAS 9.4M4 and in earlier releases, you can specify a valid color name or color code that does not exceed 8 characters. Starting with SAS 9.4M5, you can specify a valid color in any of the color-naming schemes described in "Color-Naming Schemes" on page 368 that does not exceed 64 characters or a valid color code that does not exceed 8 characters.

Restriction In SAS 9.4M4 and in earlier releases, the color name cannot exceed 8 characters.

Notes If you do not specify a border color, then the GREPLAY procedure does not draw a border around the output panel when you replay graphics output to the output panel. In that case, unless the original graphic output contains a border around the graphic, the output panel will not have a border. See "Considerations for Replaying a Graph" on page 1420.

A template that contains a panel without a border color is assigned a color when it is previewed with the PREVIEW statement.

COPY=*panel-number*
specifies the panel number definition to be copied to this panel.

DEF
specifies a default panel with specific coordinates.

Table 44.3 Default Panel Characteristics

Panel Corner	Coordinates
lower left	(0,0)
upper left	(0,100)
upper right	(100,100)
lower right	(100,0)

DELETE
deletes a panel.

Alias DEL

LLX=*x*
specifies the X coordinate of the lower left corner of the panel. Units for *x* are a percentage of the graphics output area.

LLY=*y*
specifies the Y coordinate of the lower left corner of the panel. Units for *y* are a percentage of the graphics output area.

LRX=*x*
specifies the X coordinate of the lower right corner of the panel. Units for *x* are a percentage of the graphics output area.

LRY=*y*
specifies the Y coordinate of the lower right corner of the panel. Units for *y* are a percentage of the graphics output area.

PANEL-NUMBER=
identifies the panel number being defined or modified.

ROTATE=*degrees*
specifies the rotation angle for the panel. Panel corner coordinates are automatically adjusted.

SCALEX=*factor*
specifies the scale factor for the X coordinates in the panel. Use this scale factor to increase or decrease the panel size in the X direction, or to reverse the X coordinates for the panel.

SCALEY=*factor*
specifies the scale factor for the Y coordinates in the panel. Use this scale factor to increase or decrease the panel size in the Y direction, or to reverse the Y coordinates for the panel.

ULX=*x*
specifies the X coordinate upper left corner of the panel. Units for *x* are a percentage of the graphics output area.

ULY=*y*
specifies the Y coordinate upper left corner of the panel. Units for *y* are a percentage of the graphics output area.

URX=*x*
specifies the X coordinate upper right corner of the panel. Units for *x* are a percentage of the graphics output area.

URY=*y*
specifies the Y coordinate upper right corner of the panel. Units for *y* are a percentage of the graphics output area.

XLATEX=*distance*
specifies the distance to move the X coordinates of the panel. Units for *distance* are a percentage of the graphics output area.

XLATEY=*distance*
specifies the distance to move the Y coordinates of the panel. Units for *distance* are a percentage of the graphics output area.

Details

To zoom in on the graphics output, use coordinate values less than 0 and greater than 100. These values can be used with the LLX= option, LLY= option, LRX= option, LRY= option, ULX= option, ULY= option, URX= option, and the URY= option. You can see the replayed graphics output portion in the graphics output area in the range from 0% to 100%.

The values that you specify for the SCALEX= option, and the SCALEY= option are used to change the size and panel orientation. The scale factors are used for the corresponding X and Y panel coordinates. If you submit:

```
scalex=.5
scaley=2
```

the X coordinates are scaled to half the original size, and the Y coordinates are scaled to twice the original size.

If you supply a scale factor of 0, all of the coordinates are set to the same value. If you use a scale factor of 1, nothing happens. If you use a scale factor greater than 1, the values of the coordinates are increased and hence the size of the panel increases. If you use a scale factor less than 1 but greater than 0, the values of the coordinates are reversed, and the panel (and any graphics output replayed in the panel) is reversed.

TDELETE Statement

Deletes templates from the template catalog.

Alias: TDEL

Note: The GREPLAY procedure does not prompt you to confirm your request to delete templates.

Syntax

TDELETE *template-entry(s)* | _ALL_;

Required Arguments

template-entry(s)
identifies a template entry to be deleted from the template entry catalog. You can submit one entry or a list of entries in one TDELETE statement.

ALL
deletes all template entries in the template entry catalog.

TEMPLATE Statement

Assigns a template to use when replaying graphics output.

Requirement: Assign a template catalog before using the TEMPLATE statement.

Note: If you specify a template that is not in the template catalog, before you assign a template entry catalog, the GREPLAY procedure issues an error message.

Syntax

TEMPLATE *template-entry*;

Required Argument

template-entry
identifies an existing template to use when replaying graphics output. Use the TREPLAY statement to replay graphics output in the template.

Example "Example 1: Creating a Template" on page 1459

TREPLAY Statement

Replays graphics entries into template panels. TREPLAY copies one or more entries from the graphics input catalog into a template-entry in the graphics output catalog, using positioning information provided by the template.

Alias: TPLAY

Requirement: Before issuing the TREPLAY statement, specify a graphics input catalog with the GREPLAY IGOUT statement, assign a template entry catalog with the GREPLAY TC statement, and choose a template with the GREPLAY TEMPLATE statement.

Notes: When you replay your graphs, use the same device that you used when you generated the original graphs. If you use a different device, your replayed graphs might be distorted.

For a graph with a border, or a graph that includes titles and footnotes, if the border, titles, and footnotes are rendered outside of the GRSEG, they are not included in the replayed graph. See "Considerations for Replaying a Graph" on page 1420.

See: "IGOUT Statement" on page 1434
"TC Statement" on page 1440
"TEMPLATE Statement" on page 1445
"Example 2: Replaying GSLIDE Procedure Output in a Template" on page 1462

Syntax

TREPLAY <DES="*description*" > <NAME="*entry-name*">
template-panel-number:*entry-id*
<*template-panel-number*:*entry-id*>
<…>;

Required Argument

template-panel-number*:*entry-id
specifies the template panel number and the name of the template entry.

template-panel-number
specifies the number of the panel in the template into which you want to replay the entry.

This number determines the position of the graph in the new entry in the graphics output catalog.

entry-id
specifies the name or number of the entry in the graphics input catalog that is to be added to the new entry in the graphics output catalog.

Optional Arguments

DES="*description*"
specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver that you are using). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.

- the **Description** field of the PROC GREPLAY window.

Default Graphics Replay

NAME="*entry-name*"
specifies the entry name for the template.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name duplicates an existing name, SAS/GRAPH adds a number to the name, or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on).

Default TEMPLATE

See "Duplicate Entry Names" on page 1419

Details

When you replay GRSEG entries in a template, the GREPLAY procedure creates and stores graphics output in the designated output catalog.

You can replay multiple entries in one TREPLAY statement as shown here:

```
treplay 1:plot1 2:plot2 3:chart1;
```

PLOT1 is placed in panel 1 of the current template. PLOT2 is placed in panel 2. CHART1 is placed in panel 3. Specify the entry name or entry number.

Usage: GREPLAY Procedure

Using the GREPLAY Procedure Windows

Opening the GREPLAY Procedure Windows

In the SAS windowing environment, you can use the GREPLAY windows instead of code-based statements to replay and manage catalog entries.

Note: The GREPLAY windows are not available in SAS Studio.

You perform tasks that use the GREPLAY procedure windows by entering values in the fields that are displayed in the windows and by issuing commands from the command line. To open the GREPLAY windows, submit the PROC GREPLAY statement without the NOFS option:

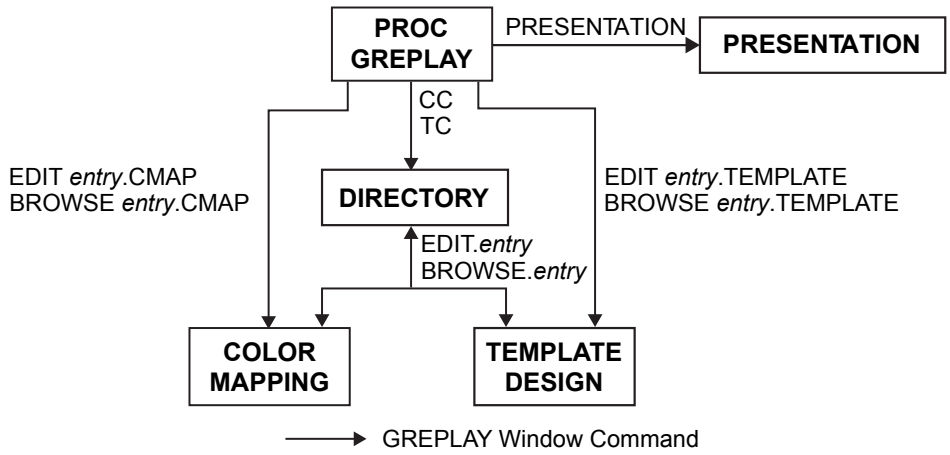
```
proc greplay;
```

SAS/GRAPH then opens the PROC GREPLAY window. The GREPLAY procedure has five windows:

- PROC GREPLAY window
- PRESENTATION window
- DIRECTORY window
- TEMPLATE DESIGN window
- COLOR MAPPING window

Figure 44.171 on page 1449 shows how these windows relate to each other. Each window can be scrolled backward or forward as needed to display additional fields and information.

Figure 44.2 GREPLAY Procedure Windows



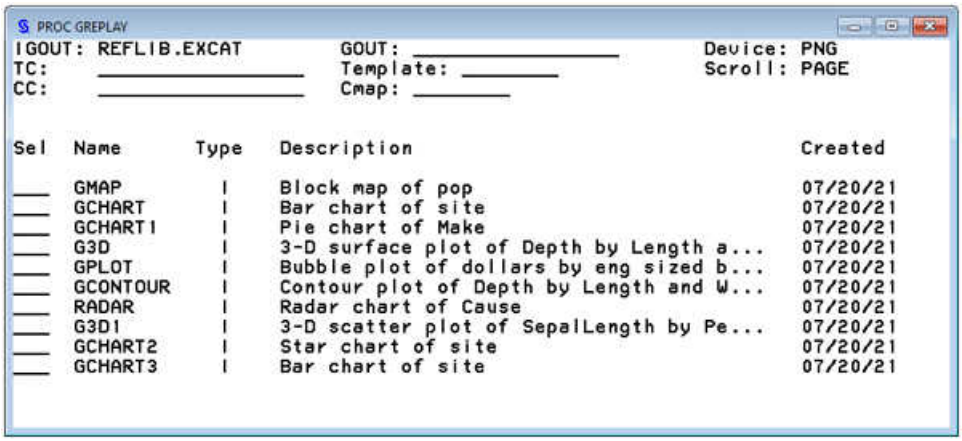
GREPLAY Window Commands

When using GREPLAY windows, tasks are performed by entering values in the fields. Each window can be scrolled forward, or backward to display additional fields and information. You can navigate, and manipulate GREPLAY windows by entering commands on the command line. For a complete description of each window and its fields, open the Help for the GREPLAY windows. You can open the Help by pressing the F1 key or by selecting **Help** ⇨ **Using This Window**. For information about navigating among these windows, see “Commands for Using the GREPLAY Procedure Windows ” on page 1452.

PROC GREPLAY Window

This window is displayed when you submit the PROC GREPLAY statement on a windowing device without the PRESENTATION or NOFS option. The PROC GREPLAY window can be used to replay graphics output, and to manage catalogs that contain graphics output.

Figure 44.3 PROC GREPLAY Window



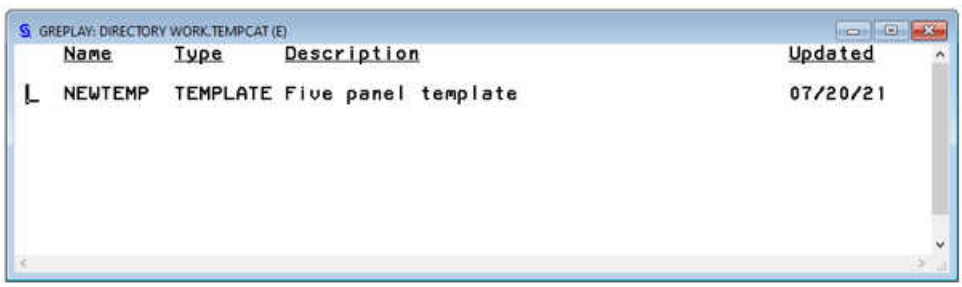
PRESENTATION Window

This window replays graphics output without modifying or deleting entries, templates, or color maps. Once you have created and organized your catalog, you can use the PRESENTATION window in an application for replaying graphics output.

DIRECTORY Window

This window lists the catalog entry names, gives a brief description of each entry, and indicates the date on which each entry was created or last modified. Although all catalog entry types are displayed in the DIRECTORY window, you can manage only entries of the type CMAP and TEMPLATE from this window.

Figure 44.4 Directory Window

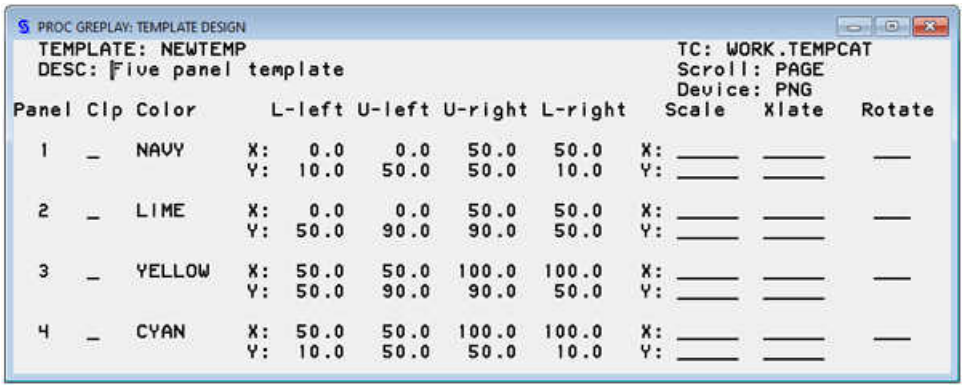


TEMPLATE DESIGN Window

This window enables you to design templates that are used to present graphics. Templates are designed by specifying the coordinates of its panels and assigning

the order in which panels are filled. Once you have entered the coordinates of a panel , you can modify them by using the Scale, Xlate (translate), and Rotate utility fields. These utility fields recalculate the coordinate values automatically.

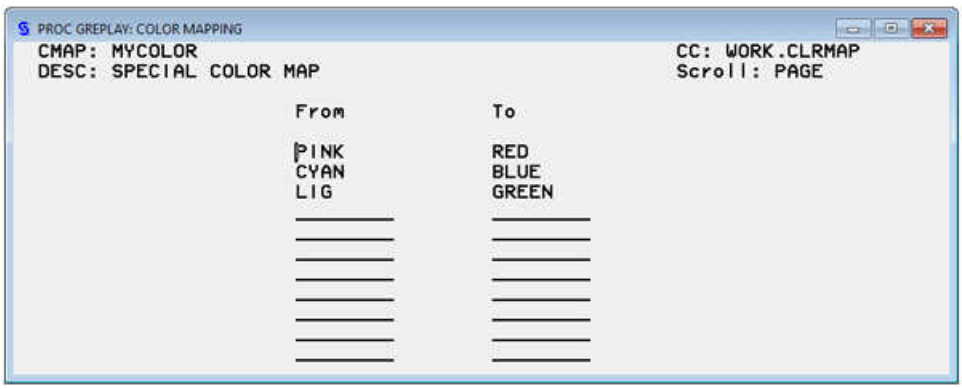
Figure 44.5 *Template Design Window*



COLOR MAPPING Window

This window enables you to map colors in existing graphics output to new colors when you replay the graphics output. Any color in the graphics output that appears in the From column of the color map, is mapped to the corresponding color in the To column of the color map. Using a color map does not change the contents of the replayed graphic output. Using a color map does not produce new graphics output. You can replay your graphics output and assign a current color map.

Figure 44.6 *Color Mapping Window*



Commands for Using the GREPLAY Procedure Windows

Table 44.4 Commands for Using the GREPLAY Procedure Windows

Location	Task	Command
PROC GREPLAY statement	Open PROC GREPLAY window.	Submit the PROC GREPLAY statement without using the PRESENTATION or NOFS options.
	Open PRESENTATION window.	Submit the PROC GREPLAY statement and include the PRESENTATION and IGOUT= options.
PROC GREPLAY window	Open PRESENTATION window.	Specify an IGOUT catalog in the GREPLAY window, and then issue the `PRES` command in the SAS windowing environment command bar.
	Open DIRECTORY window.	Specify a template catalog (TC) in the GREPLAY window, and then issue the `TC` command in the SAS windowing environment command bar. OR Specify a color map catalog (CC) in the GREPLAY window, and then issue the `CC` command in the SAS windowing environment command bar.
	Open TEMPLATE DESIGN window.	Specify a template catalog (TC) in the GREPLAY window, and then issue the following command in the SAS windowing environment command bar: `edit` *`template-name`*`.template`

Location	Task	Command
	Open COLOR MAPPING window.	Specify a color map catalog (CC) in the GREPLAY window, and then issue the following command in the SAS windowing environment command bar: `edit` *`color-map-name`*`.cmap`
DIRECTORY window	Open TEMPLATE DESIGN window.	Place an S beside the name of an existing template in the DIRECTORY window. OR Issue the following command in the SAS windowing environment command bar: `edit` *`template-name`*`.template`
	Open COLOR MAPPING window.	Place an S beside the name of an existing color map in the DIRECTORY window. OR Issue the following command in the SAS windowing environment command bar: `edit` *`color-map-name`*`.cmap`

Using Code-based Statements to Run the GREPLAY Procedure

Invoking the GREPLAY Procedure

If you prefer to run code-based statements in a windowing environment, invoke the GREPLAY procedure with the NOFS option as follows:

```
proc greplay nofs;
```

Once you submit the PROC GREPLAY statement, you can enter and submit statements without resubmitting the PROC GREPLAY statement.

To exit the GREPLAY procedure, you can submit any of the following:

- an END, QUIT, or STOP statement
- another PROC statement or DATA step

Managing Catalogs, Color Maps, and Templates

You can replay entries, manage color maps and templates, or perform catalog management tasks with GREPLAY code-based statements. This section lists several common tasks, and the statements to perform them.

Table 44.5 *GREPLAY Procedure Statements for Managing Color Maps, Templates, and Catalogs*

Task	Statement
assign a color map catalog	CC statement
copy a color map from another catalog, or within the same catalog	CCOPY statement
define or modify a color map in the current catalog	CDEF statement
assign a color map to use when you replay graphics output	CMAP statement
delete unneeded GRSEG entries	DELETE statement
assign a template catalog	TC statement
copy a template from another catalog, or within the same catalog	TCOPY statement
delete a template	TDELETE statement
define a template	TDEF statement
display the panel outlines for a template	PREVIEW statement
assign a template to use when you replay graphics output	TEMPLATE statement
replay an entry in a template panel	TREPLAY statement

Managing GRSEG Catalog Entries

You can replay entries or perform a variety of catalog management tasks with GREPLAY code-based statements. The following table lists several common tasks and the statements that you use to perform them.

Table 44.6 *GREPLAY Procedure Statements for Managing GRSEG Catalog Entries*

Task	Statement
copy GRSEG entries from an input catalog to an output catalog[1]	COPY statement
group GRSEG entries	GROUP statement
move GRSEG entries	MOVE statement
delete GRSEG entries	DELETE statement
modify GRSEG entry names or descriptions in an input catalog	MODIFY statement
replay GRSEG entries from an input catalog	REPLAY statement
replay GRSEG entries into template panels	TREPLAY statement

1 You must assign an output catalog before copying graphics output.

Using the GREPLAY Procedure to Replay Catalog Entries

To select catalog entries for replay, first assign an input catalog that contains the graphics output that is to be replayed. Then assign the entry with the REPLAY statement. To select a catalog entry or entries for replay:

1. Start the GREPLAY procedure.
2. Define the input catalog that contains the graphics to be replayed with the IGOUT= option.
3. Specify the entry or entries that you want to replay (GCHART in the example that follows) with the REPLAY statement.
4. End the GREPLAY procedure with the QUIT statement.

For example, the following statements replay the GRSEG entry named GCHART from the catalog WORK.GSEG, which is assigned with the IGOUT= option:

```
proc greplay igout=work.gseg nofs;
   replay gchart;
quit;
```

To replay all the graphics output stored in the WORK.GSEG catalog submit this code:

```
proc greplay nofs;
   igout work.gseg;
   replay _all_ ;
quit;
```

Note: Graphics output is created only when you use the GREPLAY procedure with a template.

Using the GREPLAY Procedure to Create Custom Templates

You can use the GREPLAY procedure to create custom templates. Custom templates are typically used to perform the following actions:

- control the layout of multiple graphs on one page, which is useful for dashboards
- replay graphics output from several catalog entries, or from the same catalog, on one display or page
- change the shape of your graphics output
- change the size of your graphics output

To define and view a custom template:

1. Start the GREPLAY procedure with the NOFS option.
2. Assign a template catalog with the TC= option.
3. Define a template with the TDEF statement.
4. Preview the template with the PREVIEW statement.
5. End the GREPLAY procedure with the QUIT statement.

Before you create a template, you must assign a template catalog. If you are using the GREPLAY procedure in line mode, use the TDEF statement to define a template and the PREVIEW statement to preview a template. For example, the following statements define and preview a template named TEMPLT:

```
proc greplay nofs tc=sasuser.cat;
   tdef templt 1/def;
   preview templt;
quit;
```

Using the GREPLAY Procedure to Replay Graphics Output in a Template

You can use the GREPLAY procedure to create new graphics output by replaying existing graphics output in templates. Templates are often used to replay several graphics entries from the same catalog on one display or page. The GREPLAY procedure creates new graphics output when replaying graphics output with a template.

You can create your own templates, or you can use the templates provided with SAS/GRAPH that are stored in the SASHELP.TEMPLT catalog.

The following guidelines describe how to generate two graphs, and replay the graphics output on one page using a SASHELP.TEMPLT entry:

1. Generate two graphs with PROC GCHART using the default names (GCHART GCHART1).
2. Start the GREPLAY procedure with the NOFS option specified.
3. Define the input catalog with the IGOUT= option (WORK.GSEG).
4. Assign the template catalog with the TC= option (SASHELP.TEMPLT contains the template entries).
5. Assign a template to replay your graphs with the TEMPLATE statement (V2).
6. Assign the graphs that you want to replay with the TREPLAY statement (GCHART GCHART1).
7. End the GREPLAY procedure with the QUIT statement.

For example, the following statements replay the entries GRAPH1 and GRAPH2 into the V2 template, which is stored in the catalog SASHELP.TEMPLT. The TC statement specifies the catalog that contains the template, and the TEMPLATE statement specifies the template. The TREPLAY statement assigns each entry to a panel. (The V2 template has two panels, so there is an assignment for panel 1 and panel 2.)

```
proc gchart data=sashelp.class;
   hbar age/discrete;
 run;
    hbar height;
run;
quit;
proc greplay igout=work.gseg nofs;
   tc sashelp.templt;
   template v2;
   treplay 1:gchart 2:gchart1;
quit;
```

Note: If the GOUT= option is not specified when creating the charts, then the output is stored in the temporary WORK.GSEG catalog.

When you replay graphics output in a template, the new GRSEG output that is created by the GREPLAY procedure is automatically provided a default name, Template, and it is stored in the output catalog WORK.GSEG. The default GRSEG description is “Graphics Replay”.

Using the GREPLAY Procedure to Create Color Maps

Color maps are useful for assigning unavailable colors on your current device to your graph. A color map is a list of up to 256 pairs of colors. By mapping the original colors to a different list of colors, you can change the colors in your graphics output.

To create a color map named CLRMAP, perform the following actions:

1. Start the GREPLAY procedure with the NOFS option.
2. Assign a color map catalog with the CC= option.
3. Define the output catalog with the GOUT= option.
4. Define a color map with the CDEF statement.
5. Remap your colors.
6. End the GREPLAY procedure with the QUIT statement.

Before you create a color map, you must assign a color map catalog. The following example defines a color map named CLRMAP:

```
proc greplay cc=clrmap gout=work nofs;
   cdef clrmap 1 / cyan : blue;
quit;
```

When you assign a color map and replay graphics output, the following occurs:

- The stored GRSEG entry or entries, retain the original foreground colors.
- The colors used to replay the graphics are not saved with the original graphics output.
- Graphics output is not created when you replay graphics output using a color map.

Examples: GREPLAY Procedure

Example 1: Creating a Template

Features: REPLAY statement options:
NOFS
TC=
TDEF statement
TEMPLATE statement

Sample library member: GRECRTM1

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example creates a template with five panels. Four panels are small and equal in size. The fifth panel is a larger, full-size panel that can be used to display a common title or footnote for the entire template. In this example, the LIST statement displays the template contents in the SAS log. Output 44.30 on page 1460 shows the template definition written to the SAS log file. The template defined here is also used in “Example 2: Replaying GSLIDE Procedure Output in a Template” on page 1462.

Output 44.1 *Defining a Template*

```
.
.
.

64        /* list template contents */
65      list template;

 NEWTEMP       Five panel template

 Pan Clp Color      Ll-x  Ll-y  Ul-x  Ul-y  Ur-x  Ur-y  Lr-x  Lr-y

  1    NAVY        0.0  10.0   0.0  50.0  50.0  50.0  10.0
  2    LIME       0.0  50.0   0.0  90.0  50.0  90.0  50.0  50.0
  3    YELLOW      50.0  50.0  50.0  90.0 100.0  90.0 100.0  50.0
  4    CYAN       50.0  10.0  50.0  50.0 100.0  50.0 100.0  10.0
  5    LIPK        0.0   0.0   0.0 100.0 100.0 100.0 100.0   0.0

66   quit;
.
.
.
```

Program

```
goptions reset=all border;

proc greplay tc=work.tempcat nofs;

tdef newtemp des="Five panel template"
     1/llx=0   lly=10
       ulx=0   uly=50
       urx=50  ury=50
       lrx=50  lry=10
       color=navy

     2/llx=0   lly=50
       ulx=0   uly=90
       urx=50  ury=90
       lrx=50  lry=50
       color=lime

     3/llx=50 lly=50
       ulx=50 uly=90
       urx=100 ury=90
       lrx=100 lry=50
       color=yellow

     4/llx=50 lly=10
       ulx=50 uly=50
       urx=100 ury=50
       lrx=100 lry=10
       color=cyan

     5/llx=0 lly=0
```

```
        ulx=0 uly=100
        urx=100 ury=100
        lrx=100 lry=0
        color=lipk;

template newtemp;

 list template;
 quit;
```

Program Description

Here is a detailed example of the example program.

Set the graphics environment.

```
goptions reset=all border;
```

Start the GREPLAY procedure. NOFS starts the procedure in line-mode. The TC= option assigns TEMPCAT as the template catalog.

```
proc greplay tc=work.tempcat nofs;
```

Define a template with five panels. The TDEF statement defines a template named NEWTEMP, and places it in the previously defined template catalog. Each definition identifies the panel number, and specifies the four corner's coordinates. The COLOR= option draws a border for each panel in the specified color.

```
tdef newtemp des="Five panel template"
     1/llx=0   lly=10
       ulx=0   uly=50
       urx=50  ury=50
       lrx=50  lry=10
       color=navy

     2/llx=0   lly=50
       ulx=0   uly=90
       urx=50  ury=90
       lrx=50  lry=50
       color=lime

     3/llx=50 lly=50
       ulx=50 uly=90
       urx=100 ury=90
       lrx=100 lry=50
       color=yellow

     4/llx=50 lly=10
       ulx=50 uly=50
       urx=100 ury=50
       lrx=100 lry=10
       color=cyan

     5/llx=0 lly=0
        ulx=0 uly=100
        urx=100 ury=100
        lrx=100 lry=0
        color=lipk;
```

Assign the template. The TEMPLATE statement assigns the created template NEWTEMP as the template.

```
template newtemp;
```

Write the template contents to the SAS log.

```
list template;
quit;
```

Example 2: Replaying GSLIDE Procedure Output in a Template

Features: GREPLAY statement options:
GOUT=,
IGOUT=
TEMPLATE=

TREPLAY statement

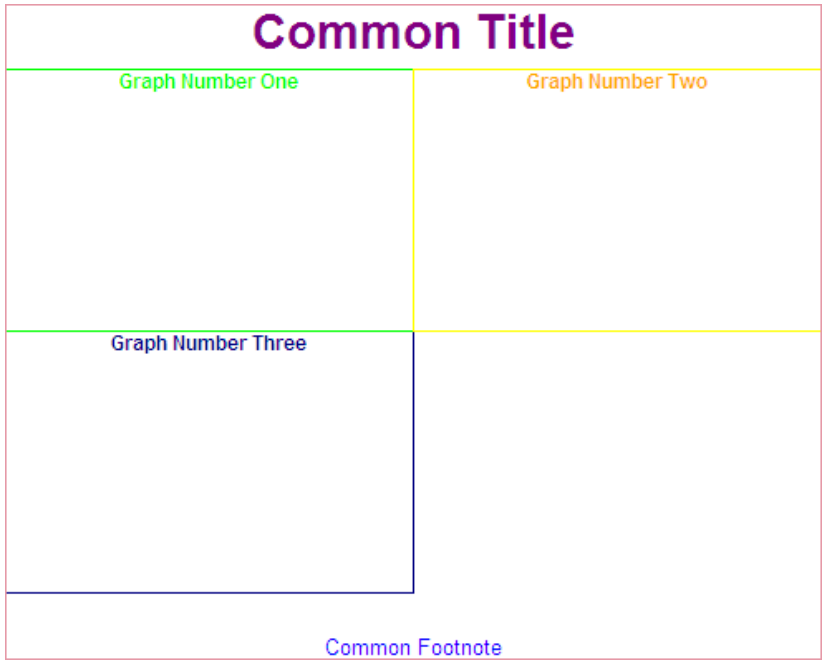
PROC GSLIDE

Sample library member: GRERGOT1

Notes: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example requires template NEWTEMP, which is created in “Example 1: Creating a Template” on page 1459. In order to run this program, you must first run the program in “Example 1: Creating a Template” on page 1459 to create the NEWTEMP template. Otherwise, an error will occur in this program.

This program uses template NEWTEMP, which is created by the program in “Example 1: Creating a Template” on page 1459. The TREPLAY statement in this example replays four catalog entries that contain graphics output into template NEWTEMP. The NEWTEMP template contains four equally sized panels, and one large, full-size panel. Note that assignments are made to all but one panel. Because the fourth panel is not listed in the TREPLAY statement, it does not appear in the graphics output. The HSIZE= option, and the VSIZE= option are adjusted, to reflect the overall output dimension. Alternatively, you could use XPIXELS= and YPIXELS= to adjust the graphics output size.



Program

```
goptions reset=all border hsize=5.14in vsize=4.2in;

proc gslide gout=grafcat;
   title c=navy "Graph Number Three";
run;
   title c=lime "Graph Number One";
run;
   title c=orange "Graph Number Two";
run;


proc gslide gout=grafcat;
   title c=purple "Common Title";
   footnote c=blue "Common Footnote";
run;

proc greplay igout=grafcat gout=excat tc=tempcat nofs;
   template=newtemp;

   treplay 1:gslide
           2:gslide1
           3:gslide2
           5:gslide3;
         quit;
```

Program Description

Here is a detailed example of the example program.

Set the graphics environment. The HSIZE= option, and the VSIZE= option are set for the overall output dimensions.

```
goptions reset=all border hsize=5.14in vsize=4.2in;
```

Generate three graphs in the WORK.GRAFCAT catalog. The GSLIDE procedure creates three text slides, and stores them in GRAFCAT as specified by the GOUT= option. These are stored as GSLIDE, GSLIDE1, and GSLIDE2.

```
proc gslide gout=grafcat;
   title c=navy "Graph Number Three";
run;
   title c=lime "Graph Number One";
run;
   title c=orange "Graph Number Two";
run;
```

Generate a text slide with PROC GSLIDE and output to GRAFCAT. Define a title and a footnote for the container output.

```
proc gslide gout=grafcat;
   title c=purple "Common Title";
   footnote c=blue "Common Footnote";
run;
```

Start the GREPLAY procedure. The IGOUT= option, assigns GRAFCAT as the input catalog. The GOUT= option assigns EXCAT as the output catalog. The TC=TEMPCAT option assigns the template catalog for the GREPLAY procedure. The TEMPLATE=NEWTEMP option assigns NEWTEMP as the current template.

```
proc greplay igout=grafcat gout=excat tc=tempcat nofs;
   template=newtemp;
```

Replay three graphs into template. The TREPLAY statement assigns three entries to panels in the NEWTEMP template. Each assignment is a panel number, and the name of a graphics output entry. Names are the default names assigned by the GSLIDE procedure.

```
   treplay 1:gslide
           2:gslide1
           3:gslide2
           5:gslide3;
         quit;
```

Example 3: Replaying Graphs into a Template

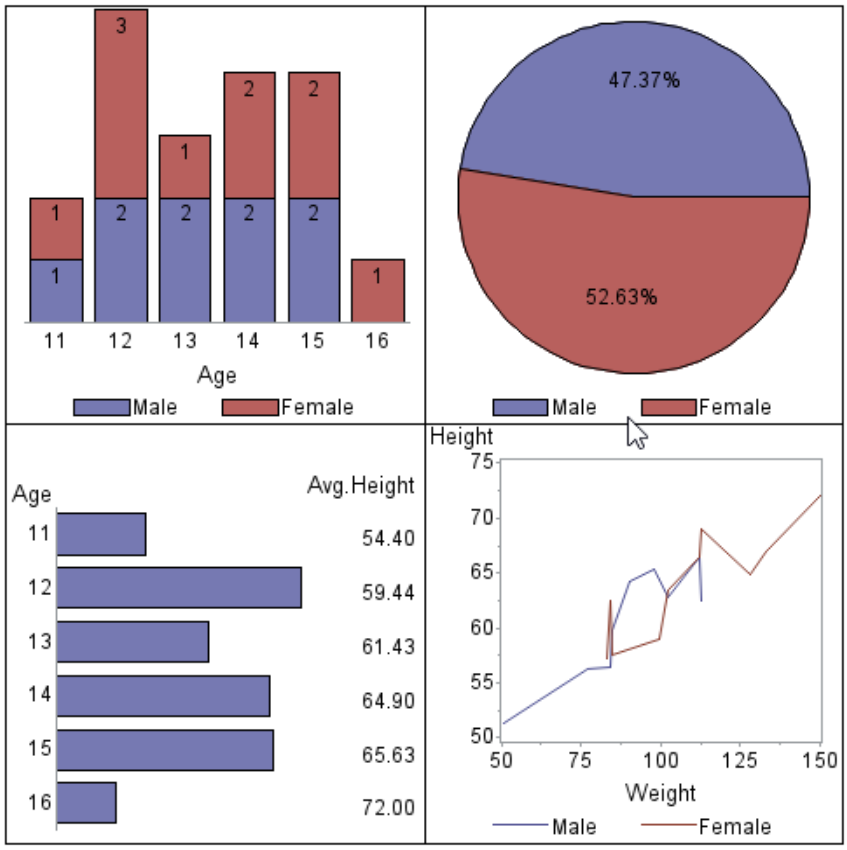
Features:

GREPLAY statement options:
- IGOUT=
- GOUT=
- NOFS
- TC=
- TEMPLATE=

DEVICE statement

TREPLAY statement

This example replays four graphs into template L2R2. The following figure shows the output of this example program.



Program

```
data work.classtemp (drop=name );
   length Gender $ 6;
   set sashelp.class;
   if sex="F" then Gender="Female";
      else Gender="Male";
run;
proc sort data=work.classtemp out=work.class;
   by weight height;
run;


proc greplay igout=work.excat nofs;
   delete _all_;
run;

goptions reset=all hsize=2.5in vsize=2.5in nodisplay;

axis1 label=none style=0 major=none value=none;
axis2 label=("Age");
axis3 label=("Height") order=50 to 75 by 5;
axis4 label=("Weight") order=50 to 150 by 25 minor=(n=1);

legend1 label=none value=("Male" "Female") Position=(bottom)
```

```
across=2 shape=bar(15, 2);
legend2 label=none value=("Male" "Female") shape=line(15);
symbol i=join;

symbol i=join;

proc gchart data=work.class gout=work.excat;
   vbar age/discrete hminor=0 subgroup=gender width=14
                 inside=freq  raxis=axis1 maxis=axis2
                 noframe legend=legend1;
run;
   hbar age/ discrete sumvar=height mean
                  meanlabel="Avg.Height" vminor=0
                  raxis=axis1 maxis=axis2;
run;
   pie gender/ noheading legend=legend1 percent=inside;
run;
proc gplot data=work.class gout=work.excat;
    plot height*weight=gender/ vminor=1 vaxis=axis3
    haxis=axis4 legend=legend2;
run;
quit;

goptions reset=all hsize=5in vsize=5in;

proc greplay gout=work.excat igout=work.excat nofs
   tc=sashelp.templt template=l2r2;
   device png;
   treplay 1:gchart 2:gchart1 3:gchart2 4:gplot;
quit;
```

Program Description

Here is a detailed example of the example program.

Prepare the data for the graphs. Drop variables NAME and SEX, and add a variable called GENDER. Change variable values, and sort the data for the line graph.

```
data work.classtemp (drop=name );
   length Gender $ 6;
   set sashelp.class;
   if sex="F" then Gender="Female";
      else Gender="Male";
run;
proc sort data=work.classtemp out=work.class;
   by weight height;
run;
```

Delete any entries in catalog Work.Excat.

```
proc greplay igout=work.excat nofs;
   delete _all_;
run;
```

Define the size of each individual graph. Each graph is replayed into a separate panel in the template.

```
goptions reset=all hsize=2.5in vsize=2.5in nodisplay;
```

Create axes definitions for the graphs.

```
axis1 label=none style=0 major=none value=none;
axis2 label=("Age");
axis3 label=("Height") order=50 to 75 by 5;
axis4 label=("Weight") order=50 to 150 by 25 minor=(n=1);
```

Create legend definitions for the graphs.

```
legend1 label=none value=("Male" "Female") Position=(bottom)
across=2 shape=bar(15, 2);
legend2 label=none value=("Male" "Female") shape=line(15);
symbol i=join;
```

Create a symbol definition for the plot.

```
symbol i=join;
```

Generate the graphs, and store them in the WORK.EXCAT catalog. Generate a vertical bar chart, a horizontal bar chart, a pie chart, and a subgrouped plot.

```
proc gchart data=work.class gout=work.excat;
   vbar age/discrete hminor=0 subgroup=gender width=14
                  inside=freq  raxis=axis1 maxis=axis2
                  noframe legend=legend1;
run;
   hbar age/ discrete sumvar=height mean
                   meanlabel="Avg.Height" vminor=0
                   raxis=axis1 maxis=axis2;
run;
   pie gender/ noheading legend=legend1 percent=inside;
run;
proc gplot data=work.class gout=work.excat;
    plot height*weight=gender/ vminor=1 vaxis=axis3
    haxis=axis4 legend=legend2;
run;
quit;
```

Define the size of the template. Each graph is replayed into a separate panel in the template. The template size accommodates the four smaller graphs.

```
goptions reset=all hsize=5in vsize=5in;
```

Replay the graphs with a template to create one graph. The graphs stored in WORK.EXCAT are replayed to create one graph. The graph is also stored in the WORK.EXCAT catalog.

```
proc greplay gout=work.excat igout=work.excat nofs
   tc=sashelp.templt template=l2r2;
   device png;
   treplay 1:gchart 2:gchart1 3:gchart2 4:gplot;
quit;
```

Example 4: Replaying a Graph to Multiple Destinations

Features:

GREPLAY statement options:
- IGOUT=
- NOFS
- TC=

DELETE statement

TDEF statement

TDELETE statement

TREPLAY statement

FILENAME statement

ODS HTML statement

ODS PDF statement

ODS RTF statement

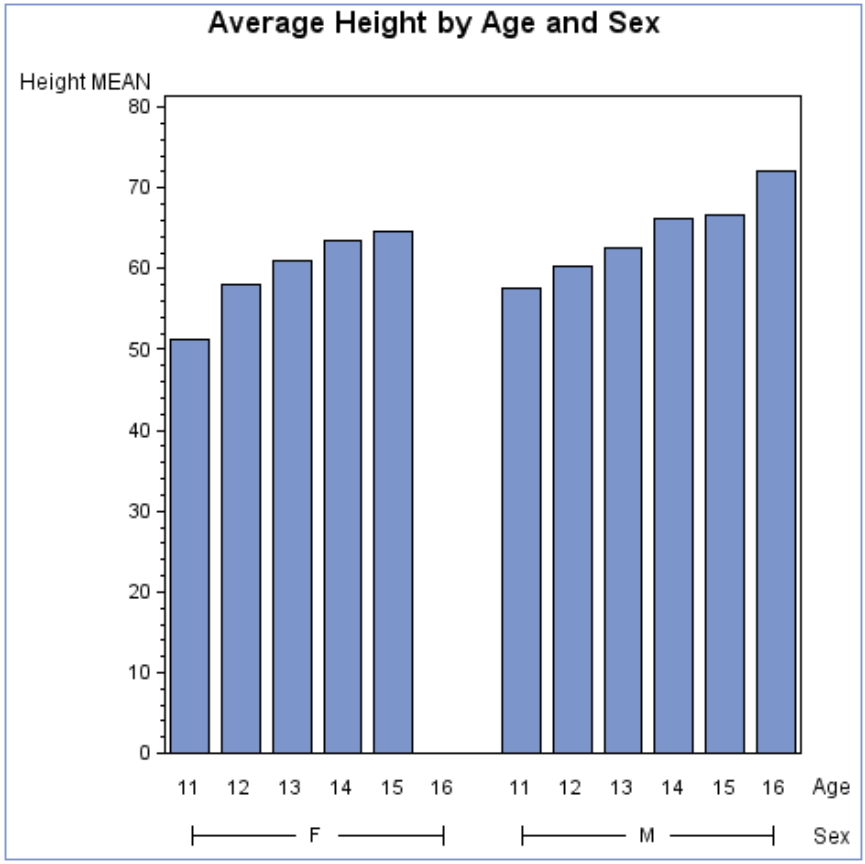
GOPTIONS statement:
- BORDER
- DEVICE=
- GSFNAME=
- NOBORDER
- NODISPLAY
- XPIXELS=
- YPIXELS=

OPTIONS statement ORIENTATION= option

PROC GCHART

The GREPLAY procedure replays the graph shown in Output 44.31 on page 1469 to a TIF file, and the ODS HTML, PDF, and RTF destinations. A template is used to generate the initial graph. The template creates a margin and a blue border around the graph. The XPIXELS= and YPIXELS= graphics options are used to set the graph size. The GREPLAY procedure TREPLAY statement is used to replay the graph to each of the destinations. For the ODS PDF and RTF destinations, the system option ORIENTATION= is set to LANDSCAPE to orient the graph in landscape in both documents.

Output 44.2 *Vertical Bar Chart Exported to Multiple Destinations*



Program

```
%let goutpath=.;
%let goutname=height;

proc greplay nofs tc=work.tempcat igout=work.gseg;
   delete &goutname;
   tdelete grtemp;
run;
quit;

proc greplay tc=work.tempcat nofs;
   tdef grtemp des="One panel template with border"
      1/llx=5   lly=5
        ulx=5   uly=95
        urx=95  ury=95
        lrx=95  lry=5
      color=cx7C95CA;
run;
quit;

goptions reset=all nodisplay noborder xpixels=520 ypixels=520;
```

```
ods _all_ close;

ods listing style=listing;

title "Average Height by Age and Sex";
proc gchart data=sashelp.class;
   vbar age / sumvar=height discrete type=mean group=sex
name="&goutname";
run;
quit;

filename tifout "&goutpath\&goutname..tif";

goptions display device=tiffp gsfname=tifout;
proc greplay nofs tc=work.tempcat igout=work.gseg;
   template grtemp;
   treplay 1:&goutname name="t&goutname";
run;
quit;

goptions reset=all device=png noborder xpixels=520 ypixels=520;

options orientation=landscape;
ods listing close;
ods html path="&goutpath" file="&goutname..html" style=listing;
ods pdf file="&goutpath\&goutname..pdf" style=listing;
ods rtf file="&goutpath\&goutname..rtf" style=listing image_dpi=72;
proc greplay nofs tc=work.tempcat igout=work.gseg;
   template grtemp;
   treplay 1:&goutname name="t&goutname";
run;
quit;

options orientation=portrait;

ods _all_ close;
ods html; /* Not required in SAS Studio */
```

Program Description

Here is a detailed example of the example program.

Define variables to use for specifying the graph output path and filename. The output path is set to the current working directory in this example. Using macro variables makes it easier to change the path and filename, if necessary.

```
%let goutpath=.;
%let goutname=height;
```

Delete the old GRSEG and template, if they exist. The DELETE statement deletes entry HEIGHT from the WORK catalog, which is specified by the IGOUT= option. The TDELETE statement deletes template GRTEMP in the WORK.TEMPCAT catalog, which is specified by the TC= option.

```
proc greplay nofs tc=work.tempcat igout=work.gseg;
   delete &goutname;
   tdelete grtemp;
run;
quit;
```

Create a one-panel template for the graph. This template specifies a 5% margin around the graph. The COLOR= option specifies a blue border around the graph.

```
proc greplay tc=work.tempcat nofs;
   tdef grtemp des="One panel template with border"
      1/llx=5   lly=5
        ulx=5   uly=95
        urx=95  ury=95
        lrx=95  lry=5
      color=cx7C95CA;
run;
quit;
```

Set the graphics options. The GOPTIONS statement NODISPLAY option prevents the chart from being displayed when it is generated. The NOBORDER option disables the graph border, since template GRTEMP draws its own border. Options XPIXELS= and YPIXELS= set the graph dimensions to 520–by-520 pixels.

```
goptions reset=all nodisplay noborder xpixels=520 ypixels=520;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Open the LISTING destination. Use the LISTING style.

```
ods listing style=listing;
```

Generate the chart. The NAME= option in the VBAR statement sets the name of the GRSEG entry to HEIGHT.

```
title "Average Height by Age and Sex";
proc gchart data=sashelp.class;
   vbar age / sumvar=height discrete type=mean group=sex
name="&goutname";
run;
quit;
```

Create a fileref for the graphics output file.

```
filename tifout "&goutpath\&goutname..tif";
```

Replay GRSEG entry HEIGHT to TIFF file height.tif. The GOPTIONS statement DEVICE= option sets the output device to TIFFP, and the GSFNAME= option specifies the output filename as height.tif. The TREPLAY statement replays GRSEG entry HEIGHT to entry THEIGHT using the panel 1 template in template GRTEMP.

```
goptions display device=tiffp gsfname=tifout;
proc greplay nofs tc=work.tempcat igout=work.gseg;
   template grtemp;
   treplay 1:&goutname name="t&goutname";
run;
quit;
```

Change device to PNG and disable borders. Because a border is included in the original GRSEG, the GOPTIONS statement NOBORDER option is used to prevent ODS from drawing another border around the image.

```
goptions reset=all device=png noborder xpixels=520 ypixels=520;
```

Replay GRSEG entry HEIGHT to the HTML, PDF, and RTF destinations. The OPTIONS statement ORIENTATION=LANDSCAPE option sets the graph orientation to landscape in the PDF and RTF documents. This option does not affect the ODS HTML destination. The LISTING style is used with each destination.

```
options orientation=landscape;
ods listing close;
ods html path="&goutpath" file="&goutname..html" style=listing;
ods pdf file="&goutpath\&goutname..pdf" style=listing;
ods rtf file="&goutpath\&goutname..rtf" style=listing image_dpi=72;
proc greplay nofs tc=work.tempcat igout=work.gseg;
   template grtemp;
   treplay 1:&goutname name="t&goutname";
run;
quit;
```

Reset the ORIENTATION= system option to PORTRAIT.

```
options orientation=portrait;
```

Close all of the open destinations, and then open the HTML destination.

```
ods _all_ close;
ods html; /* Not required in SAS Studio */
```

Example 5: Creating a Color Map

Features:	GREPLAY statement options: CC= GOUT= CDEF statement CMAP statement LIST statement
Sample library member:	GRECRCM1
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses the CDEF statement to define a color map. The LIST statement is used in this example to display the color map definition in the SAS log. Output 44.32 on page 1473 shows a partial SAS log listing.

Output 44.3 *Defining a Color Map (GRECRCM1)*

```
.
.
75        /* list  color map contents */
76     list cmap;

 MYCOLOR       Special Color Map

       FROM        TO

  1    PINK        RED
  2    CYAN        BLUE
  3    LIG         GREEN

77  quit;
.
```

Program

```
goptions reset=all border;

proc greplay cc=clrmap gout=excat nofs;

  cdef mycolor des="Special Color Map"
        1 / pink  : red
        2 / cyan  : blue
        3 / lig     : green;

   cmap mycolor;
      list cmap;
   quit;
```

Program Description

Here is a detailed example of the example program.

Set the graphics environment.

```
goptions reset=all border;
```

Start the GREPLAY procedure. The CC= option assigns CLRMAP as the color map catalog. The GOUT= option assigns EXCAT as the graphics output catalog. In order for the color map to be created, both options must be assigned.

```
proc greplay cc=clrmap gout=excat nofs;
```

Define a color map. The CDEF statement defines a color map named MYCOLOR that contains three color pairs.

```
cdef mycolor des="Special Color Map"
      1 / pink  : red
      2 / cyan  : blue
      3 / lig     : green;
```

Specify a current color map, and write contents to the SAS log. The CMAP statement assigns MYCOLOR as the current color map. The contents of CMAP are listed in the SAS log.

```
cmap mycolor;
   list cmap;
quit;
```

Chapter 45

GSLIDE Procedure

Overview: GSLIDE Procedure

About the GSLIDE Procedure

The GSLIDE procedure is useful for creating text slides for presentations. You can overlay text slides on other graphics output with the GREPLAY procedure. The GSLIDE procedure produces graphics output that consists of text and straight lines that are generated by TITLE, FOOTNOTE, and NOTE statements. In addition, the procedure provides an easy way to add titles, notes, and footnotes to output that is produced entirely with an Annotate data set.

About Text Slides

Text slides contain text and graphics that are generated by SAS/GRAPH statements.

Figure 45.176 on page 1476 shows a slide containing text that was produced with TITLE, FOOTNOTE, and NOTE statements.

Figure 45.1 *Text Slide Produced by the GSLIDE Procedure (GSLTEXTS)*

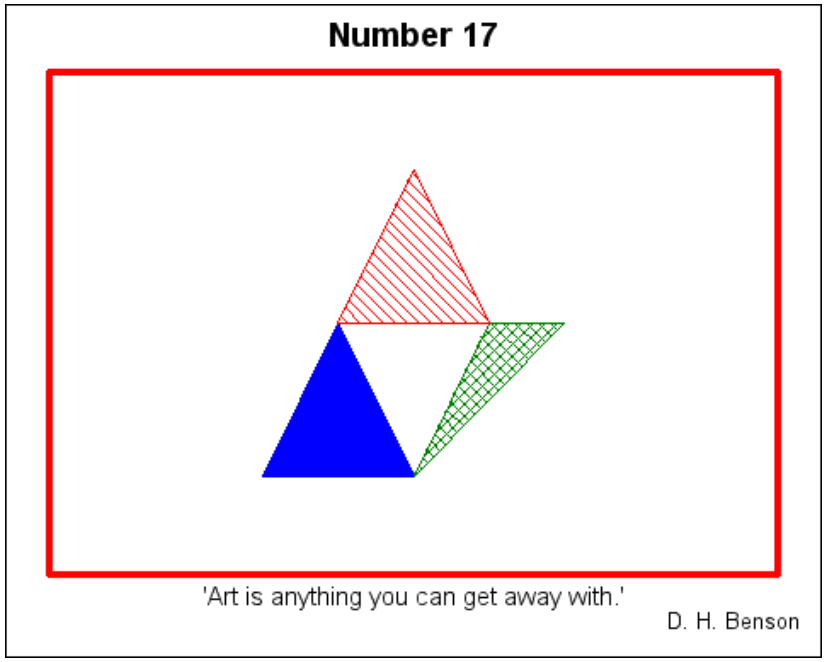
The program for this slide is in "Example 1: Producing Text Slides" on page 1483.

About Annotate Output

Annotate output is generated by commands that are stored in an Annotate data set. Use the GSLIDE procedure to display Annotate output when you want to include TITLE and FOOTNOTE statements on the output and use certain graphics options such as the BORDER option. To display Annotate graphics without these, use the GANNO procedure. See Chapter 28, "Using Annotate Data Sets," on page 721 for more information about creating and displaying Annotate data sets.

Figure 45.177 on page 1477 shows output from an Annotate data set that is displayed with titles and footnotes that were generated by TITLE and FOOTNOTE statements.

Figure 45.2 *Output from an Annotate Data Set Displayed with the GSLIDE Procedure (GSLANNOT)*



The program for this slide is in "Example 2: Displaying Annotate Graphics" on page 1484.

Syntax: GSLIDE Procedure

Restriction: The GSLIDE procedure is not supported by the Java and ActiveX devices. If you use one of these devices with the GSLIDE procedure, the SAS/GRAPH software uses the PNG device driver to generate the output instead.

Requirement: At least one of these is required: a TITLE, FOOTNOTE, or NOTE statement; an appearance option; the BORDER graphics option.

Supported Statements: FOOTNOTE, TITLE

Supports: RUN-group processing

Note: The procedure can include the SAS/GRAPH NOTE statement. See "TITLE (global) Statement, FOOTNOTE (global) Statement, NOTE (local) Statement".

Tip: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, `ODS PDF; procedure-code; ODS PDF CLOSE;`). See Chapter 7, "Using Run-Group Processing," on page 75 for more information.

PROC GSLIDE <*options*>;

PROC GSLIDE Statement

Creates a text slide. Can also provide a border, specify annotation, and assign an output catalog. This is the only statement in the procedure.

Syntax

PROC GSLIDE <*options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set that includes Annotate variables that identify graphics commands and parameters.

BORDER
: draws a border around the graphics output area, which includes the title area, the footnote area, and the procedure output area.

CFRAME=*frame-color*
: draws a frame around the procedure output area in the specified color.

FRAME
: draws a frame around the procedure output area.

IFRAME=*fileref* | '*external-file*'
: specifies the image file that you want to apply to the procedure output area.

IMAGESTYLE=TILE | FIT
: specifies whether to tile the image specified with the IFRAME= option to fill the backplane or to stretch the image to fit the backplane.

LFRAME=*line-type*
: specifies the line type for a frame and draws a frame around the procedure output area.

WFRAME=*n*
: specifies the width of the frame where *n* is a number.

Catalog options

DESCRIPTION='*description*'
: specifies a description of the output.

GOUT=<*libref.*>*output-catalog*
: specifies the SAS catalog in which to save the graphics output produced by the GSLIDE procedure.

NAME="*name*"
: specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Image map options

IMAGEMAP= *output-data-set*
creates a temporary SAS data set that is used to generate an image map.

Optional Arguments

You can specify as many options as you want and list them in any order.

ANNOTATE=*Annotate-data-set*
specifies a data set that includes Annotate variables that identify graphics commands and parameters.

Alias ANNO=*Annotate-data-set*

See Chapter 28, "Using Annotate Data Sets," on page 721

Example "Example 2: Displaying Annotate Graphics" on page 1484

BORDER
draws a border around the graphics output area, which includes the title area, the footnote area, and the procedure output area. A color specification for the border is searched for in the following order:

1. the CTITLE= option in a GOPTIONS statement.
2. the CTEXT= option in a GOPTIONS statement.
3. the color of the current style. If the NOGSTYLE option is specified, then the color is the first color in the device's color list

See "Adding Frames, Borders, and Images" on page 1482

CFRAME=*frame-color*
draws a frame around the procedure output area in the specified color. If you use both the CFRAME= and FRAME options, the FRAME option is ignored. If you use the IFRAME= option, the specified image fills the background of the slide.

Note The CFRAME= option does not color the background of the slide.

See "Adding Frames, Borders, and Images" on page 1482

Example "Example 1: Producing Text Slides" on page 1483

DESCRIPTION='*description*'
specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver that you are using). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

Alias DES=

Default Graphics text slide

FRAME
draws a frame around the procedure output area. By default, the frame color is the color of the current style. If the NOGSTYLE option is specified, then the color is the first color in the device's color list. If you want to specify a different color for the frame, use the CFRAME= option instead.

See "Adding Frames, Borders, and Images" on page 1482

GOUT=<*libref.*>*output-catalog*
specifies the SAS catalog in which to save the graphics output produced by the GSLIDE procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

See "Specifying the Catalog Name and Entry Name for Your GRSEGs " on page 138

IFRAME=*fileref* | '*external-file*'
specifies the image file that you want to apply to the procedure output area.

Interaction This option is overridden by the NOIMAGEPRINT graphics option.

See the IMAGESTYLE= option

IMAGEMAP= *output-data-set*
creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 255 and "Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" on page 262

IMAGESTYLE=TILE | FIT
specifies whether to tile the image specified with the IFRAME= option to fill the backplane or to stretch the image to fit the backplane. The TILE value is the default. See also the IFRAME= option.

LFRAME=*line-type*
specifies the line type for a frame and draws a frame around the procedure output area. Values for *line-type* are 1 through 46. Line types are shown in Figure 25.52 on page 503 . By default, the line type is specified by the current style. LFRAME=1, which produces a solid line, is the default.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.

- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
  - If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

    Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

  - The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GSLIDE

WFRAME=*n*

specifies the width of the frame where *n* is a number. The thickness of the frame increases directly with *n*, but the thickness of the line can vary from device to device. By default, the line width is specified by the current style. WFRAME=1, which is the thinnest line, is the default. The WFRAME= option also draws the frame.

See "Adding Frames, Borders, and Images" on page 1482

Example "Example 1: Producing Text Slides" on page 1483

Details

Adding Frames, Borders, and Images

Like the BORDER option in a GOPTIONS statement, the BORDER option in the PROC GSLIDE statement draws a box around the graphics output area. However, the border generated by the GSLIDE procedure remains in effect only for the duration of the procedure.

Both BORDER options use the color specified by the CTITLE= or CTEXT= graphics option if either of these options is used. Otherwise, the border color is the color specified by the current style. If the NOGSTYLE option is specified, then the color is the first color in the device's color list.

The BORDER option draws a box around the graphics output area. The FRAME option draws a box or frame around the procedure output area. In this case, titles and footnotes are outside of the frame. (See “Overview” on page 79 for a description of the procedure output area.) Use the FRAME option to draw a frame in the default color, line type, and width. Otherwise, use one or more of the CFRAME=, LFRAME=, or WFRAME= options.

You can specify a colored frame with the CFRAME= option. Note that the CFRAME= option does not fill the procedure output area with color. However, you can use the CBACK= graphics option to provide a background color for the graphics output area. You can specify the type of line for the frame with the LFRAME= option and the width of the frame with the WFRAME= option.

You can also use the IFRAME= option to fill the background of your slide with an image. If an image is specified, it completely fills the background of the slide, obscuring any frame or border specifications.

Using Data-Dependent Coordinates

If you use the GSLIDE procedure with Annotate data sets that contain data-dependent coordinates, the resulting coordinate values can exceed the range of the graphics output area. The range is 0 to 100. Some of the output might not be displayed. In this case, use the GANNO procedure, which can scale the output to fit the available space. See also Chapter 33, “GANNO Procedure,” on page 887 for details.

Using RUN Groups

Although the GSLIDE procedure has no action statements, it can use RUN-group processing. This displays all currently defined titles, footnotes, notes, and specified annotation, each time you submit a RUN statement. TITLE and FOOTNOTE statements that are defined while the GSLIDE procedure is active remain in effect after the procedure ends. NOTE definitions remain in effect until the GSLIDE procedure ends, at which time they are canceled. To cancel NOTE definitions while the procedure is active, specify RESET=NOTE in a GOPTIONS statement or submit a null NOTE statement. See Chapter 7, “Using Run-Group Processing,” on page 75 for details.

Examples: GSLIDE Procedure

Example 1: Producing Text Slides

Features: PROC GSLIDE options BORDER, CFRAME=, and WFRAME=
NOTE Statement

Sample library member: GSLTEXTS

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example uses FOOTNOTE, NOTE, and TITLE statements to produce a text slide. PROC GSLIDE statement options add both a border and a frame.

Program

```
goptions reset=all cback="cx4682B4"
           border;
```

```
title color=yellow font="albany amt/bold" height=1.5 "New Directions";
footnote1 color=yellow j=l "  ABC Engineering, Inc";
footnote2 color=yellow j=l "  January 2020";

proc gslide border
    cframe=darkred
    wframe=2;
    note height=14;
    note height=5
       justify=center
       color=white
       "Goals and strategies"
       justify=center
       "for the coming year";
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all cback="cx4682B4"
           border;
```

Define titles and footnotes.

```
title color=yellow font="albany amt/bold" height=1.5 "New Directions";
footnote1 color=yellow j=l "  ABC Engineering, Inc";
footnote2 color=yellow j=l "  January 2020";
```

Generate the slide and define additional text. The BORDER option draws a box around the entire graphics output area. The CFRAME= option draws a dark red box around the procedure output area. The WFRAME= option specifies the thickness of the frame. The COLOR= option specifies the color of the note text. The first NOTE statement, which has no text, simply leaves a large blank line above the text specified by the second NOTE statement. The second JUSTIFY= option causes a line break.

```
proc gslide border
    cframe=darkred
    wframe=2;
    note height=14;
    note height=5
       justify=center
       color=white
       "Goals and strategies"
       justify=center
       "for the coming year";
run;
quit;
```

Example 2: Displaying Annotate Graphics

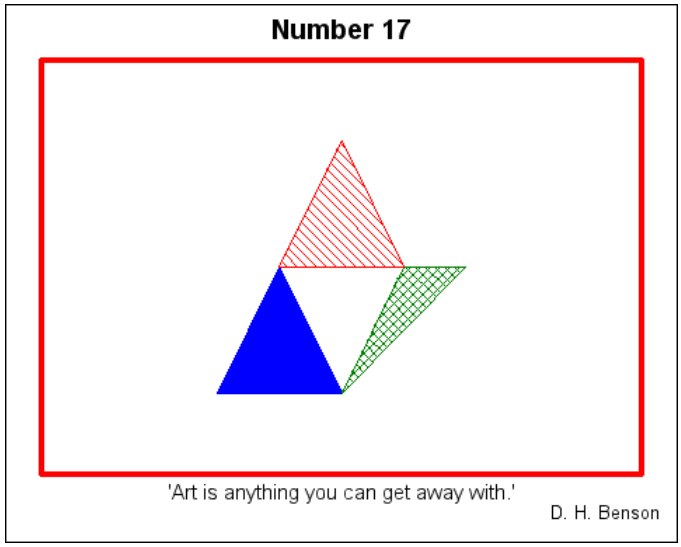
Features: PROC GSLIDE option ANNOTATE=

Annotate data set

Sample library member: GSLANNOT

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

In this example, the GSLIDE procedure displays Annotate graphics along with current TITLE and FOOTNOTE definitions. See Chapter 28, “Using Annotate Data Sets,” on page 721 for information about creating Annotate data sets.



Program

```
goptions reset=all  border;

data art;
   length function color style $ 8;
   input function $ x y color $ style $;
   xsys="5"; ysys="5";
   datalines;
poly      30  20  blue    solid
polycont  50  20  .       .
polycont  40  50  .       .
poly      50  20  green   x1
polycont  70  50  .       .
polycont  60  50  .       .
poly      40  50  red     l1
polycont  60  50  .       .
polycont  50  80  .       .
;

title "Number 17";
footnote1 h=2 "'Art is anything you can get away with.'";
footnote2 j=r "D. H. Benson   ";
```

```
footnote4 angle=90;
footnote5  angle=-90;

proc gslide annotate=art
            border
            wframe=6
            cframe=red;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all  border;
```

Create the Annotate data set, ART. ART contains the commands that draw the design of triangles.

```
data art;
   length function color style $ 8;
   input function $ x y color $ style $;
   xsys="5"; ysys="5";
   datalines;
poly      30  20  blue    solid
polycont  50  20  .       .
polycont  40  50  .       .
poly      50  20  green   x1
polycont  70  50  .       .
polycont  60  50  .       .
poly      40  50  red     l1
polycont  60  50  .       .
polycont  50  80  .       .
;
```

Define the title and footnotes displayed by the procedure. FOOTNOTE statements 4 and 5 have no text and are angled vertically to add space on the left and right sides between the border of the output and the frame that surrounds the procedure output area.

```
title "Number 17";
footnote1 h=2 "'Art is anything you can get away with.'";
footnote2 j=r "D. H. Benson   ";
footnote4 angle=90;
footnote5  angle=-90;
```

Display the annotate graphics on the slide with the title and footnotes. The GSLIDE procedure displays the graphics elements drawn by the commands in the Annotate data set specified by the ANNOTATE= option.

```
proc gslide annotate=art
            border
            wframe=6
            cframe=red;
run;
quit;
```

Chapter 46

GTILE Procedure

Overview: GTILE Procedure

What Does the GTILE Procedure Do?

The GTILE procedure creates charts that consist of rectangles or squares that are divided into tile-shaped segments. These charts are sometimes referred to as rectangular treemaps. The GTILE charts represent the relative sizes of tiles to one

another and to the whole. The GTILE procedure provides three statements that you can use to define the layout in order to visualize your data. The statements require one numeric variable. This variable defines the size of the chart tiles.

The TILEBY= statement is followed by any number of numeric or character variables that are delineated by a comma or a blank space. See “Chart Variables” on page 1488. By providing multiple TILEBY variables, and specifying the ACTIVEX device with the DEVICE= option, you can use the GTILE procedure to create interactive charts. These charts enable you to display subsets (or levels) of your data. You can assign an additional numeric variable as a color variable using the COLORVAR= option.

Concepts: GTILE Procedure

Chart Variables

The GTILE procedure produces charts based on the values of the chart's size variable, and the values of a TILEBY level variable. The chart's size variable must be numeric. All the values are treated as discrete. The sum of the chart's size variable value determines the size of each tile. The chart's size variable is also used to color each tile, unless a color variable is specified with the COLORVAL= option.

At least one TILEBY= variable is required. The values of this variable or variable list determine the tile categories, as well as the chart levels (or subsets).

The levels are visually represented in the chart by line colors and line style. The top level is indicated by a thick line and the darkest color. The next level is represented by a thinner line and the darkest color. The third level and any subsequent levels are represented by the thickest line and the darkest color. Each level is also represented in the navigation status bar in the top left corner of the chart.

Note: For Java, an indicator to the left of the legend identifies the name of the chart's size variable.

Default Tile Color

In a GTILE tile chart, each tile represents that tile’s size value relative to the other tiles and to the total size for the entire chart. Consider the following simple example.

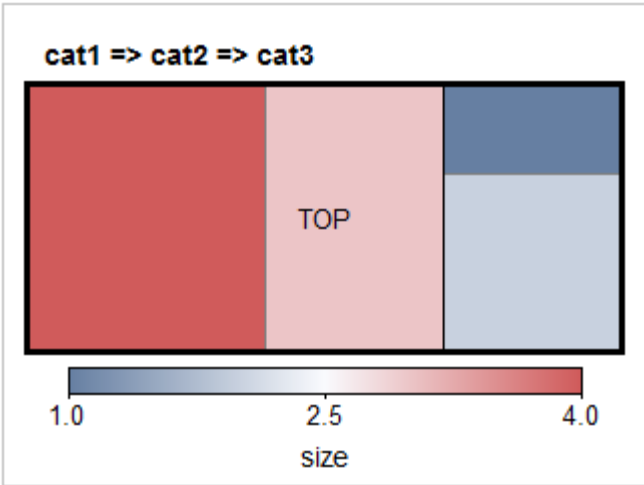
Example Code 46.1 *Simple Three-Level Tile Chart*

```
filename odsout ".";

data tiledata;
   input cat1 $1-4 cat2 $5-10 cat3 $11-13 size;
```

```
datalines;
TOP BIG   B1 4
TOP BIG   B2 3
TOP SMALL S1 2
TOP SMALL S2 1
;
run;

ods _all_ close;
goptions reset=all device=activex xpixels=320 ypixels=260 border;
ods graphics / reset=index;
ods html path=odsout file="tilechart.html";

proc gtile data=tiledata;
   tile size tileby=(cat1, cat2, cat3);
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */
```

Here is the output.

Figure 46.1 *Output for Simple Three-Level Tile Chart*



By default, the detail level is set to 3 to match the number of variables that are specified by the GTILE statement TILEBY= option. This displays all of the tile segments. The label level is set to 1, which displays the label for the top-level tile TOP. In this case, the size of each tile segment is derived directly from the data. The size value also determines the tile color. For continuous colors, the size value is used to select a color from the color ramp. The smallest tile is assigned the starting color in the color ramp, and the largest tile is assigned the ending color. The remaining tiles are assigned colors between the minimum and maximum size values sequentially in ascending order.

When discrete colors are used, by default, each tile is assigned a color from a GraphData1–GraphDataN style element based on size in ascending order. The smallest tile is assigned the color from GraphData1, the next largest tile is assigned the color from GraphData2, and so on.

Tile Summarization

When continuous colors are used, tile summarization can occur when the detail level is set to a value that is less than the levels specified in the TILEBY= option. When a tile is summarized, its segments are merged into the parent tile. The size of the summarized tile is the sum of the segment sizes. The color value of a summarized tile is a size-weighted average of the segment color values. Since the sum of the weighting values is always 1 in this case, the size-weighted color value is computed as follows:

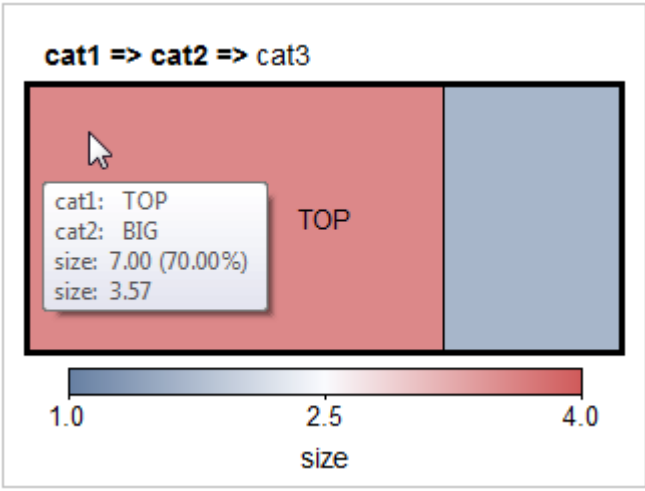
$$SummarizedColorValue = \sum_{i=1}^{n} \left(\left(\frac{SegmentSize_i}{ParentTileSize} \right) \times SegmentColorValue_i \right)$$

The resulting color provides a visual indication that the segments have been merged into one.

Note: Tile summarization does not occur when discrete colors are used.

If the detail level is set to 2 in Example Code 46.6 on page 1488, for example, each CAT2 tile color represents a size-weighted average of its CAT3 segment color values. The following figure shows the result.

Figure 46.2 *Summarization of Segments BIG and SMALL*

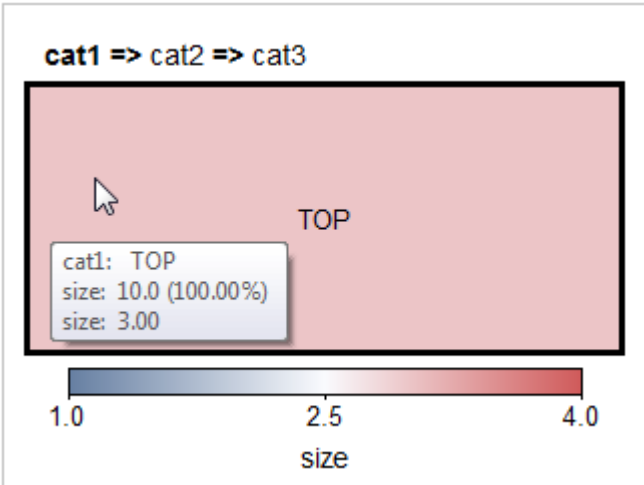


In the output, a data tip shows the results of the summarization of segment BIG. The last two items in the data tip show the tile size and color values respectively. The data tip shows that variable SIZE is assigned to the size role. The size value is the sum of the segment B1 and B2 size values 4 and 3. Because the COLORVAR= option is not specified, the data tip shows that the SIZE variable is also assigned to the color role. The color value is the size-weighted average of the segment B1 and B2 color values 4 and 3, which computes to 3.57 using the summarized color value equation shown previously.

Note: For information about how the COLORVAR= option affects the color values, see "Using a Color Variable" on page 1505.

A similar summarization occurs for segment SMALL, which results in a size of 3 and a size-weighted color value of 1.67. If the detail level is set to 1 in Example Code 46.6 on page 1488, only the top-level tile TOP is shown. It is a summarization of all of its constituent segments. The following figure shows the result.

Figure 46.3 *Summarization of Tile TOP*



Missing Values, Negative Values, and Zero Values

When the GTILE procedure finds missing values, negative values, or zero values, it does the following:

- The chart's size variable requires nonmissing, positive values to create a tile for that observation. If the value of the size variable is missing, a negative value, or a zero value, the observation is not included in the chart.
- The TILEBY= variable displays tiles with missing values, negative values, and zero values. Each tile is included in the chart, and its value is displayed on the tile and in the chart's data tip.
- The COLORVAR= variable displays missing values in a color that can be distinguished from the colors in the color ramp.

Note: For Java, an indicator to the right of the legend identifies the color assigned to missing values.

Syntax: GTILE Procedure

Restrictions: The graphics device that you can use with the GTILE procedure depends on your version of SAS. See "Specifying a Graphics Device" on page 1504.

The plot legend does not include the size variable when the ACTIVEX or ACTXIMG device is used. To include the size variable in the legend, use the JAVAIMG device.

Custom chart tips and drill-down links are not supported.

The chart legend cannot be suppressed.

For a discrete legend, binning is not supported.

For a discrete legend, user-specified legend values are not supported. The legend values are determined by the data.

Requirement: At least one FLOW, TILE, or TOGGLE statement is required.

Supported Statements: FOOTNOTE, GOPTIONS, TITLE

z/OS specifics: Starting with SAS 9.4M8, the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices are no longer supported in the z/OS operating environment. If you attempt to use one of these devices on z/OS, an error results. Because the GTILE procedure supports only the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices, starting with SAS 9.4M8, the GTILE procedure no longer runs on z/OS.

Supports: RUN-Group Processing

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, "Plotting a Cloud Analytic Services (CAS) In-Memory Table," on page 153.

The procedure can include the SAS/GRAPH statements BY on page 424 and NOTE on page 507, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See "SAS/GRAPH Statements" on page 397 and *SAS DATA Step Statements: Reference* for more information.

Tips: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, `ODS PDF; procedure-code; ODS PDF CLOSE;`). See Chapter 7, "Using Run-Group Processing," on page 75 for more information.

When the graphics output is sent to a graphics output file, this procedure uses ODS Graphics Indexing when necessary to avoid overwriting existing output files. See "About Filename Indexing" on page 137. To reset the current index value and overwrite output files that already exist, use the following statement:

```
ods graphics / reset=index;
```

PROC GTILE < DATA=*input-data-set* >;

FLOW *size-variable* TILEBY=(*levels-variable* | *levels-variable-list*) </*options*>;

TILE *size-variable* TILEBY=(*levels-variable* | *levels-variable-list*) </*options*>;

TOGGLE *size-variable* TILEBY=(*levels-variable* | *levels-variable-list*) </*options*>;

PROC GTILE Statement

Identifies the data set containing the chart variables.

Requirement: An input data set is required.

Syntax

PROC GTILE < DATA=*input-data-set* >;

Optional Argument

PROC GTILE statement options affect all graphs produced by the procedure.

DATA=*input-data-set*
specifies the SAS data set that contains the variables to chart. By default, the procedure uses the most recently created SAS data set.

See "About the Input Map Data Set" in *SAS/GRAPH and Base SAS: Mapping Reference*

"The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

FLOW, TILE, and TOGGLE Statement

Create a tile chart in one of the three display layouts.

Requirement: At least one numeric chart size variable and one TILEBY= variable is required.

Supported Statements: FOOTNOTE, GOPTIONS, TITLE

Supports: Drill-down functionality

Syntax

FLOW *size-variable*TILEBY=(*levels-variable* | *levels-variable-list*) < /*options* >;
TILE *size-variable* TILEBY=(*levels-variable* | *levels-variable-list*) < /*options* >;
TOGGLE *size-variable* TILEBY=(*levels-variable* | *levels-variable-list*) < /*options* >;

Summary of Optional Arguments

Appearance options

CDEFAULT=*missing-value-color*
: specifies the color to be assigned to tiles when the COLORVAR= value is missing.

COLORPOINTS=(*point-1 point-2 … point-n*)
: specifies where the colors need to transition on the color ramp with respect to the normalized values of the color variable.

COLORRAMP=(*color-ramp color-list*)
: specifies the colors to be distributed continuously across the range of data values.

COLORTYPE=CONTINUOUS | DISCRETE
: specifies whether a color ramp or discrete colors are used to color the tiles.

COLORVAR=*color-variable*
: specifies a variable whose values determine the color of the tiles.

DETAILLEVEL=*1-to-the-number-of-variables-specified-by-TILEBY=*
: specifies the number of levels to display.

LABELLEVEL=*1-to-the-number-of-levels-variables-specified-by-TILEBY=*
: specifies the number that corresponds to the level of labels to display.

MAXLEGENDVALUE=*max-value*
: specifies the maximum endpoint value of a continuous legend range.

MINLEGENDVALUE=*min-value*
: specifies the minimum endpoint value of a continuous legend range.

NOLEGEND
: suppresses the legend that is automatically generated.

Catalog entry description options

DESCRIPTION="*description*"
: specifies a description of the output.

NAME="*name*"
: specifies the name of any graphics output file created.

Midpoint options

BASELINE=*midpoint*
: specifies the midpoint value for the tiles.

Required Arguments

size-variable

specifies a numeric variable from the input data set. The values of this variable are used to determine the size of each tile.

Note If the COLORVAR= option is not specified, then the size variable is used to assign a color to each tile. In that case, the size variable is assigned both the size and color roles in the chart data tips.

TILEBY=(*levels-variable* | *levels-variable-list*)

specifies the variable, or a list of variables, that define the tile-segments and the chart levels. The variables can be character or numeric. The variable or

variables must be enclosed in parentheses, and you can use either commas or blank spaces as delimiters in a variable list.

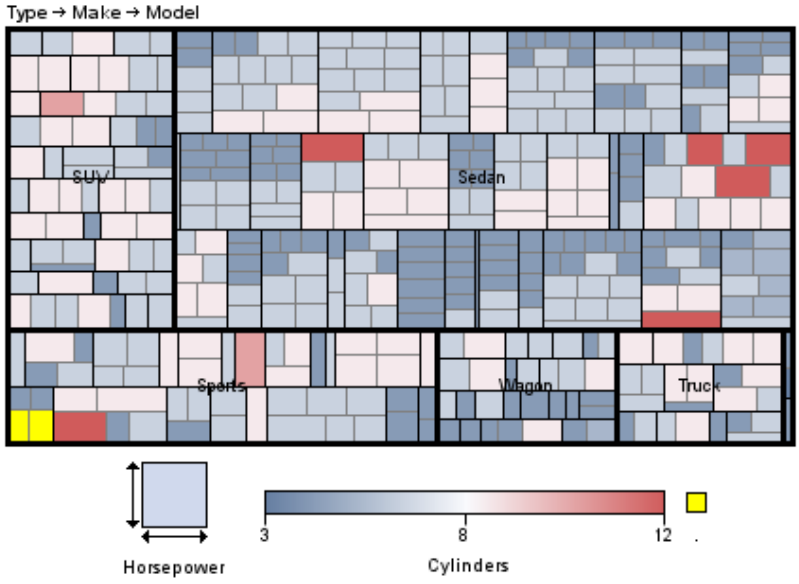
Optional Arguments

The options in a GTILE statement affect all chart levels. Specify as many options as you want and list them in any order.

BASELINE=*midpoint*
specifies the midpoint value for the tiles.

CDEFAULT=*missing-value-color*
specifies the color to be assigned to tiles when the COLORVAR= value is missing. The color used to identify missing values is represented by a missing color indicator to the right of the legend.

Figure 46.4 *CDEFAULT=yellow*



Alias CDEF=

Restriction Partially supported by ActiveX.

Interaction This option is ignored when COLORTYPE=DISCRETE.

COLORPOINTS=(*point-1 point-2 … point-n*)
specifies where the colors need to transition on the color ramp with respect to the normalized values of the color variable. The colors are specified by the COLORRAMP= option. A value must be provided for each color that is specified by the COLORRAMP= option. You can use a space or a comma to separate the values. Each transition point is specified as a percentage of the total range of the color ramp. For example, if COLORRAMP=(red green blue), COLORPOINTS=(0, 0.75 1), and the range of the color variable is 100, the color ramp begins with red at 0, transitions to green at 75, and then transitions to blue at 100. By default, the range is set to the range of the color variable. The range can be modified with the MAXLEGENDVALUE= and MINLEGENDVALUE= options.

Default The color transition points are distributed equally along the color ramp.

Range 0 to 1

Requirements The COLORRAMP= option must be specified before you can use the COLORPOINTS= option.

The number of values that are specified in the COLORPOINTS= option must exactly match the number of colors that are specified in the COLORRAMP= option.

Interactions This option is ignored when COLORTYPE=DISCRETE.

The MAXLEGENDVALUE= and MINLEGENDVALUE= options modify the range of the color ramp.

See COLORRAMP= on page 1496

MAXLEGENDVALUE= on page 1502

MINLEGENDVALUE= on page 1503

“Specifying a Custom Range for the Color Ramp” on page 1508

Example “Example 2: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range” on page 1511

COLORRAMP=(*color-ramp color-list*)

specifies the colors to be distributed continuously across the range of data values. The three-color gradient legend provides a key to the value of the colors plotted on the chart. The legend label is the variable used to color the tiles. The legend displays the variable's minimum value, the maximum value, and the midpoint value.

Note: When the COLORRAMP= option is used with the ACTIVEX device, the graph viewers cannot change the color ramp colors using the ActiveX Control Properties dialog box. In that case, the color gradient color pickers are disabled on the **Tile** tab.

Two colors are required to create a color ramp. However, the number of colors that can be provided is not limited. These values specify the minimum, and the maximum values of the color ramp. If only two colors are specified, the legend midpoint is not be labeled. The delimiter can be either blank spaces or commas. All of the color-naming schemes supported by SAS/GRAPH are valid.

Figure 46.5 COLORRAMP=(cxbcd3aa cxae9aeb)

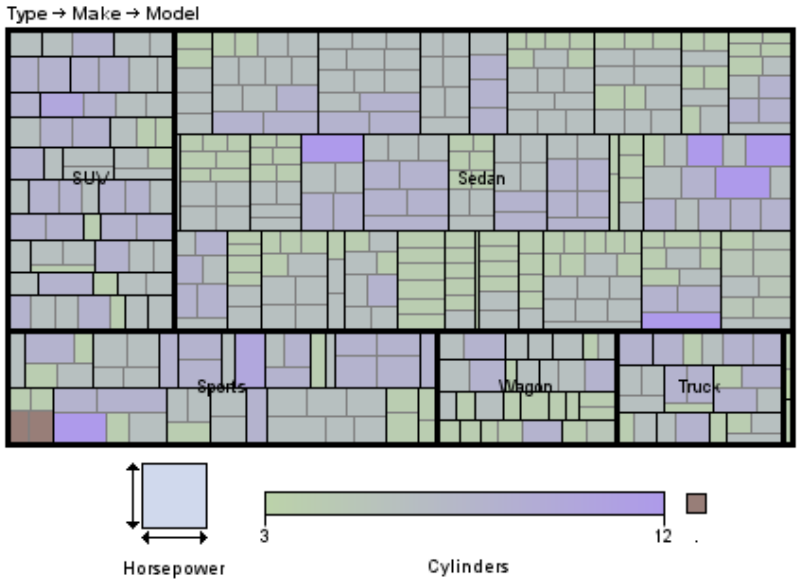
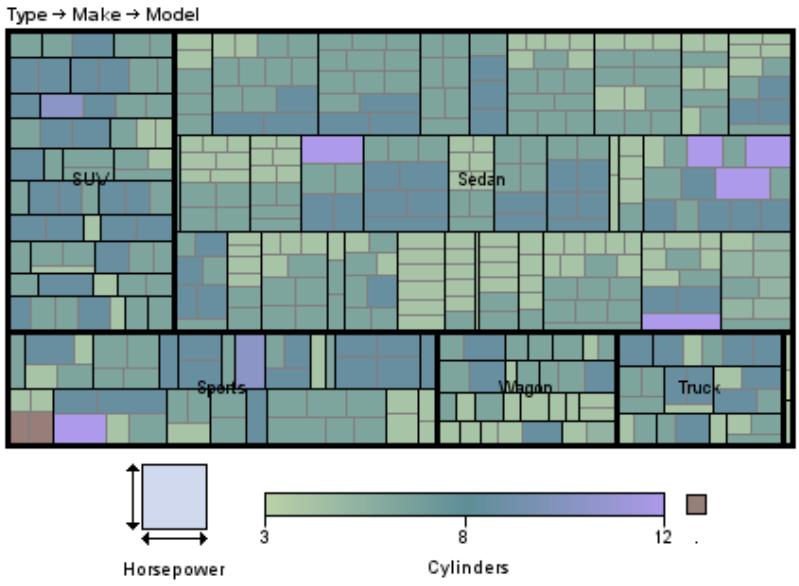


Figure 46.6 COLORRAMP=(cxbcd3aa cx5f8e97 cxae9aeb)



Alias RAMP=

Style reference Color attribute of the ThreeColorRamp element

Interaction This option is ignored when COLORTYPE=DISCRETE.

Note When the COLORRAMP= option is used to specify custom colors and the ACTIVEX device is used to generate the chart, the chart users cannot change the chart's gradient colors from the **Tile** tab of the ActiveX Control Properties dialog box. In that case, **Color** under **Color gradient** is disabled on the **Tile** tab.

COLORTYPE=CONTINUOUS | DISCRETE

specifies whether a color ramp or discrete colors are used to color the tiles. When COLORTYPE=CONTINUOUS, a color ramp is used to color the tiles. A continuous legend is used in that case to denote the tile colors. When COLORTYPE=DISCRETE, discrete colors are used to color the tiles. A discrete legend is used to denote the tile colors in that case.

Defaults

CONTINUOUS when COLORVAR= is not specified or when COLORVAR= specifies a numeric variable

DISCRETE when COLORVAR= specifies a character variable

Style references

GraphData1–GraphDataN (DISCRETE only)

the color attribute of the ThreeColorRamp element (CONTINUOUS only)

Restrictions

The legend is always positioned at the bottom of the chart. Because the LEGEND statement is not supported, the legend cannot be relocated.

When COLORTYPE=DISCRETE, you must use the COLORS= graphics option to change the chart colors from the default GraphData1 through GraphDataN colors. See "COLORS Device Parameter and Graphics Option" on page 597.

Interaction

When COLORTYPE=DISCRETE, the COLORPOINTS=, COLORRAMP=, DETAILLEVEL=, MINLEGENDVALUE=, and MAXLEGENDVALUE= options are ignored.

Note

When the COLORTYPE=DISCRETE option is used and the ACTIVEX device is used to generate the chart, the chart users cannot change the chart's gradient colors using the ActiveX control pop-up menu. **Colors** and **Center** are disabled on the **Tile** tab on the Properties dialog box in that case.

See

COLORRAMP= on page 1496

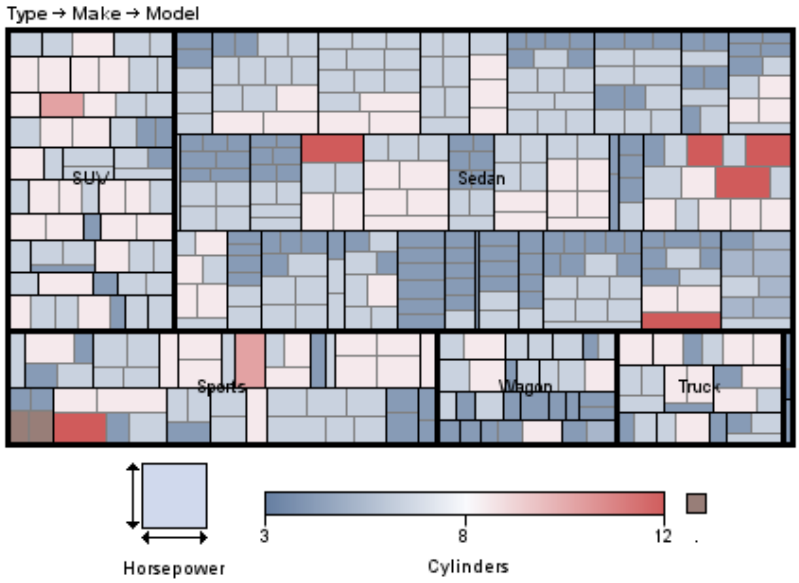
Example

"Example 3: Specifying Discrete Tile Colors" on page 1513

COLORVAR=*color-variable*

specifies a variable whose values determine the color of the tiles. The variable can be character or numeric. The values of this variable are sorted in ascending order. The smallest value is assigned to the first color in the color ramp, and the largest value is assigned the last color in the color ramp. Each of the remaining values is assigned a color from the gradient list of colors between the first, and the last colors in the color ramp.

Figure 46.7 COLORVAR=cylinders



Default The size variable is used to determine the tile colors

Note When COLORTYPE=DISCRETE and the data contains duplicate TILEBY= values, the COLORVAR= values for the duplicate TILEBY= values are dropped from the chart. However, when the ACTIVEX or ACTXIMG device is used, the dropped COLORVAR= values appear in the legend. In that case, the legend contains items that are not represented in the chart. To remove the dropped COLORVAR= values from the legend, use the JAVAIMG device instead.

See "Default Tile Color" on page 1488

"Using a Color Variable" on page 1505

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver that you are using). See "Chart Descriptions for Web Presentations" on page 251 for more information.
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description field** of the PROC GREPLAY window.

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the description text exceeds 256

characters, it is truncated to 256 characters, and then the substitution is performed.

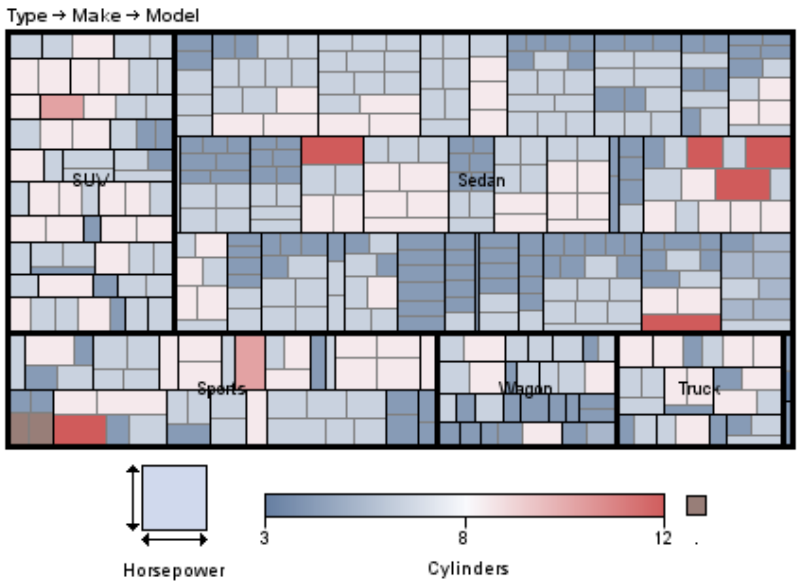
Alias DES=

Default Tile chart of *tileby*

Restriction Partially supported by ActiveX and Java.

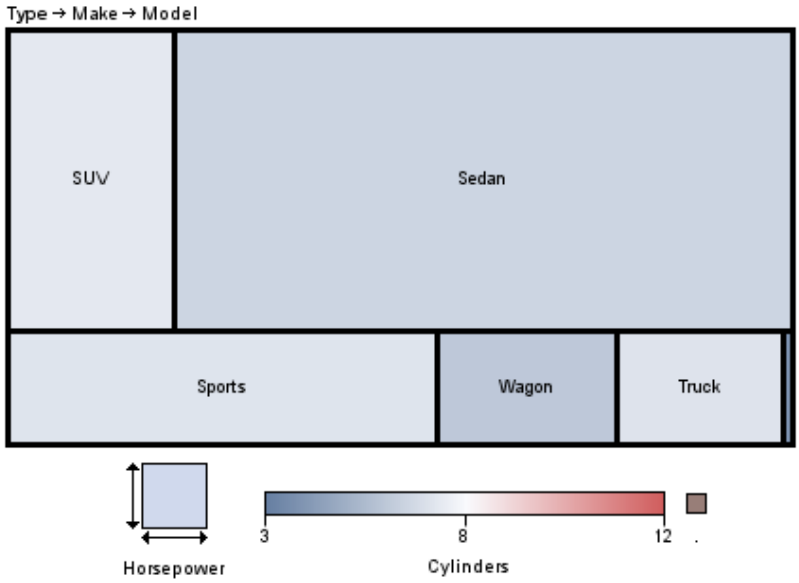
DETAILLEVEL=*1-to-the-number-of-variables-specified-by-TILEBY=*
specifies the number of levels to display. The valid values for the DETAILLEVEL= option are from one to the number of variables listed in the TILEBY= levels list. Each level has a unique outline. As you drill down through the levels, the second-level lines are thinner in weight and lighter in color. As you drill down to the third and lower levels, the outlines are the same as the top level. The levels are listed above the chart on the left. The DETAILLEVEL= option does not affect the drill-down functionality.

Figure 46.8 *DETAILLEVEL= Option Is Omitted*



If the DETAILLEVEL=1, only the first level of detail is initially displayed. Only the details of the current level are displayed at any given point in the navigation.

Figure 46.9 DETAILLEVEL=1



Alias DLEVEL=

Default The number of variables specified in TILEBY=

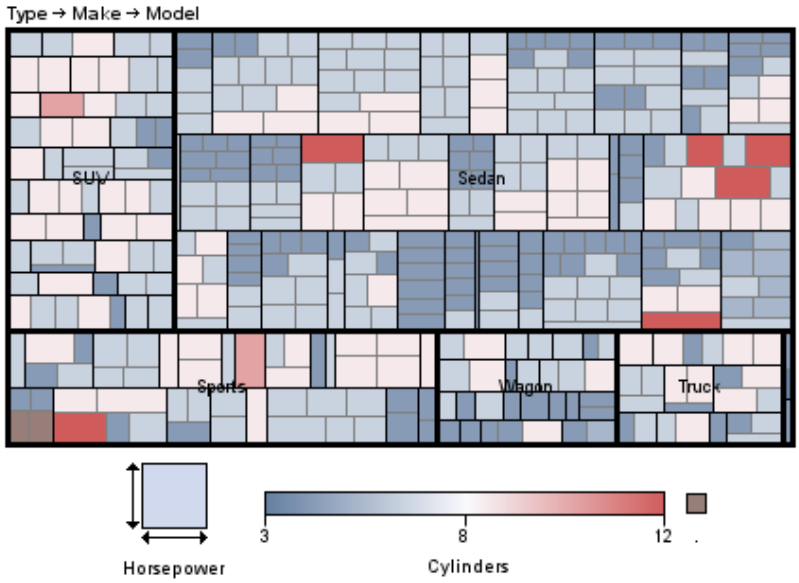
Interaction This option is ignored when COLORTYPE=DISCRETE.

Note If the DETAILLEVEL= option value is less than the number of tile-by variables and continuous colors are used, tile summarization occurs. See "Tile Summarization" on page 1490.

LABELLEVEL=*1-to-the-number-of-levels-variables-specified-by-TILEBY=*
specifies the number that corresponds to the level of labels to display. The valid values for the LABELLEVEL= option are from one to the number of variables listed in the TILEBY=levels list. The levels are listed above the chart, on the left. The LABELLEVEL= option does not affect the drill-down functionality.

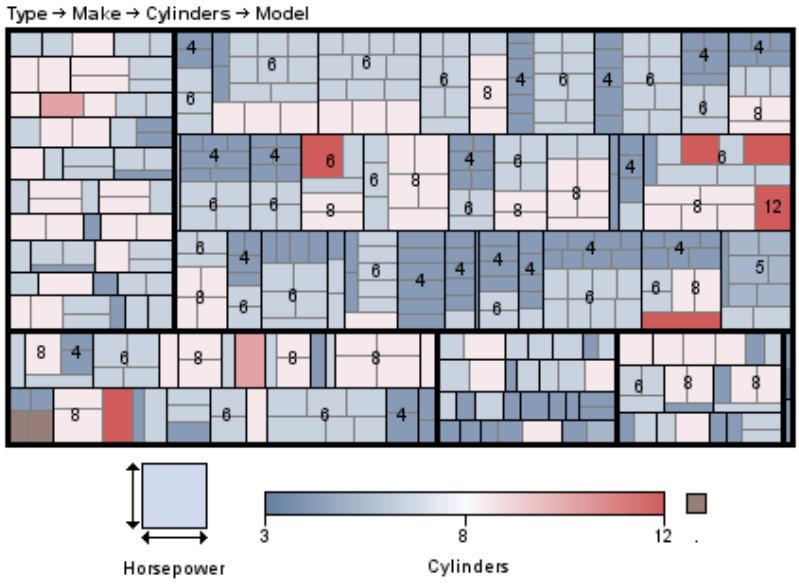
If the LABELLEVEL= option is omitted, level 1 labels are initially displayed.

Figure 46.10 *LABELLEVEL= Option Is Not Used*



If LABELLEVEL=3, the third-level labels are displayed. Once you navigate past the LABELLEVEL specified, subsequent levels display their respective labels.

Figure 46.11 *LABELLEVEL=3*



Alias LLEVEL=

Default 1

MAXLEGENDVALUE=*max-value*

specifies the maximum endpoint value of a continuous legend range. By default, the maximum legend endpoint value is set to the highest data value. The MAXLEGENDVALUE= option overrides the default value. Tiles whose data value exceeds MAXLEGENDVALUE are colored gray.

Interaction This option is ignored when COLORTYPE=DISCRETE.

See "Specifying a Custom Range for the Color Ramp" on page 1508

Example "Example 2: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range" on page 1511

MINLEGENDVALUE=*min-value*

specifies the minimum endpoint value of a continuous legend range. By default, the minimum legend endpoint value is set to the lowest data value. The MINLEGENDVALUE= option overrides the default value. Tiles whose data values are less than MINLEGENDVALUE are colored gray.

Interaction This option is ignored when COLORTYPE=DISCRETE.

See "Specifying a Custom Range for the Color Ramp" on page 1508

Example "Example 2: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range" on page 1511

NAME="*name*"

specifies the name of any graphics output file created.

The following applies to *name*:

- **For the graphics output file name:**
  - All characters are represented in lowercase.
  - The ACTXIMG or JAVAIMG device must be used in order to generate a graphics output file.
  - The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.
  - The name can include special characters.
  - Each time the graph is generated in a SAS session, SAS/GRAPH adds a number to the name, or increments the last number used to create a unique file name for the output. See "About Filename Indexing" on page 137.

Default graph.png

NOLEGEND

suppresses the legend that is automatically generated.

Note: This option is valid starting with SAS 9.4M2.

When this option is specified, the legend bar and, for graphs generated with the JAVAIMG device, the size legend, are not generated for the graph.

Default A legend is generated for the size variable or, if specified, the COLORVAR= variable.

Details

The FLOW, TILE, and TOGGLE statements specify the type of layout, the variables that define the chart levels, and the tile sizes used to display the data. One of the following display layouts is required:

FLOW
: creates a chart that honors data order and can be read from left to right.

TILE
: creates a chart that orders data by value, descending from bottom left to top right.

TOGGLE
: creates a chart that honors data order and can be read from left to right. When changing levels, the display toggles from one row to one column.

Usage: GTILE Procedure

Specifying a Graphics Device

The graphics device that you can use with the GTILE procedure depends on your version of SAS. The following table lists the graphics devices that are supported by the GTILE procedure in your version of SAS.

Table 46.1 *Graphics Devices That You Can Use with the GTILE Procedure*

SAS Release	Supported Devices	Notes
SAS 9.4M8 and later releases	JAVAIMG device only	If any other device is specified, the JAVAIMG device is used instead.[1]
SAS 9.4M5 through SAS 9.4M7, inclusive, and Viya 3.5	JAVAIMG, ACTIVEX, and ACTXIMG devices only	If any other device is specified, the JAVAIMG device is used instead.[1]
SAS 9.4M4 and prior releases	JAVAIMG, ACTIVEX, and ACTXIMG devices only	If any other device is specified, a blank graph is generated and an error message is written to the SAS log.

1 When a device change occurs, a note is written to the SAS log indicating the device change.

See “Using Graphics Devices” on page 89.

Using a Color Variable

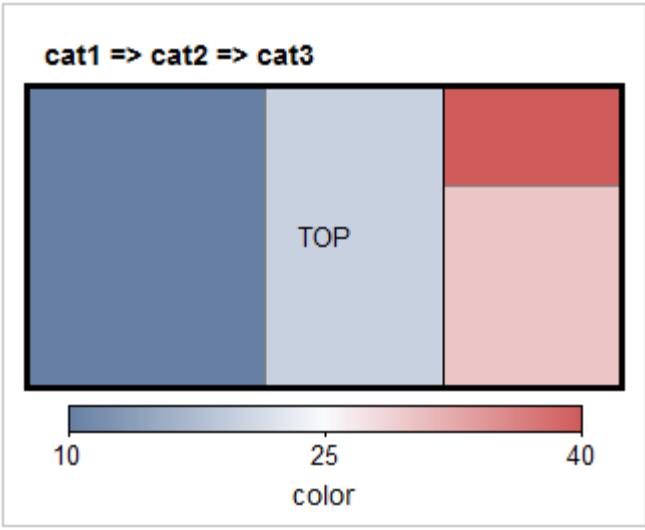
By default, the tile colors are determined by the SIZE variable values. You can use the COLORVAR= option in the GTILE statement to specify a different variable that determines the tile colors. The color variable values can be character or numeric. When the COLORVAR= option is specified, the variable values are first sorted in ascending order. For discrete values, the smallest value is assigned the GraphData1 color. The remaining values are assigned a GraphData2–GraphDataN color sequentially in ascending order. For continuous colors, the smallest value is assigned the first color in the color ramp, and the highest value is assigned the last color. The remaining colors are assigned a color from the color ramp sequentially between the minimum and maximum colors in ascending order.

Here is Example Code 46.6 on page 1488 modified to use variable COLOR as the chart color variable.

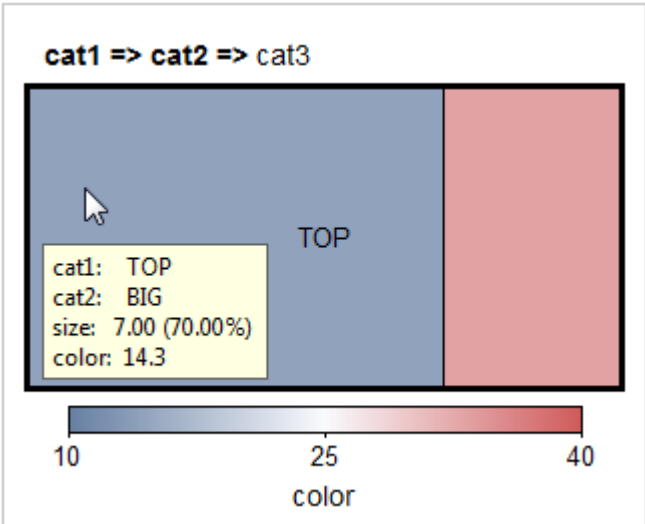
Example Code 46.2 *Three-Level Tile Chart with a Color Variable*

```
filename odsout ".";

data tiledata;
   input cat1 $1-4 cat2 $5-10 cat3 $11-13 size color;
datalines;
TOP BIG   B1 4 10
TOP BIG   B2 3 20
TOP SMALL S1 2 30
TOP SMALL S2 1 40
;
run;

ods _all_ close;
goptions reset=all device=activex xpixels=320 ypixels=260 border;
ods graphics / reset=index;
ods html path=odsout file="tilechartcolorvar.html";

proc gtile data=tiledata;
   tile size tileby=(cat1, cat2, cat3) /
      colorvar=color;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */
```

Here is the output.



As shown, the tile color values now follow the COLOR variable and are reversed from the colors used in the default case shown in Figure 46.178 on page 1489. As in the default case, for continuous colors, if tile summarization occurs, a size-weighted color value is computed as described in "Tile Summarization" on page 1490. If the detail level in Example Code 46.7 on page 1505 is changed to 2, for example, segment BIG is summarized as shown in the following figure.



In the output, the data tip for segment BIG shows that the COLOR variable is assigned the color role. Summarization for the remaining cases is similar.

Assigning Custom Colors

Custom Color Ramp Colors

When COLORTYPE=CONTINUOUS, the COLORRAMP= option enables you to customize the chart's color ramp. You can specify colors for the color ramp using any of the color-naming schemes supported by SAS/GRAPH.

Table 46.2 *Examples of Specifying Colors*

Color-Naming Scheme	Example
RGB	`COLORS=(cx98FB98 cxDDA0DD cxFFDAB9 cxDB7093 cxB0E0E6)`
RGBA[1]	`COLORS=(a98FB9833 aDDA0DD66 aFFDAB999 aDB7093CC aB0E0E6FF)`
CMYK	`COLORS=("FF00FF00" "00FFFF00" "FFFFFF00")`
HLS	`COLORS=(H14055FF H0F060FF H0B485FF H07880FF)`
HSV	`COLORS=(V0F055FF v010FFFF v03BFFFF v12C55E8)`
Gray Scale	`COLORS=(GRAY4F GRAY6D GRAY8A GRAYC3)`
SAS Registry Colors	`COLORS=(palegreen plum peachpuff palevioletred powderblue)`
CNS Color Names	`COLORS=("very light purplish blue" "light vivid green" "medium strong yellow" "dark grayish green")`

1 RGBA color mode is supported by the ACTIVEX and ACTXIMG devices when the output is used in Microsoft applications.

For information about color naming schemes, see Chapter 23, “Using Colors in SAS/GRAPH Programs,” on page 363.

The COLORPOINTS= option enables you to specify the points at which the colors transition on the color ramp. The values in the COLORPOINT= option map directly to the colors that are listed in the COLORRAMP= option. Each value specifies the transition point as a percentage of the total color ramp range. You can use the MINLEGENDVALUE= and MAXLEGENDVALUE= options to modify the legend range. See “Specifying a Custom Range for the Color Ramp” on page 1508.

Custom Discrete Colors

When COLORTYPE=DISCRETE, the COLORS= graphics option enables you to customize the chart's colors. By default, a color from the GraphData1–GraphDataN color list is assigned to each unique value of the color variable. You can use the COLORS= graphics option to override the default colors with your own color list as shown in Table 46.99 on page 1507. For information about the COLOR= graphics option, see "COLORS Device Parameter and Graphics Option" on page 597.

Specifying a Custom Range for the Color Ramp

By default, the range for a color ramp is set to the range of the color variable. The MINLEGENDVALUE= and MAXLEGENDVALUE= options enable you to override the default range with your own range. When one or both of the MINLEGENDVALUE= and MAXLEGENDVALUE= options are used, the range is computed as shown in the following table.

Options Used	Color Ramp Range
(default)	*min-color-variable-value* to *max-color-variable-value*
MINLEGENDVALUE=*n1* and MAXLEGENDVALUE=*n2*	*n1* to *n2*
MINLEGENDVALUE=*n1*	*n1* to *max-color-variable-value*
MAXLEGENDVALUE=*n1*	*min-color-variable-value* to *n1*

If you specify a custom range and the value of the color variable falls outside of the specified range, the tile or tiles associated with the out-of-range color value are colored gray.

Note: The out-of-range tile color is always gray and cannot be changed.

Examples: GTILE Procedure

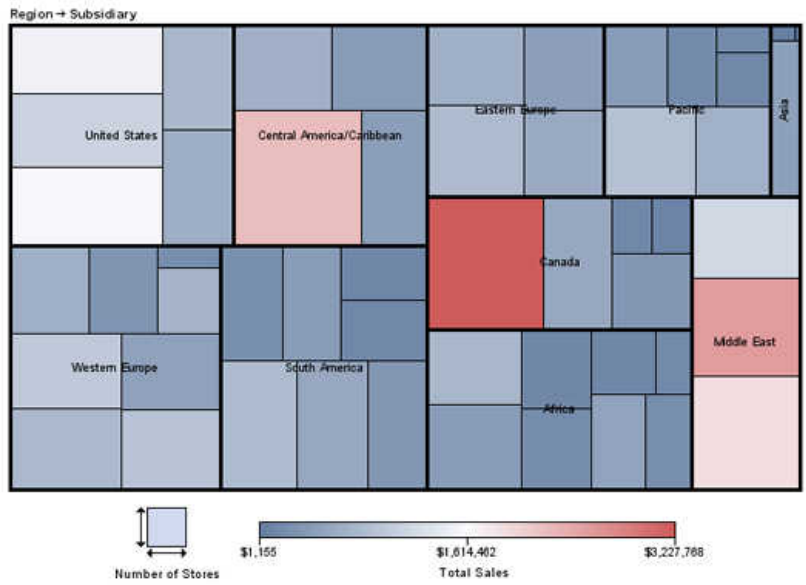
Example 1: Simple GTILE with the COLORVAR= Option

Features: TILE statement options TILEBY= and COLORVAR=

Data set: SASHELP.SHOES

In this example, PROC GTILE generates the following chart for the SASHELP.SHOES data set. The size of each tile represents the number of stores. The COLORVAR=SALES option specifies that the color of each tile represents the sales revenue for that tile. The visualization of the data with the GTILE procedure makes it easy to see the data extremes for sales revenue relative to the number of stores.

Output 46.1 *GTILE Chart of SASHELP.SHOES*



Program

```
filename odsout ".";
ods _all_ close;
```

```
goptions reset=all device=javaimg noborder;

ods html path=odsout file='shoe_sales.html';

proc gtile data=sashelp.shoes;
  tile stores tileby=(region subsidiary)
  / colorvar=sales ;
run;

quit;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Here is a detailed description of this program.

Create a file reference for the output. The current working directory is specified in this example.

```
filename odsout ".";
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Set the graphics options.

```
goptions reset=all device=javaimg noborder;
```

Open ODS HTML. The PATH= option specifies the file reference that was defined previously. The FILE= option specifies a name for the HTML output file.

```
ods html path=odsout file='shoe_sales.html';
```

Generate the tile chart using SALES as the color variable. The chart variable STORES specifies the size of the tiles. The TILE layout arranges the tiles. The TILEBY=*(levels-list)* variable list defines the tile segments and the chart levels. The COLORVAR=*SALES* option specifies the variable to use to color the tiles.

```
proc gtile data=sashelp.shoes;
  tile stores tileby=(region subsidiary)
  / colorvar=sales ;
run;

quit;
```

Close ODS HTML. Closing ODS HTML closes the HTML output file.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```
ods html; /* Not required in SAS Studio */
```

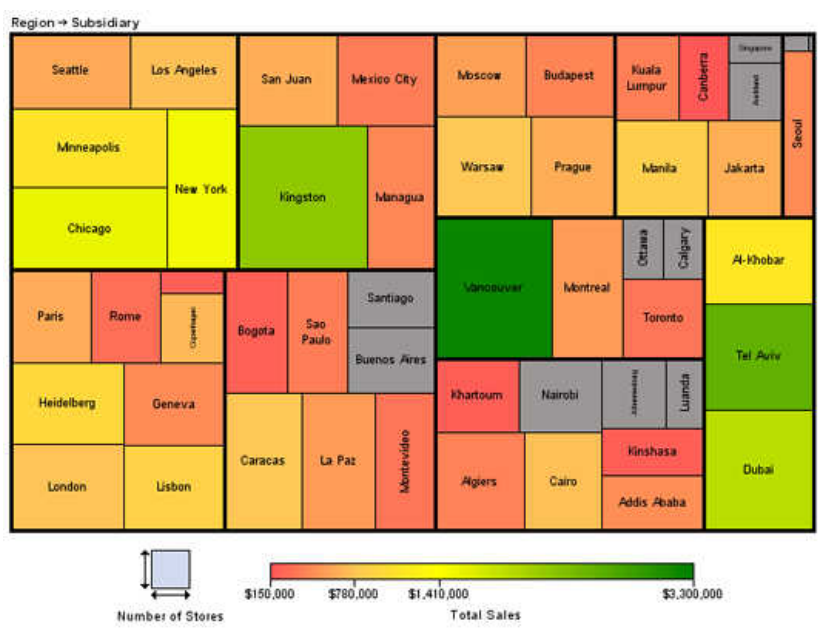
Example 2: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range

Features: TILE statement options
- COLORVAR=
- COLORRAMP=
- COLORPOINTS=
- MINLEGENDVALUE=
- MAXLEGENDVALUE=
- LABELLEVEL=

Data set: SASHELP.SHOES

In this example, PROC GTILE generates the chart shown in Output 46.34 for SASHELP.SHOES. The TILE statement defines the layout of the data. The tile sizes are determined by the number of stores in each subsidiary of each region. The COLORVAR=SALES specifies that the color of each tile represents the sales revenue for that subsidiary in that region. The COLORRAMP= option specifies that the color ramp colors consist of light red, light orange, medium yellow, and green. The COLORPOINTS= option specifies that the colors on the color ramp start at 0, and transition at 20%, 40%, and 100% of the legend value range. The MINLEGENDVALUE= and MAXLEGENDVALUE= options specify a custom legend range of 150000 to 3300000. Tiles whose SALES value falls outside of the legend range are colored gray as shown.

Output 46.2 *Tile Chart with Custom Color Points and a Custom Color Ramp Range*



Program

```
filename odsout ".";

ods _all_ close;

ods html path=odsout file="colorpoints.html";

goptions reset=all noborder device=javaimg border;

proc gtile data=sashelp.shoes;
  tile stores tileby=(region subsidiary) /
     colorvar=sales

     colorramp=(lightred lightorange mediumyellow green)
     colorpoints=(0 0.2 0.4 1)

     minlegendvalue=150000
     maxlegendvalue=3300000

     labellevel=2;
run;
quit;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Here is a detailed description of the SAS program.

Create a file reference for the output. The current working directory is specified in this example.

```
filename odsout ".";
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Open ODS HTML. The PATH= option specifies the output file reference that was defined previously. The FILE= option specifies a name for the HTML output file.

```
ods html path=odsout file="colorpoints.html";
```

Specify the graphics options. Remove the border around the chart and specify the JAVAIMG device to create a non-interactive image.

```
goptions reset=all noborder device=javaimg border;
```

Generate the tile chart using SALES as the color variable.

```
proc gtile data=sashelp.shoes;
  tile stores tileby=(region subsidiary) /
     colorvar=sales
```

Define the color ramp and specify the color transition points. Specify the colors light red, light orange, medium yellow, and green for the color ramp colors. Set the transition points at 0%, 20%, 40%, and 100%.

```
     colorramp=(lightred lightorange mediumyellow green)
     colorpoints=(0 0.2 0.4 1)
```

Specify a custom color ramp range of 150000 to 3300000. Any tiles that fall outside of this range are colored gray.

```
        minlegendvalue=150000
        maxlegendvalue=3300000
```

Specify a label level of 2.

```
        labellevel=2;
   run;
   quit;
```

Close ODS HTML. Closing ODS HTML closes the HTML output file.

```
   ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```
   ods html; /* Not required in SAS Studio */
```

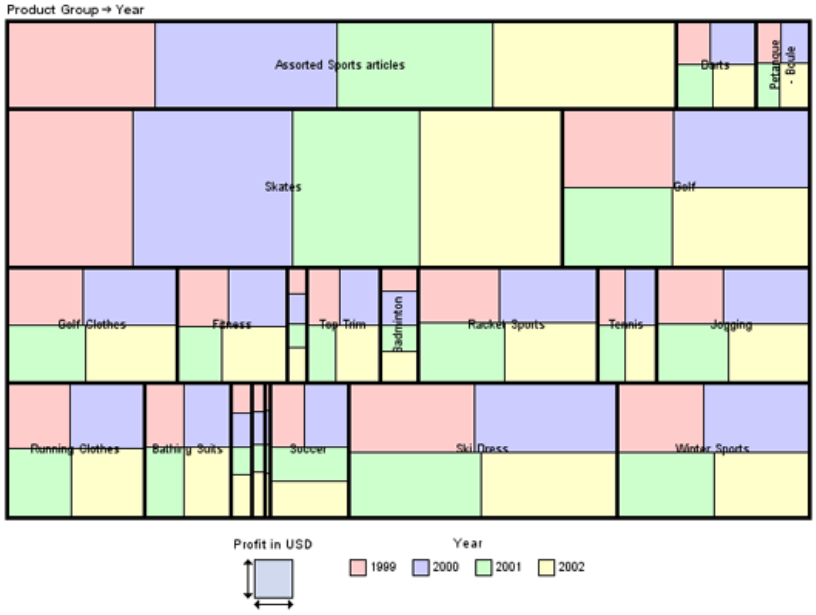
Example 3: Specifying Discrete Tile Colors

Features: FLOW statement options COLORVAR= and COLORTYPE=

Data set: SASHELP.ORSALES

In this example, PROC GTILE generates the chart shown in Output 46.35 for a subset of SASHELP.ORSALES. The FLOW statement defines the layout of the data. The tile sizes are determined by the total profits that were generated by each product group for each year. The COLORVAR=YEAR specifies that the color of each tile represents the sales revenue for that product group for that year. The COLORTYPE= option specifies that discrete colors are to be used. The GOPTIONS statement COLORS= option is used to specify that the discrete colors CXFFCCCC (light red), CXCCCCFF (light blue), CXCCFFCC (light green), and CXFFFFCC (light yellow) are to be used in the chart instead of the default GraphData1 through GraphData4 colors.

Output 46.3 *Tile Chart That Uses Discrete Tile Colors*



Program

```
filename odsout ".";

ods _all_ close;

ods html path=odsout file="discretecolors.html";

goptions reset=all border device=javaimg
   colors=(cxffcccc cxcccccff cxccffcc cxffffcc);

proc gtile data=sashelp.orsales (where=(product_line="Sports"));
    flow profit tileby=(product_group year) /
        colorvar=year
        colortype=discrete;
run;
quit;

ods html close;

ods html; /* Not required in SAS Studio */
```

Program Description

Here is a detailed description of the SAS program.

Create a file reference for the output. The current working directory is specified in this example.

```
filename odsout ".";
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Open ODS HTML. The PATH= option specifies the output file reference that was defined previously. The FILE= option specifies a name for the HTML output file.

```
ods html path=odsout file="discretecolors.html";
```

Specify the graphics options. Specify JAVAIMG as the device and specify custom chart colors.

```
goptions reset=all border device=javaimg
   colors=(cxffcccc cxccccff cxccffcc cxffffcc);
```

Generate the tile chart using YEAR as the color variable and DISCRETE as the color type.

```
proc gtile data=sashelp.orsales (where=(product_line="Sports"));
    flow profit tileby=(product_group year) /
        colorvar=year
        colortype=discrete;
run;
quit;
```

Close ODS HTML. Closing ODS HTML closes the HTML output file.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```
ods html; /* Not required in SAS Studio */
```

Chapter 47

G3D Procedure

Overview: G3D Procedure

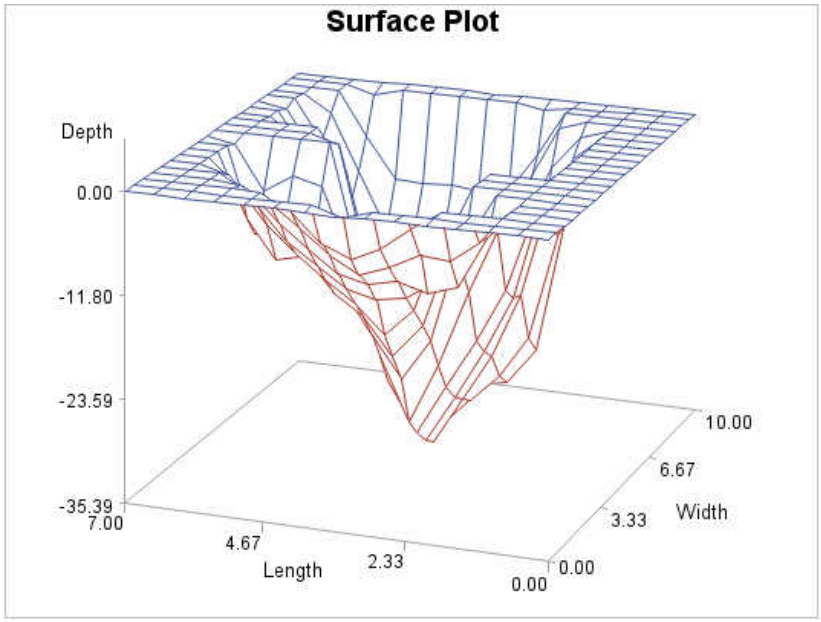
Surface Plots

Surface plots represent the shape of the surface that is described by the values of three variables, X, Y, and Z. The values of the X and Y variables are plotted to form

a horizontal plane. The values of the Z variable, create a vertical axis that is perpendicular to the X-Y plane. Combined, these three axes, form a three-dimensional surface.

The surface plot in the following figure displays various depths of a lake. The dimensions of the lake are plotted on the X-Y axes. The Z variable is plotted as the third dimension. The coordinates of each point correspond to the values of the three numeric variable values in an observation from the selected input data set.

Figure 47.1 *G3D Surface Plot*



With the PLOT statement, you can do the following actions:

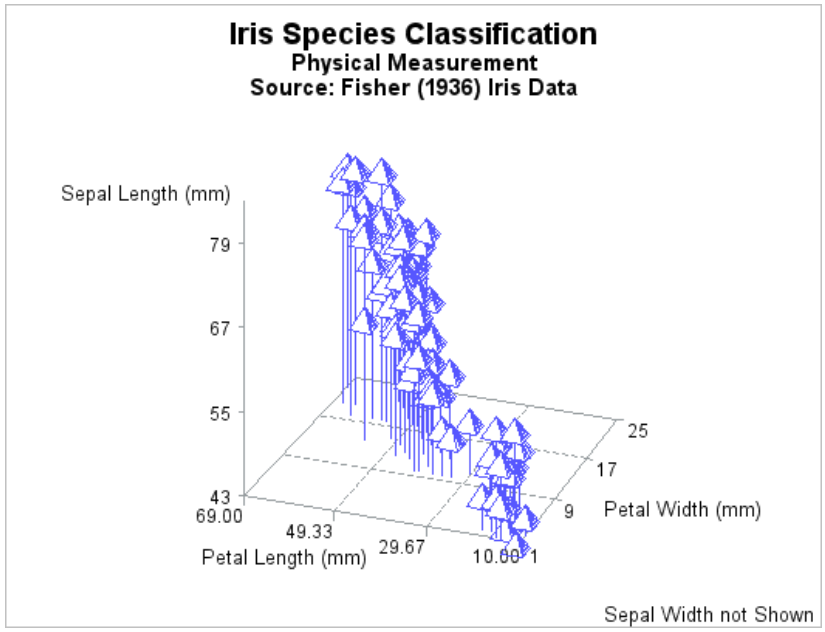
- show the three-dimensional shape of your data (useful for examining data trends).
- change the data ranges that are displayed.
- rotate and tilt the plot to enhance viewing angles.
- customize the axes.

The program for this figure is featured in "Example 1: Generating a Surface Plot" on page 1541. For more information about producing surface plots, see the "PLOT Statement" on page 1524.

Scatter Plots

Scatter plots represent the data as points. As with surface plots, the values of the X and Y variables are plotted to form a horizontal plane. The values of the Z variable create a vertical axis that is perpendicular to the X-Y horizontal plane. The values of the Z variable are represented as individual symbols. By default, these symbols are connected to the horizontal plane with lines, referred to as *needles*.

Figure 47.2 G3D Scatter Plot



With the SCATTER statement, you can do the following actions:

- change the symbols used to represent your data points
- categorize your data with colors, shapes, sizes
- change the data ranges that are displayed
- rotate and tilt the plot to enhance the viewing angles
- customize the axes

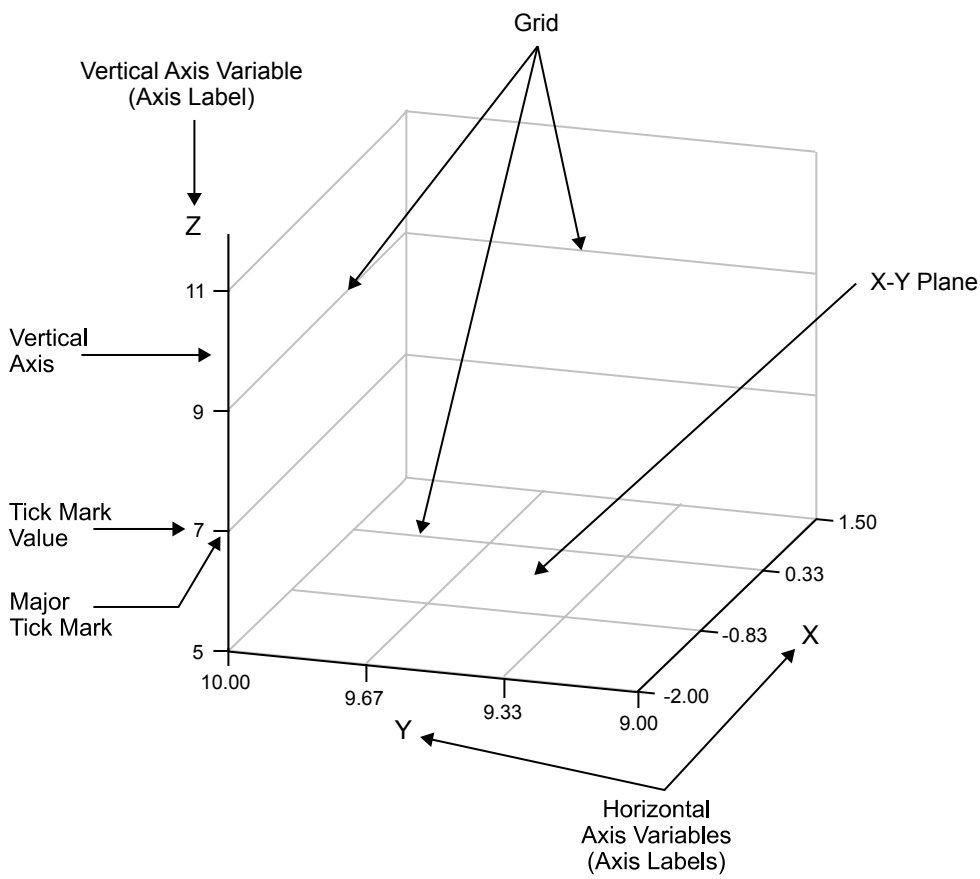
For more information about producing scatter plots, see the "SCATTER Statement" on page 1533.

Concepts: G3D Procedure

G3D Procedure Terms

The following illustration provides the terminology used to describe the elements of the three-dimensional plots generated with G3D.

Figure 47.3 Elements of a Three-Dimensional Plot



The Input Data Set

About the G3D Data Set

The G3D procedure requires three numeric variables to produce a plot. The input data set forms a rectangular grid from the values of X and Y. One value of Z is required for each X-Y grid location. If multiple observations have the same Z value for any X-Y combination, only the last observation is plotted.

Note: The Java and ActiveX drivers support multiple points with identical X-Y combinations.

Data for Surface Plots

The G3D procedure requires nonmissing Z values for at least 50% of the grid cells. When the procedure cannot produce a satisfactory surface plot because of missing Z values, a warning message is issued and a graph might not be produced. To correct this problem, you can grid the data set with the G3GRID procedure. The G3GRID procedure interpolates the necessary values to produce a data set with nonmissing Z values for every X-Y combination. The G3GRID procedure can also smooth data for use with the G3D procedure. The output data set produced by the G3GRID procedure can be used as the input data set for the G3D procedure. See Chapter 48, "G3GRID Procedure," on page 1555 for more information.

Data for Scatter Plots

In order to properly scale the axes, the G3D procedure requires at least two observations. These observations must contain unique values for each of the three variables that are specified in the plot request. If these requirements are not met, an error message is issued, and a graph is not produced.

Changing Data Ranges

For both surface plots and scatter plots, the range of the Z axis is defined by the minimum and maximum data values for Z. To increase or decrease the range of the Z axis, you can use the ZMIN= option and the ZMAX= option in the PLOT or SCATTER statements. To restrict the range of an X axis or a Y axis, you can use a WHERE clause in the PROC step to subset the data. A DATA Step with a WHERE clause, or an IF statement can also be used to subset the data.

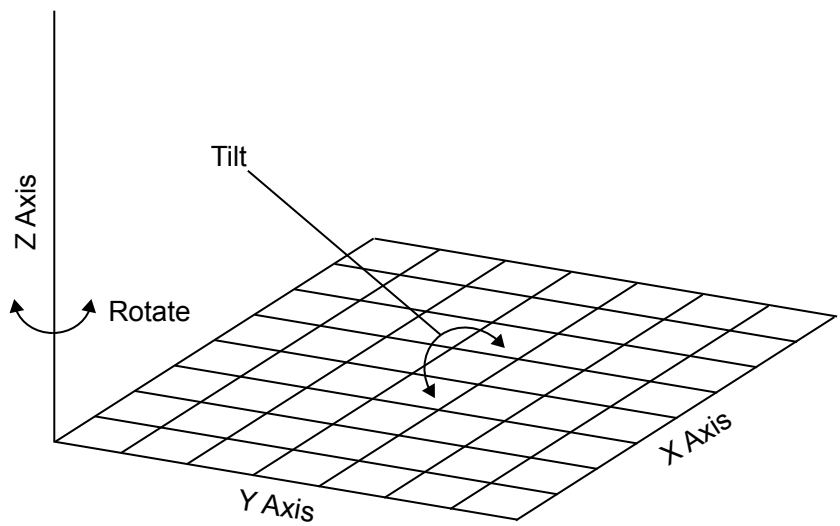
Note: See "SCATTER Statement" on page 1533 for information about controlling axis labels and tick mark values with SCATTER statement options.

Rotating and Tilting the Plot

For both surface plots and scatter plots, you can rotate the X-Y plane around the Z axis or tilt the X-Y plane toward you. When you rotate a plot, you can view data from any angle around the three-dimensional graph. Rotating a plot is useful for bringing into view data points that might be obscured by other data points. Tilting a plot enables you to accentuate the location of data points.

The following diagram illustrates how the TILT= option, and the ROTATE= option change the viewing angles of a plot.

Figure 47.4 Rotating and Tilting a Plot



Note: Certain combinations of the TILT= option, and the ROTATE= option can cause the tick mark values to overlap.

Controlling the Axes

Because the relationship between a plot's surface and the actual data values can be difficult to interpret, the readability of the plot can be enhanced by changing the number of tick marks on the axes or restricting the vertical axis range.

The G3D procedure supports AXIS definitions for Java and ActiveX only. However, you can use the functionality of PLOT and SCATTER statements to do the following:

- suppress the axes
- suppress axis labels
- suppress tick mark values
- specify the number of tick marks
- specify minimum and maximum values for the *Z* axis
- specify whether grid lines connect axis tick marks

The font and height of the graph's text can be changed with the GOPTIONS FTEXT= option and the GOPTIONS HTEXT= option, respectively. The GOPTIONS FBY= option can be used to specify the font for the BY-labels for BY-group graphs.

Syntax: G3D Procedure

Restriction: The AXIS statement is supported by the Java and ActiveX devices only.

Requirement: At least one PLOT or SCATTER statement is required.

Supported Statements: AXIS, FOOTNOTE, GOPTIONS, TITLE

Supports: BY-group processing

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 153.

When plotting a large data set with the G3D procedure, SAS might not be able to allocate enough memory to process the data. In that case, the G3D procedure step terminates, and an error message is written to the SAS log. To correct the problem, increase the amount of memory that is available to SAS or reduce the data through subsetting or some other means.

When the ACTIVEX device is used to generate the graph, the graph data tips show the interpolated values rather than the actual data values.

The procedure can include the SAS/GRAPH statements BY on page 424 and NOTE on page 507, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See “SAS/GRAPH Statements” on page 397 and *SAS DATA Step Statements: Reference* for more information.

PROC G3D <DATA=*input-data-set*>
<ANNOTATE=*annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>;
PLOT *plot-request* </*options*>;
SCATTER *plot-request* </*options*>;

PROC G3D Statement

Can identify the data set that contains the plot variables. Can also specify an annotate data set and an output catalog.

Syntax

PROC G3D <DATA=*input-data-set*>
<ANNOTATE=*annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
;

Optional Arguments

PROC G3D statement options affect all graphs produced by the procedure.

ANNOTATE=*annotate-data-set*
specifies an annotate data set to annotate all of the graphs that are produced by the G3D procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

See Chapter 28, “Using Annotate Data Sets,” on page 721

DATA=*input-data-set*
specifies the SAS data set that contains the variables to plot. By default, the G3D procedure uses the most recently created SAS data set.

See “The SAS Data Set: Your Key to the SAS System” in *Step-by-Step Programming with Base SAS*

“About Data Set Options” in *SAS Data Set Options: Reference*

“The Input Data Set” on page 1520

GOUT=<*libref.*>*output-catalog*
specifies the SAS catalog in which to save the graphics output that is produced by the G3D procedure. If you omit the libref, the output is placed in the temporary catalog WORK.GSEG. The temporary output catalog is created if it does not already exist.

See Chapter 11, “SAS/GRAPH Output,” on page 119

PLOT Statement

Creates three-dimensional surface plots using values of three numeric variables from the input data set.

Restriction:	The AXIS statement is partially supported by the Java and ActiveX devices only.
Requirement:	One plot request is required.
Supported Statements:	AXIS, BY, FOOTNOTE, GOPTIONS, NOTE, TITLE
Note:	The procedure can include the FORMAT, LABEL, and WHERE statements.

Syntax

PLOT *y*x=z* <*/options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE=*annotate-data-set*
: specifies an annotate data set to annotate plots that are produced by the PLOT statement.

CBOTTOM=*bottom-surface-color*
: specifies a color for the bottom of the plot surface.

CTOP=*top-surface-color*
: specifies a color for the top of the plot surface.

ROTATE=*angle-list*
: specifies one or more angles at which to rotate the X-Y plane around the perpendicular Z axis.

SIDE
: produces a surface graph with a side wall.

TILT=*angle-list*
: specifies one or more angles to tilt the graph toward you.

XYTYPE=0 | 1 | 2 | 3
: specifies the direction of lines that are used to represent the plot's surface.

Axes options

CAXIS=*axis-color*
: specifies a color for all the axes lines and tick marks.

CTEXT=*text-color*
: specifies a color for the axis labels and axis tick mark values.

GRID
: draws reference lines at the major tick marks on all axes.

NOAXIS
: specifies that the plot has no axes, axes labels, or tick mark values.

NOLABEL
: specifies that the plot has no axis labels or tick mark values.

XAXIS=AXIS*n*
: assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

XTICKNUM=*number-of-major-tick–marks*
: specifies the number of major tick marks that are located on a plot's *x* axis.

YAXIS=AXIS*n*
: assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

YTICKNUM=*number-of-major-tick-marks*
: specifies the number of major tick marks that are located on a plot's Y axis.

ZAXIS=AXIS*n*
: assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

ZMAX=*maximum-axis-value*
: specifies the maximum value that is displayed on a plot's Z axis.

ZMIN=*minimum-axis-value*
: specifies the minimum value that is displayed on a plot's Z axis.

ZTICKNUM=*number-of-major-tick-marks*

specifies the number of major tick marks that are located on a plot's Z axis.

Catalog entry description options

DESCRIPTION="*description*"
specifies a description of the output.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Required Argument

y*x=z;
specifies the three numeric variables from the input data set:

Y
is the horizontal variable whose values are plotted on the Y axis

X
is the horizontal variable whose values are plotted on the X axis

Z
is the vertical variable whose values are plotted on the Z axis

Optional Arguments

Options in a PLOT statement affect all graphs that are produced by that statement. You can specify as many options as you want, and list them in any order.

ANNOTATE=*annotate-data-set*
specifies an annotate data set to annotate plots that are produced by the PLOT statement.

Alias ANNO=

See Chapter 28, “Using Annotate Data Sets,” on page 721

CAXIS=*axis-color*
specifies a color for all the axes lines and tick marks.

Style reference Color attribute of the GraphAxisLines element

Restriction The AXIS statement is partially supported by Java and ActiveX. If the AXIS statement specifies general axis colors with the COLOR= option, the CAXIS= option overrides the AXIS statement general COLOR= option.

CBOTTOM=*bottom-surface-color*
specifies a color for the bottom of the plot surface.

Style reference Color attribute of the GraphData2 element

Restriction Not supported by Java

CTEXT=*text-color*

specifies a color for the axis labels and axis tick mark values. The G3D procedure uses the first color that it finds from the following list:

- colors specified for labels and values on assigned AXIS statements, which override the CTEXT= option in the PLOT statement. (Colors specified in AXIS statements are supported by the Java and ActiveX devices only.)
- the color specified by the CTEXT= option in the PLOT statement
- the color specified by the CTEXT= option in a GOPTIONS statement
- the color specified in the current style, or the first color in the color list for all of the other devices

if the NOGSTYLE system option is specified, the CTEXT= option color is assigned as follows:

- for the Java and ActiveX devices the default color is black
- for all other devices, the first color in the device's color list

Style reference Color attribute of the GraphValueText and the GraphLabelText elements

Notes If you use a BY statement in the procedure, the color of the BY variable label is controlled by the CBY= option in the GOPTIONS statement.

For Java and ActiveX, specific text options specified in the AXIS statement override the CTEXT= option.

CTOP=*top-surface-color*

specifies a color for the top of the plot surface.

Style reference Color attribute of the GraphData1 element

Example “Example 2: Generating a Rotated Surface Plot” on page 1543

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on which device driver you are using). See “Chart Descriptions for Web Presentations” on page 251 for more information.
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 1076. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the description text exceeds 256

characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default 3D surface plot of *z* by *x* and *y*

Restriction Partially supported for ActiveX and Java

See "Substituting BY Line Values in a Text String" on page 1076

GRID

draws reference lines at the major tick marks on all axes.

Restriction Not supported by Java

Example "Example 2: Generating a Rotated Surface Plot" on page 1543

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output file name:**
  - The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
  - If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default G3D

NOAXIS
specifies that the plot has no axes, axes labels, or tick mark values. Use this option if you want to generate axis labels and tick mark values with an annotate data set, or with the AXIS statement for Java and ActiveX.

Alias NOAXES

NOLABEL
specifies that the plot has no axis labels or tick mark values. Use this option if you want to generate axis labels and tick mark values with an annotate data set, or with the AXIS statement for Java and ActiveX.

ROTATE=*angle-list*
specifies one or more angles at which to rotate the X-Y plane around the perpendicular Z axis. Specify the values in degrees. The values specified in the *angle-list* can be negative or positive. If you specify a sequence of angles, separate graphs are produced for each angle. The angles that are specified in the ROTATE= option are paired with any angles that are specified with the TILT= option. If one option contains fewer values than the other, the last value in the shorter list is paired with the remaining values in the longer list. The *angle-list* list is in one of the following forms:

- an explicit list of values: *n* <*...n*>
- a starting value and an ending value with an interval increment: *n* TO *n* <BY *increment*>
- a combination of both forms: *n* <*...n*> TO *n* <BY *increment* > <*n* <*...n*> >

Default 70 degrees

Restriction Not supported by ActiveX

Example "Example 2: Generating a Rotated Surface Plot" on page 1543

SIDE
produces a surface graph with a side wall.

Restriction Partially support by Java

Example "Example 3: Generating a Tilted Surface Plot" on page 1544

TILT=*angle-list*
specifies one or more angles to tilt the graph toward you. The values must be specified in degrees. The valid values specified in the *angle-list* are 0 through 90. To generate a sequence of graphs, specify multiple angles, a graph is generated for each angle. The angles that are specified in the TILT= option are paired with any angles that are specified in the ROTATE= option. If one option contains fewer values than the other, the last value in the shorter list is paired with the remaining values in the longer list. The *angle-list* is in one of the following forms:

- an explicit list of values: *n* <*...n*>
- a starting value and an ending value with an interval increment: *n* TO *n* <BY *increment*>
- a combination of both forms: *n* <*...n*> TO *n* <BY *increment* > <*n* <*...n*> >

Default 70 degrees

Example "Example 3: Generating a Tilted Surface Plot" on page 1544

XAXIS=AXIS*n*
assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

Restriction Partially supported by Java and ActiveX only

XTICKNUM=*number-of-major-tick–marks*
specifies the number of major tick marks that are located on a plot's *x* axis. At least two values are needed to generate the axis.

Default 4 (except Java and ActiveX are 5)

Restriction Not supported by Java and ActiveX

XYTYPE=0 | 1 | 2 | 3
specifies the direction of lines that are used to represent the plot's surface. Both X and Y are displayed by default. The valid values for the XYTYPE= option are as follows:

- XYTYPE=0 (Java and ActiveX only) No lines are displayed. The plot is displayed as a solid surface.
- XYTYPE=1 draws lines that are parallel to the X axis. The surface is displayed by using lines that represent Y-axis values.
- XYTYPE=2 draws lines that are parallel to the Y axis. The surface is displayed by using lines that represent X-axis values.
- XYTYPE=3 draws lines that are parallel to both the X and Y axes. Displays the surface by using lines that represent values for both X and Y.

Restriction Not supported by Java

Example "Changing the Surface Appearance" on page 1532

YAXIS=AXIS*n*
assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

Restriction Partially supported by Java and ActiveX only

YTICKNUM=*number-of-major-tick-marks*
specifies the number of major tick marks that are located on a plot's Y axis. At least two values are needed to generate the axis.

Default 4 (except Java and ActiveX are 5)

Restriction Not supported by Java

ZAXIS=AXIS*n*
assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

Restriction Partially supported by Java and ActiveX only

ZMAX=*maximum-axis-value*
specifies the maximum value that is displayed on a plot's Z axis. Defining the ZMAX= option value greater than the data that is in the input data set, extends the plot's Z axis. Defining the ZMAX= option value less than the maximum value in the input data set displays all Z values in the range of ZMIN-to-ZMAX, and might cause data clipping.

The value of the ZMAX= option must be greater than the value of the ZMIN= option.

Default The maximum value of the *Z* variable

Restriction Not supported by Java

Example "Example 2: Generating a Rotated Surface Plot" on page 1543

ZMIN=*minimum-axis-value*
specifies the minimum value that is displayed on a plot's Z axis. Defining the ZMIN= option value less than the minimum value in the input data set extends the plot's Z axis. Defining the ZMIN= value greater than the minimum value in the input data set displays all Z values in the range of ZMIN-to-ZMAX, and might cause data clipping.

The value of the ZMIN= option must be less than the value of the ZMAX= option.

Default The minimum value of the *Z* variable

Restriction Not supported by Java

Example "Example 2: Generating a Rotated Surface Plot" on page 1543

ZTICKNUM=*number-of-major-tick-marks*
specifies the number of major tick marks that are located on a plot's Z axis. At least two values are needed to generate the axis.

Default 4 (except ActiveX is 5)

Restriction Not supported by Java

Details

Description

The PLOT statement specifies one plot request that identifies the three numeric variables to plot. The statement also does the following actions:

- scales the axes to include the minimum data values and the maximum data values for each of the plotted variables *X*, *Y*, and *Z*
- labels each axis with the name of the variable or its associated label
- derives its colors from the ODS style

In addition to the Global statement options, the following Plot statement options enable you to specify the appearance of many of the plot's elements.

Changing the Surface Appearance

The XYTYPE= option specifies the direction of the lines that form the surface plot. The following plots show examples of each type of plot surface.

Figure 47.5 *Surface Appearance for XYTYPE=1*

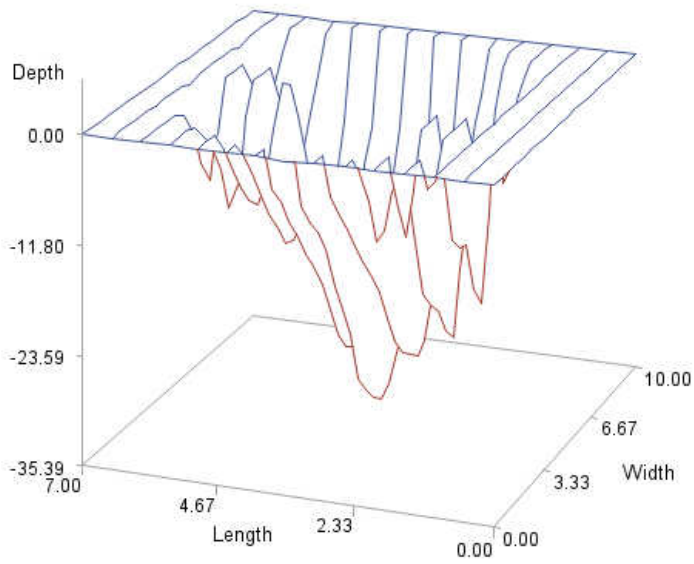


Figure 47.6 *Surface Appearance for XYTYPE=2*

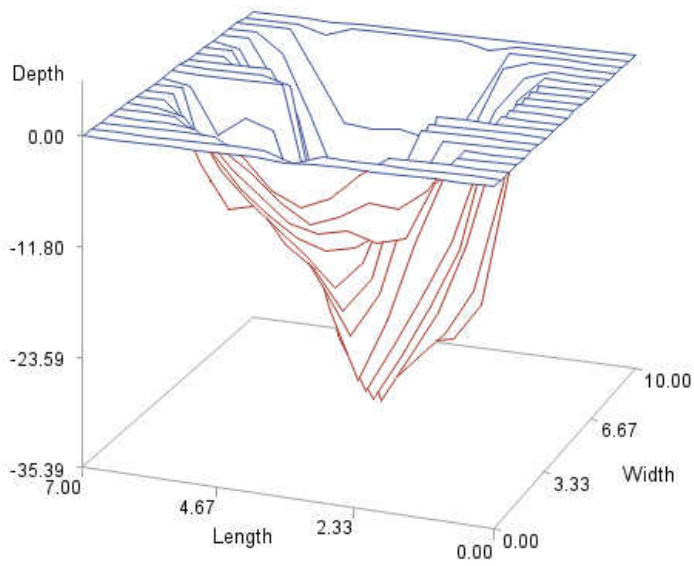
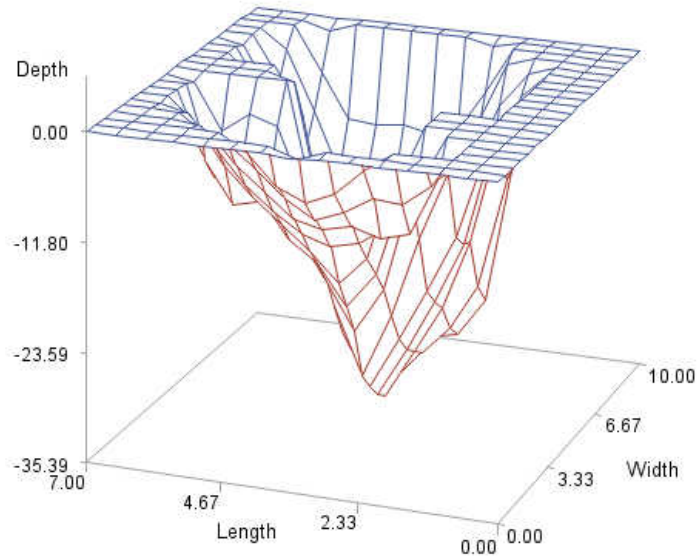


Figure 47.7 *Surface Appearance for XYTYPE=3*



SCATTER Statement

Creates three-dimensional scatter plots using values of three numeric variables from the input data set.

Alias:	SCAT
Restriction:	The AXIS statement is partially supported by Java and ActiveX devices only.
Requirement:	One plot request is required.
Supported Statements:	AXIS, BY, FOOTNOTE, GOPTIONS, NOTE, TITLE
Notes:	The procedure can include the FORMAT, LABEL, and WHERE statements. The SCATTER statement does not require a full grid of observations to generate a plot.

Syntax

SCATTER *y*x=z* <*/options*>;

Summary of Optional Arguments

Appearance options

ANNOTATE=*annotate-data-set*
: specifies an annotate data set to annotate plots that are produced by the SCATTER statement.

COLOR="*data-point-color*" | *data-point-color-variable*
specifies a color name or a character variable in the input data set whose values are color names.

NONEEDLE
specifies that a plot has no lines that connect the shapes representing data points to the X-Y plane.

ROTATE=*angle-list*
specifies one or more angles at which to rotate the X-Y plane around the perpendicular Z axis.

SHAPE="*symbol-name*" | *shape-variable*
specifies a symbol name or a character variable whose values are symbol names.

SIZE=*symbol-size* | *size-variable*
specifies either a constant or a numeric variable, the values of which determine the size of symbol shapes on the scatter plot.

TILT=*angle-list*
specifies one or more angles at which to tilt the graph toward you.

Axes options

CAXIS=*axis-color*
specifies a color for axis lines, tick marks, and horizontal grid lines.

CTEXT=*text-color*
specifies a color for all text on the axes, including tick mark values and axis labels.

GRID
draws reference lines at the major tick marks on all axes.

NOAXIS
specifies that a plot has no axes, including labels, tick marks, and values.

NOLABEL
specifies that a plot has no axes labels or tick mark values.

XAXIS=AXIS*n*
assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

XTICKNUM=*number-of-tick-marks*
specify the number of major tick marks that are located on a plot's X axis.

YAXIS=AXIS*n*
assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

YTICKNUM=*number-of-tick-marks*
specify the number of major tick marks that are located on a plot's Y axis.

ZAXIS=AXIS*n*
assigns an axis definition, where `n` is the axis-statement number, 1 to 99.

ZMAX=*maximum-value*
specify the maximum data value that is displayed on a plot's Z axis.

ZMIN=*minimum-value*
specifies the minimum value that is displayed on a plot's Z axis.

ZTICKNUM=*number-of-tick-marks*
specify the number of major tick marks that are located on a plot's Z axis.

Catalog entry description options

DESCRIPTION="*description*"
specifies the description of the plot.

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Required Argument

y*x=z;

specifies three numeric variables from the input data set:

Y
specifies a horizontal variable whose values are plotted on the Y axis

X
specifies a horizontal variable whose values are plotted on the X axis

Z
specifies a vertical variable whose values are plotted on the Z axis

Optional Arguments

Options in a SCATTER statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order.

ANNOTATE=*annotate-data-set*

specifies an annotate data set to annotate plots that are produced by the SCATTER statement.

Alias ANNO=

Restriction Partially supported by Java and ActiveX

See Chapter 28, “Using Annotate Data Sets,” on page 721

CAXIS=*axis-color*

specifies a color for axis lines, tick marks, and horizontal grid lines.

Style reference Color attribute of the GraphAxisLines element

Restriction The AXIS statement is partially supported by Java and ActiveX. When the AXIS statement specifies only general axis colors with its COLOR= option, it is overridden by the CAXIS= color option.

COLOR="*data-point-color*" | *data-point-color-variable*

specifies a color name or a character variable in the input data set whose values are color names. These color values determine the color or colors of the shapes that represent a plot's data points. Color values must be valid color names for the device that is used.

Using a list of colors in the value of the *data-point-color-variable* enables you to assign different colors to the shapes to classify data.

Style reference Color attribute of the GraphData1 element

CTEXT=*text-color*

specifies a color for all text on the axes, including tick mark values and axis labels. The G3D procedure uses the first color that it finds from the following list:

- colors specified for labels and values on assigned axis statement

- the color specified by the CTEXT= option in a SCATTER statement
- the color specified by the CTEXT= option in a GOPTIONS statement

if the NOGSTYLE system option is specified, the CTEXT= option color is assigned as follows:

- for the Java and ActiveX devices the default color is black
- for all other devices, the first color in the device's color list

Style reference Color attribute of the GraphValueText and GraphLabelText elements

Notes If you use a BY statement in the procedure, the color of the BY variable label is controlled by the CBY= option in the GOPTIONS statement.

For Java and ActiveX only, specific text options specified in the AXIS statement override the CTEXT= option.

DESCRIPTION="*description*"
specifies the description of the plot. The descriptive text is displayed as follows:

- the description in the Results window
- the properties that you view from the Explorer window
- the description in the Explorer view of the catalog entry
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the G3D output is generated while the contents page is open
- the description field of the PROC GREPLAY window
- the ALT= text in the HTML file when the output destination is ODS HTML
- customized by inserting BY variable values with #BYLINE, #BYVAL(n), and #BYVAR(n)

Alias DES=

Default 3D surface plot of *z* by *x* and *y*

See “Substituting BY Line Values in a Text String” on page 1076

GRID
draws reference lines at the major tick marks on all axes.

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.

- If the name begins with a number, an underscore is prepended to the name.
- If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

For the graphics output file name:

- The file name is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output file name. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" on page 135.
- All characters are represented in lowercase.
- If a number is added to the GRSEG name, the same number is added to the output file name. See "About Filename Indexing" on page 137.
- If the NAME= value is 8 characters or less, the file name is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the file name. When an index number is used in the GRSEG entry name, that index number is appended to the output file name. See "About Filename Indexing" on page 137.

  Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the file name.

- The maximum allowable file name length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default G3D

NOAXIS
specifies that a plot has no axes, including labels, tick marks, and values. Use this option if you want to generate axes with an annotate data set.

Alias NOAXES

NOLABEL
specifies that a plot has no axes labels or tick mark values. Use this option if you want to generate axis labels and tick mark values with an annotate data set.

NONEEDLE
specifies that a plot has no lines that connect the shapes representing data points to the X-Y plane.

Restriction The NONEEDLE option has no effect when SHAPE="PILLAR" or when SHAPE="PRISM"

ROTATE=*angle-list*
specifies one or more angles at which to rotate the X-Y plane around the perpendicular Z axis. Specify the value in degrees. The values specified in the *angle-list* can be negative or positive. The value can be greater than 360 degrees. If you specify a sequence of angles, separate graphs are produced for each angle. The angles that are specified in the ROTATE= option are paired with

any angles that are specified with the TILT= option. If one option contains fewer values than the other, the last value in the shorter list is paired with the remaining values in the longer list. The *angle-list* list is in one of the following forms:

- an explicit list of values: *n* <*...n*>
- a starting value and an ending value with an interval increment: *n* TO *n* <BY *increment*>
- a combination of both forms: *n* <*...n*> TO *n* <BY *increment* > <*n* <*...n*> >

Default 70 degrees

Restriction Not supported by ActiveX

SHAPE="*symbol-name*" | *shape-variable*

specifies a symbol name or a character variable whose values are symbol names. If you specify SHAPE=*"symbol-name"*, all data points are drawn in that shape.

If you specify SHAPE=*shape-variable*, the shape of the data point is determined by the value of the shape variable, in the input data set, for that observation. For example, the procedure uses the value of the variable CLASS for a particular observation as the shape for that data point when you specify:

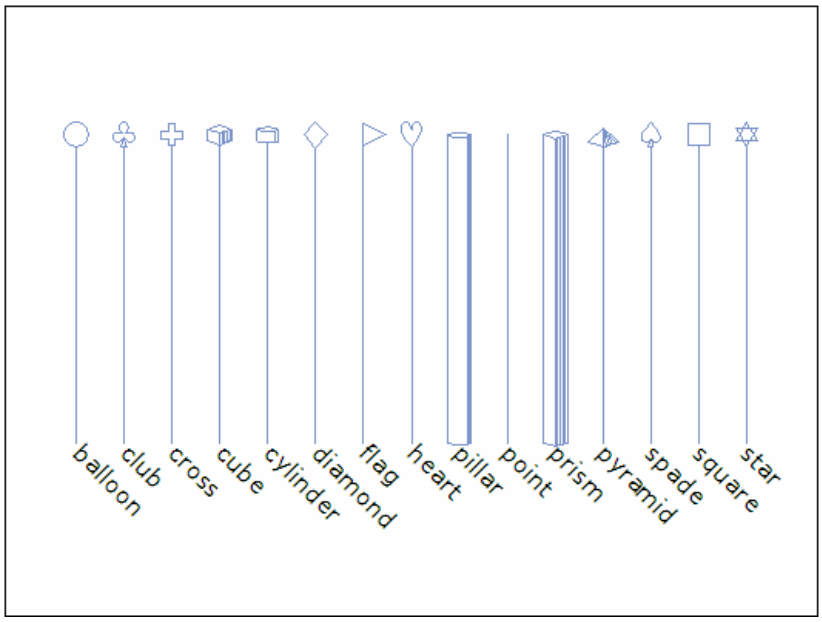
```
shape=class
```

Using a list of values in the variable named in SHAPE=*shape-variable* enables you to assign different shapes to the data points, to categorize your data.

Valid values for *symbol-name* are as follows:

- BALLOON
- CLUB
- CROSS
- CUBE
- CYLINDER
- DIAMOND
- FLAG
- HEART
- PILLAR
- POINT
- PRISM
- PYRAMID
- SPADE
- SQUARE
- STAR

Figure 47.8 Scatter Plot Symbols



Default Pyramid

Restriction These symbols might vary for Java and ActiveX

SIZE=*symbol-size* | *size-variable*

specifies either a constant or a numeric variable, the values of which determine the size of symbol shapes on the scatter plot. If you specify SIZE=*symbol-size*, all data points are drawn in that size.

If you specify SIZE=*size-variable*, the size of the data point is determined by the value of the size variable, in the input data set for that observation. For example, when you specify SIZE=CLASS, the procedure uses the value of the variable CLASS, for each observation in the input data set as the size of that data point. If you use a list of sizes as the value of the variable named in SIZE=*size-variable*, you can assign different sizes to the data points to categorize your data.

TILT=*angle-list*

specifies one or more angles at which to tilt the graph toward you. The value must be specified in degrees. The valid values specified in the *angle-list* are 0 through 90. To generate a sequence of graphs, specify different angles, and a graph is generated for each angle. The angles that are specified in the TILT= option are paired with any angles that are specified with the ROTATE= option. If one option contains fewer values than the other, the last value in the shorter list is paired with the remaining values in the longer list. The *angle-list* is in one of the following forms:

- an explicit list of values: *n* <*...n*>
- a starting value and an ending value with an interval increment: *n* TO *n* <BY *increment*>
- a combination of both forms: *n* <*...n*> TO *n* <BY *increment* > <*n* <*...n*> >

Default 70 degrees

XAXIS=AXIS*n*

assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

Restriction Partially supported by Java and ActiveX

XTICKNUM=*number-of-tick-marks*
specify the number of major tick marks that are located on a plot's X axis. At least two values are needed to generate the axis.

Default 4 (except Java and ActiveX are 5)

YAXIS=AXIS*n*
assigns an axis definition, where *n* is the axis-statement number, 1 to 99.

Restriction Partially supported by Java and ActiveX only

YTICKNUM=*number-of-tick-marks*
specify the number of major tick marks that are located on a plot's Y axis. At least two values are needed to generate the axis.

Default 4 (except Java and ActiveX are 5)

ZAXIS=AXIS*n*
assigns an axis definition, where `n` is the axis-statement number, 1 to 99.

Restriction Partially supported by Java and ActiveX

ZMAX=*maximum-value*
specify the maximum data value that is displayed on a plot's Z axis. You can use the ZMAX= option to extend the Z axis beyond the value range. The value that is specified by the ZMAX= option must be greater than that specified by the ZMIN= option. If you specify the ZMAX= option within the range of the Z variable values, the plot's data values are clipped at the level that you specified.

Default Maximum value of *Z* variable

ZMIN=*minimum-value*
specifies the minimum value that is displayed on a plot's Z axis. Defining the ZMIN= value less than the minimum value in the input data set extends the plot's Z axis. Defining the ZMIN= value greater than the minimum value in the input data set displays all Z values in the range of ZMIN-to-ZMAX, and might cause data clipping.

The value of the ZMIN= option must be less than the value of the ZMAX= option.

Default The minimum value of the *Z* variable

ZTICKNUM=*number-of-tick-marks*
specify the number of major tick marks that are located on a plot's Z axis. At least two values are needed to generate the axis.

Default 4 (except ActiveX is 5)

Details

Description

The SCATTER statement specifies one plot request that identifies the three numeric variables to plot. The statement also does the following actions:

- scales the axes to include the minimum and maximum values for each of the plotted variables *X*, *Y*, and *Z*
- labels each axis with the name of the plotted variable or its associated label
- uses reference lines to mark the major tick marks on the *X* and *Y* axes, creating a grid on the horizontal plane
- represents each data point with a pyramid that is connected to the horizontal plane with a needle
- derives its colors from the ODS style

In addition to the Global Statement options, the following Scatter statement options enable you to specify the appearance of many of the plot's elements.

Changing the Appearance of the Data Points

Use the COLOR=, SHAPE=, and SIZE= options to change the appearance of your scatter plot or to classify data using color, shape, size, or any combination of these features. Figure 47.196 on page 1539 illustrates the shape names that you can specify in the SHAPE= option. To make all of the data points red balloons at twice the normal size, use the following code:

```
scatter y*x=z /color="red" shape="balloon" size=2;
```

To size your points according to the values of the variable TYPE in your input data set, use the following code:

```
scatter y*x=z / size=type;
```

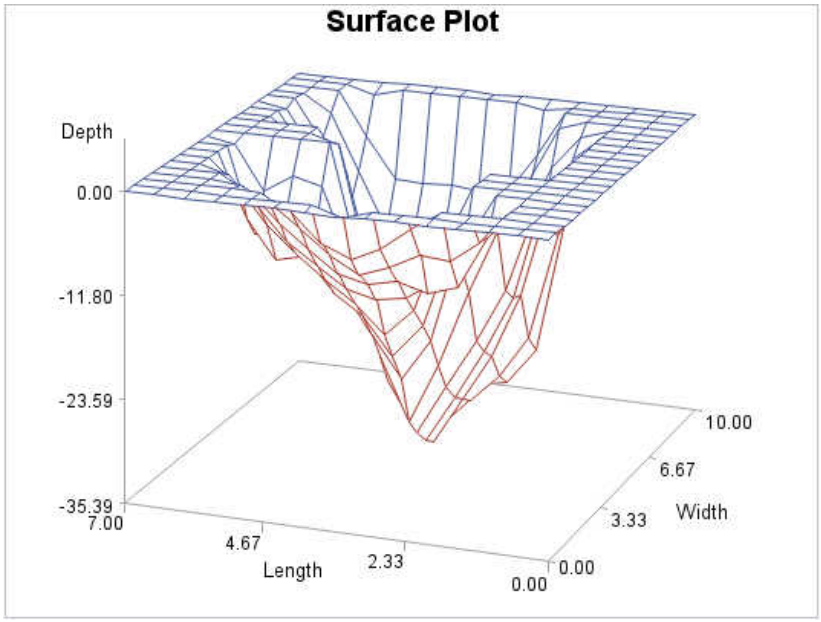
Examples: G3D Procedure

Example 1: Generating a Surface Plot

Features:	PLOT statement
Sample library member:	GTDSURFA
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form

from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This surface plot reveals the shape of a generated data set named LAKE. The axes are scaled to include all data values. Each axis is labeled with the name or label of the corresponding variable. The tick marks on the axes are divided into three even intervals. The horizontal plane is rotated 70° around the Z axis. The graph is tilted 70 degrees toward you. The colors are derived from the ODS style.



Program

```
goptions reset=all border;

title "Surface Plot";

proc g3d data=sashelp.lake;
   plot length*width=depth;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title.

```
title "Surface Plot";
```

Generate the surface plot.

```
proc g3d data=sashelp.lake;
   plot length*width=depth;
```

```
run;
quit;
```

Example 2: Generating a Rotated Surface Plot

Features: PLOT statement

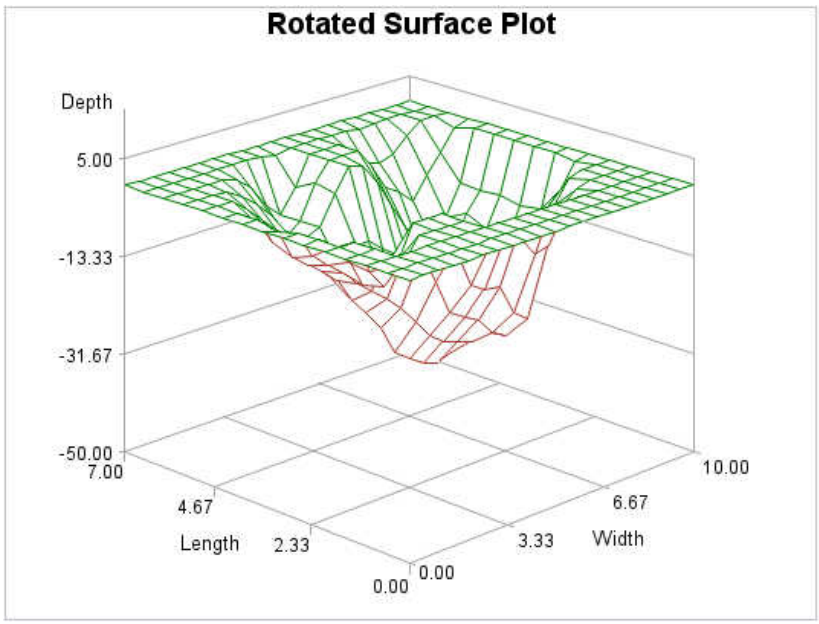
PLOT statement options
- CTOP=
- GRID
- ROTATE=
- ZMAX=
- ZMIN=

Data set: SASHELP.LAKE

Sample library member: GTDROTAT

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

The surface plot shown in this example illustrates enhancements to the axes and the presentation. The plot illustrates a grid originating from the tick marks. A Z– axis range increase raised the plot above the horizontal X-Y plane. CTOP= green changed the top color and ROTATE= rotated the plot 45 degrees toward the viewer.



Program

```
goptions reset=all border;
```

```
title "Rotated Surface Plot";

proc g3d data=sashelp.lake;
   plot length*width=depth/
      ctop=green
      grid
      rotate=45
      zmax=5
      zmin=-50;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title.

```
title "Rotated Surface Plot";
```

Generate the surface plot. CTOP=green changes the color of the plot's top surface. The GRID option draws reference lines originating from the tick marks on all the axes. The ROTATE= option specifies a rotation angle of 45°. ZMAX=5 specifies the maximum value for the Z axis. ZMIN= –50 specifies the minimum value for the Z axis. Specifying a ZMIN= value that is below the minimum value in the input data set raises the plot above the horizontal plane. Data is not displayed if it exceeds the range specified by the ZMIN= and ZMAX= options.

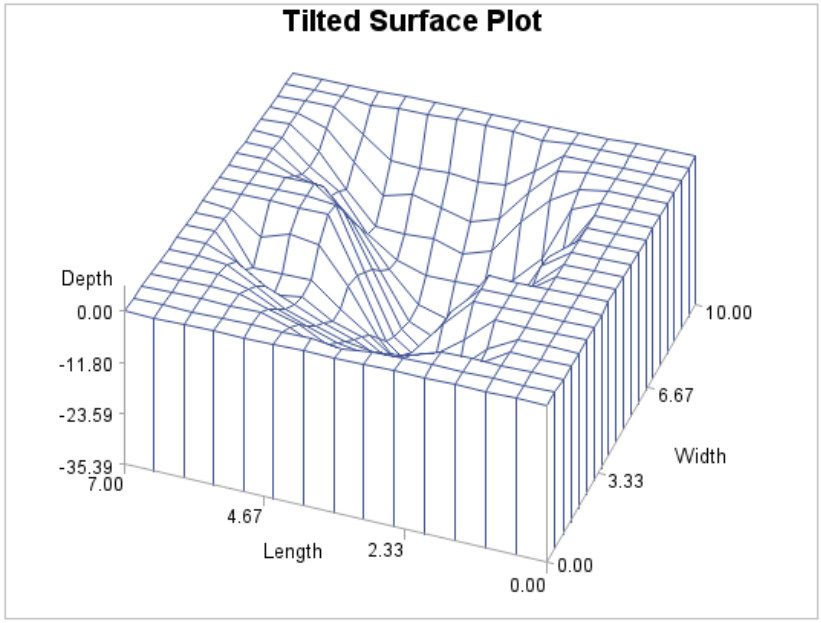
```
proc g3d data=sashelp.lake;
   plot length*width=depth/
      ctop=green
      grid
      rotate=45
      zmax=5
      zmin=-50;
run;
quit;
```

Example 3: Generating a Tilted Surface Plot

Features:	PLOT statement PLOT statement options SIDE TILT=
Data set:	SASHELP.LAKE
Sample library member:	GTDTILT
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form

from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

Simple modifications displayed in "Example 1: Generating a Surface Plot" on page 1541 are generated by tilting the surface plot 30 degrees toward you, and adding a side wall.



Program

```
goptions reset=all border;

title "Tilted Surface Plot";

proc g3d data=sashelp.lake;
   plot length*width=depth/
      side
      tilt=30;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define title.

```
title "Tilted Surface Plot";
```

Generate the surface plot. The SIDE option draws a side wall for the graph. The TILT= option specifies a tilt angle of 15° for the plot. The initial rotation of 70° is not affected by the TILT= option.

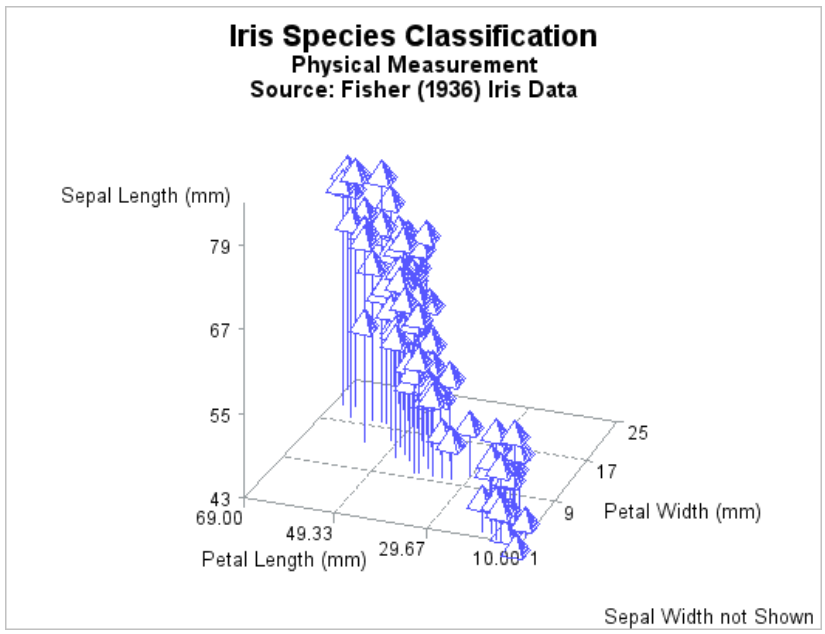
```
proc g3d data=sashelp.lake;
   plot length*width=depth/
      side
      tilt=30;
run;
quit;
```

Example 4: Generating a Scatter Plot

Features: SCATTER statement

SCATTER statement options
- COLOR=
- SIZE=

This scatter plot examines the results of measuring the petal length, petal width, and sepal length for the flowers of three species of irises. The Scatter statement in this example relies on the procedure defaults to scale the axes to include all the data values, label the axes with the variable's labels, divide the axes into three even intervals, rotate the horizontal plane 70 degrees around the vertical axis, tilt the plot 70 degrees toward you, and display the plot with the default ODS style.



Program

```
goptions reset=all border;

title1 "Iris Species Classification";
title2 "Physical Measurement";
title3 "Source: Fisher (1936) Iris Data";
footnote1 j=r "Sepal Width not Shown";
```

```
proc g3d data=sashelp.iris;
  scatter PetalLength*PetalWidth=SepalLength /
     color="light blue"
     size=1.5;

run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles and footnote.

```
title1 "Iris Species Classification";
title2 "Physical Measurement";
title3 "Source: Fisher (1936) Iris Data";
footnote1 j=r "Sepal Width not Shown";
```

Generate the surface plot. Set the marker color to light blue and the marker size to 1.5 cells.

```
proc g3d data=sashelp.iris;
  scatter PetalLength*PetalWidth=SepalLength /
     color="light blue"
     size=1.5;

run;
quit;
```

Example 5: Generating a Scatter Plot with Modified Shapes

Features:

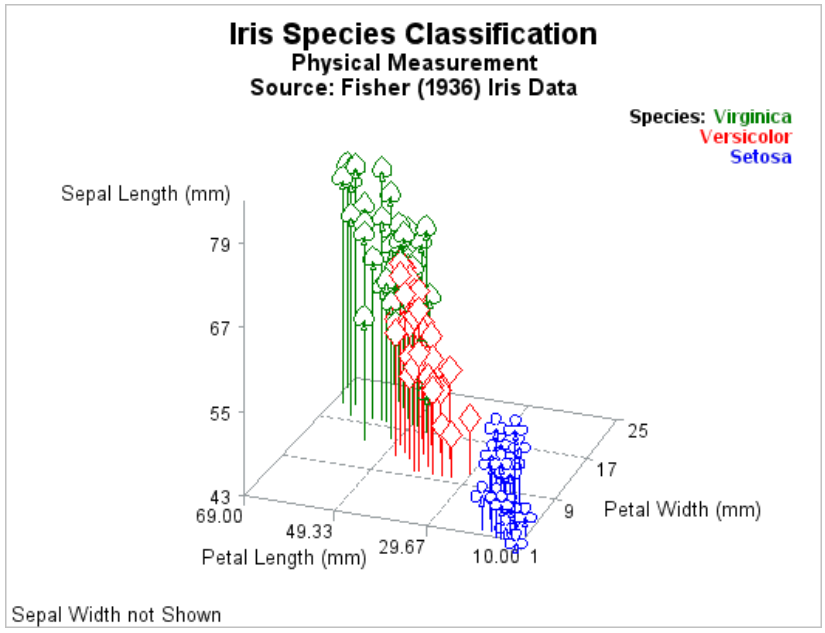
SCATTER statement

SCATTER statement options
- COLOR=
- SHAPE=
- SIZE=

NOTE statement

DATA Step

This scatter plot modifies the results of measuring the petal length, petal width, and sepal length for the flowers of three species of irises. It uses a DATA step to add a color variable and a shape variable to the data set, shapes to distinguish iris species, colors to distinguish iris species, and a NOTE statement to simulate a legend.



Program

```
goptions reset=all border;

title1 "Iris Species Classification";
title2 "Physical Measurement";
title3 "Source: Fisher (1936) Iris Data";
footnote1 j=l "Sepal Width not Shown";

data iris;
  set sashelp.iris;
  length color shape $8.;
  if species="Setosa" then do; shape="club"; color="blue"; end;
  if species="Versicolor" then do; shape="diamond"; color="red"; end;
  if species="Virginica" then do; shape="spade"; color="green"; end;
run;

proc g3d data=iris;
note j=r f="Albany AMT/bo" "Species:  " c=green "Virginica      "
         j=r    c=red "Versicolor      "
         j=r    c=blue "Setosa      ";
 scatter PetalLength*PetalWidth=SepalLength/
      color=color
      shape=shape
      size=1.5;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles and footnote.

```
title1 "Iris Species Classification";
title2 "Physical Measurement";
title3 "Source: Fisher (1936) Iris Data";
footnote1 j=l "Sepal Width not Shown";
```

Add variables to original data set.

```
data iris;
  set sashelp.iris;
  length color shape $8.;
  if species="Setosa" then do; shape="club"; color="blue"; end;
  if species="Versicolor" then do; shape="diamond"; color="red"; end;
  if species="Virginica" then do; shape="spade"; color="green"; end;
run;
```

Generate the surface plot.

```
proc g3d data=iris;
note j=r f="Albany AMT/bo" "Species:  " c=green "Virginica      "
         j=r    c=red "Versicolor      "
         j=r    c=blue "Setosa      ";
 scatter PetalLength*PetalWidth=SepalLength/
      color=color
      shape=shape
      size=1.5;
run;
quit;
```

Example 6: Generating a Scatter Plot with Modified Shapes and a Grid

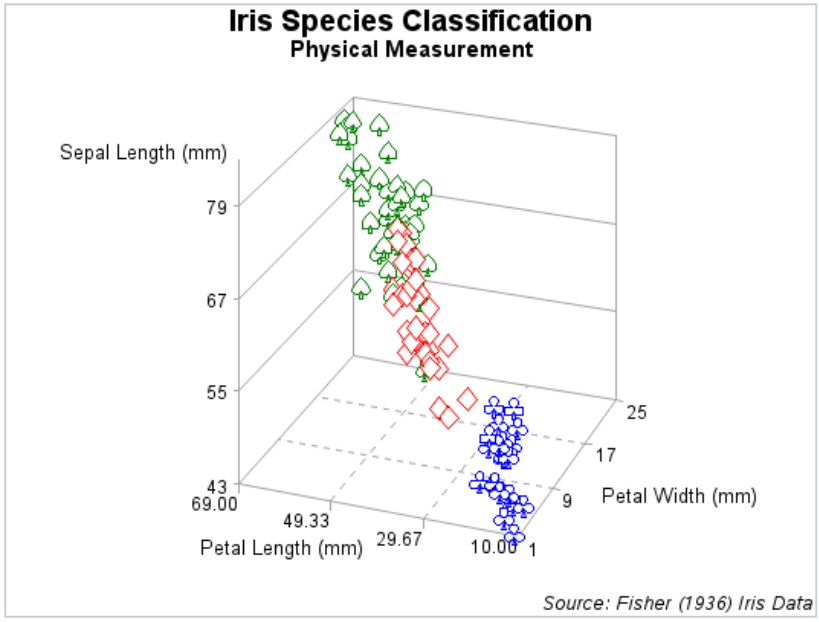
Features:

SCATTER statement

SCATTER statement options
- COLOR=
- GRID
- NONEEDLE
- SHAPE=
- SIZE=

DATA Step

This scatter plot modifies the results of measuring the petal length, petal width, and sepal length for the flowers of three species of irises. It uses a DATA step to add a color variable and a shape variable to the data set, shapes to distinguish iris species, and colors to distinguish iris species. It also removes needles from data points and adds a grid.



Program

```
goptions reset=all border;

title1 "Iris Species Classification";
title2 "Physical Measurement";
footnote1 j=r f="Albany AMT/it" "Source: Fisher (1936) Iris Data";

data iris;
   set sashelp.iris;
   length color shape $8.;
   if species="Setosa" then do; shape="club"; color="blue"; end;
   if species="Versicolor" then do; shape="diamond"; color="red"; end;
   if species="Virginica" then do; shape="spade"; color="green"; end;
run;

proc g3d data=iris;
    scatter PetalLength*PetalWidth=SepalLength/
      color=color
      shape=shape
      size=1.5
      noneedle
      grid;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles and footnote.

```
title1 "Iris Species Classification";
title2 "Physical Measurement";
footnote1 j=r f="Albany AMT/it" "Source: Fisher (1936) Iris Data";
```

Add variables to original data set.

```
data iris;
   set sashelp.iris;
   length color shape $8.;
   if species="Setosa" then do; shape="club"; color="blue"; end;
   if species="Versicolor" then do; shape="diamond"; color="red"; end;
   if species="Virginica" then do; shape="spade"; color="green"; end;
run;
```

Generate the surface plot.

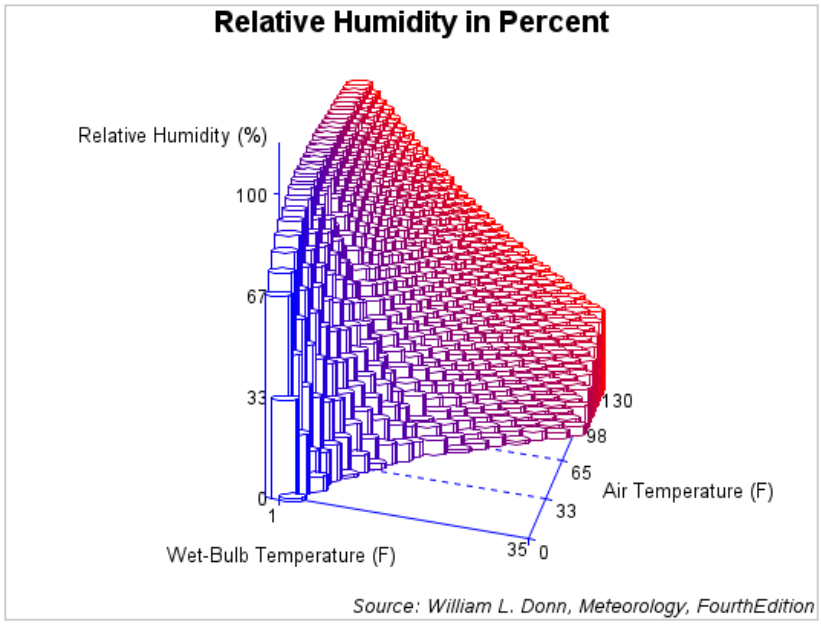
```
proc g3d data=iris;
    scatter PetalLength*PetalWidth=SepalLength/
      color=color
      shape=shape
      size=1.5
      noneedle
      grid;
run;
quit;
```

Example 7: Generating a Rotated Scatter Plot with Modified Axes

Features: Scatter statement

SCATTER statement options
- CAXIS=
- COLOR=
- ROTATE=
- SHAPE=
- SIZE=
- XTICKNUM=
- YTICKNUM=
- ZTICKNUM=
- ZMAX=
- ZMIN=

This scatter plot modifies the procedure defaults. It specifies a shape for the data points and classifies the data by color. For the axes, it specifies blue as the axis color. It specifies five major tick marks for the Y axis and two major tick marks for the X axis. For the Z axis, it specifies five major tick marks, zero as the minimum axis value, and one hundred as the maximum axis value. It also rotates the X-Y plane –15 degrees around the perpendicular Z axis.



Program

```
goptions reset=all border;

title1 "Relative Humidity in Percent";
footnote1 j=r f="Albany AMT/it"
   "Source: William L. Donn, Meteorology, Fourth
Edition";

proc g3d data=sashelp.humid;
   scatter airtemp*bulbtemp=humidity/
       shape="pillar"
       color=colorvar
       caxis=blue
       rotate=-15
       yticknum=5
       xticknum=2
       zticknum=4
       zmin=0
       zmax=100;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles and footnote.

```
title1 "Relative Humidity in Percent";
footnote1 j=r f="Albany AMT/it"
```

```
   "Source: William L. Donn, Meteorology, Fourth
Edition";
```

Generate the surface plot.

```
proc g3d data=sashelp.humid;
   scatter airtemp*bulbtemp=humidity/
       shape="pillar"
       color=colorvar
       caxis=blue
       rotate=-15
       yticknum=5
       xticknum=2
       zticknum=4
       zmin=0
       zmax=100;
run;
quit;
```

References

Fisher, R.A. 1936. “The Use of Multiple Measurements in Taxonomic Problems.” *Annals of Eugenics* : 7, 179–188..

Watkins, S.L. 1974. “Algorithm 483, Masked Three-Dimensional Plot Program with Rotations (J6).” *Collected Algorithms from ACM,* : New York: Association for Computing Machinery..

Chapter 48

G3GRID Procedure

Overview: G3GRID Procedure

What Does the G3GRID Procedure Do?

The G3GRID procedure processes an existing SAS data set to create a data set that the G3D procedure or the GCONTOUR procedure can use to produce a three-dimensional surface plot or a contour plot. The procedure creates a data set whose horizontal X-Y variable values form a complete grid, and it interpolates the values of the vertical Z variable for each point on the X-Y plane.

Using the G3GRID procedure, you can do the following actions:

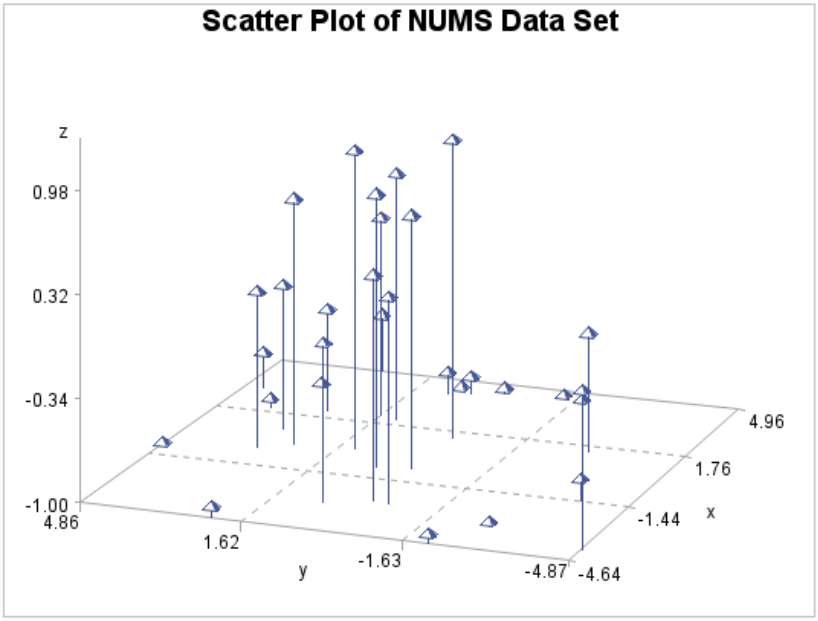
- create a rectangular grid of interpolated or smoothed values from irregularly spaced observations for use in a three-dimensional surface or contour plot
- complete a rectangular grid of interpolated or smoothed values for an input data set that has an insufficient number of observations to produce a three-dimensional surface or contour plot
- interpolate or smooth data for a three-dimensional plot

The G3GRID procedure does not produce graphics output. PROC G3GRID produces an output data set that you can use as the input data set for PROC G3D or PROC GCONTOUR.

Figure 48.197 on page 1556, and Figure 48.198 on page 1557 illustrate the effect of the G3GRID procedure on data.

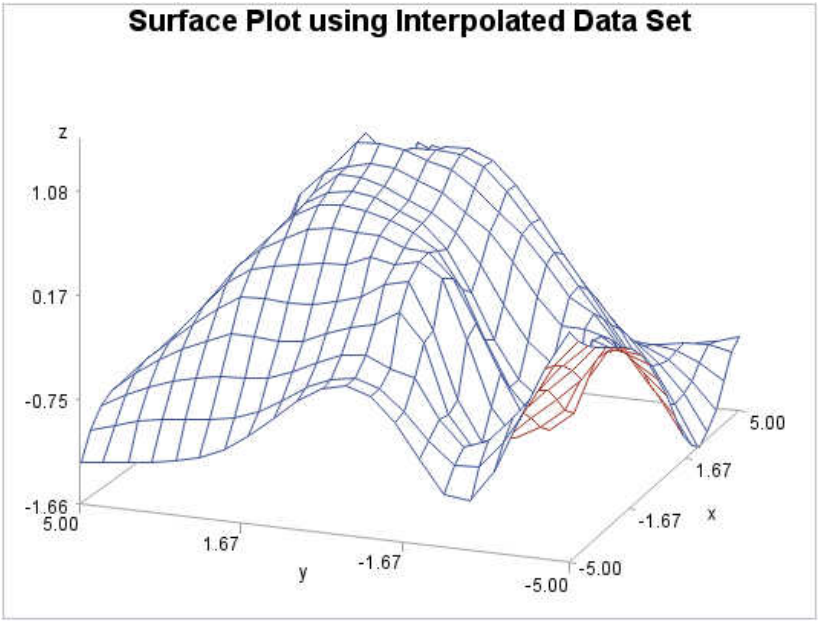
This figure shows a collection of data points, where *z=f*(*x*,*y*). These points are randomly distributed, and cannot be displayed with a G3D surface plot, although they can be displayed with a scatter plot.

Figure 48.1 *Scatter Plot of Data Set before G3GRID Processing (gtgdefin)*



The following figure shows a surface plot of the data set that is created by a G3GRID interpolation of the original data set shown in the preceding figure. The evenly distributed horizontal (*x*,*y*) data points form a grid for the three-dimensional plot.

Figure 48.2 Surface Plot of Data Set after G3GRID Processing (gtdefin)



Concepts: G3GRID Procedure

The Input Data Set

The input data set must contain at least three numeric variables:

- two horizontal variables (*x*, *y*)
- one or more vertical variables, *z* through *z-n*, that is interpolated or smoothed as if it were a function of the two horizontal variables

The G3GRID procedure can process multiple vertical variables for each pair of horizontal variables that you specify:

- if you specify more than one vertical variable, the G3GRID procedure performs a separate analysis and produces interpolated or smoothed values for each vertical variable
- if more than one observation in the input data set has the same values for both horizontal variables, *x* and *y*, only the first observation is used in the interpolation. A warning message is printed to the log.
- by default, the interpolation is performed after both variables are similarly scaled, because the interpolation methods assume that the scales of *x* and *y* are comparable

Multiple Vertical Variables

The GRID statement, enables you to name multiple vertical variables (*z* – *z-n*), to produce a data set that contains two horizontal variables and multiple vertical variables. The resulting data set enables you to produce plots of the relationships of the two horizontal variables, to different vertical variables.

Horizontal Variables along a Nonlinear Curve

If the points that are generated by the horizontal variables tend to lie along a curve, a poor interpolation or spline can result. In such cases, the vertical variable(s), and one of the horizontal variables should be modeled as a function of the remaining horizontal variable. A scatter plot of the two horizontal variables enable you to determine the appropriate function.

If the horizontal variable points are collinear, the procedure interpolates the function as constant, along lines perpendicular to the line in the plane that is generated by the input data points.

The Output Data Set

The output data set contains:

- the two horizontal variables
- the interpolated or smoothed vertical variables
- any BY variables

G3Grid enables you to control both the number of *x* and *y* values in the output data set, and the values themselves. In addition, you can specify an interpolation method.

Interpolation Methods

Types of Interpolation Methods

The G3GRID procedure can use one of three interpolation methods: bivariate interpolation (the default), spline interpolation, and smoothing spline interpolation.

Bivariate Interpolation

Unless you specify the SPLINE option, the G3GRID procedure is an interpolation procedure. It calculates the *z* values for *x*, *y* points that are missing from the input data set. The surface that is formed by the interpolated data passes precisely through the data points in the input data set.

This method of interpolation works best for fairly smooth functions, with values given at uniformly distributed points in the plane. If the data points in the input data set are erratic, the default interpolated surface can be erratic.

This default method is a modification of that described by Akima (1978). This method consists of the following actions:

1. dividing the plane into non-overlapping triangles that use the positions of the available points
2. fitting a bivariate fifth degree polynomial within each triangle
3. calculating the interpolated values by evaluating the polynomial at each grid point that falls in the triangle

The coefficients for the polynomial are computed based on the following criteria:

- the values of the function at the vertices of the triangle
- the estimated values for the first, and second derivatives of the function at the vertices

The estimates of the first, and second derivatives are computed using the *n* nearest neighbors of the point, where *n* is the number specified in the GRID statement's NEAR= option. A Delauney triangulation is used for the default method (Ripley 1981, p. 38). The coordinates of the triangles are available in an output data set, if requested by the OUTTRI= option, in the PROC G3GRID statement. This is the default interpolation method.

Spline Interpolation

If you specify the SPLINE option, a method is used that produces either an interpolation or smoothing that is optimally smooth. See Harder and Desmarais (1972), Meinguet (1979), and Green and Silverman (1994). The surface that is generated can be thought of as one that would be formed if a stiff, thin metal plate were forced through, or near the given data points. For large data sets, this method is substantially more expensive than the default method.

The function *u*, formed when you specify the SPLINE option, is determined by letting

$$t_j = (x_j, y_j)$$

$$t = (x, y)$$

and

$$|t - t_j| = \left((x - x_j)^2 + (y - y_j)^2\right)^{1/2}$$

$$u(x, y) = \sum_{j=1}^{n} c_j \ \mathrm{E}(t, t_j) + d_0 + d_1 x + d_2 y$$

where

$$\mathrm{E}(s, t) = |s - t| \ \log(|s - t|)$$

The coefficients c_1, c_2,..., c_n, and d_1, d_2, d_3 of this polynomial are determined by the following equations:

$$(\mathrm{E} + \mathrm{n}\lambda\, \mathrm{I})\, \mathrm{c} + \mathrm{Td} = \mathrm{z}$$

and

$$\mathrm{T'c} = 0$$

where:

E	is the $n \times n$ matrix $\mathrm{E}(t_i, t_j)$
I	is the $n \times n$ identity matrix
λ	is the smoothing parameter that is specified in the SMOOTH= option
c	is $(c_1, \ldots, c_n)$
z	is $(z_1, \ldots, z_n)$
d	is (d_1, d_2, d_3)
T	is the $n \times$ three matrix whose ith row is $(1, x_i, y_i)$

See Wahba (1990) for more detail.

Spline Smoothing

Using the SMOOTH= option in the GRID statement with the SPLINE option, enables you to produce a smoothing spline. See Eubank (1988) for a general discussion of spline smoothing. The value or values specified in the SMOOTH= option are substituted for λ in the equation that is described in "Spline Interpolation" on page 1559. A smoothing spline trades closeness to the original data points for smoothness. To find a value that produces the best balance between smoothness, and fit to the original data, several values for the SMOOTH= option can be run.

Syntax: G3GRID Procedure

Requirement: Exactly one GRID statement is required.

Notes: This procedure can use data that has been processed in CAS. For more information, see Chapter 12, "Plotting a Cloud Analytic Services (CAS) In-Memory Table," on page 153.

The procedure can include the "BY Statement" on page 424.

PROC G3GRID <DATA=*input-data-set*>
<OUT=*output-data-set*>
<OUTTRI=*output-data-set*>;
GRID *grid-request* <*/options*>;

PROC G3GRID Statement

Identifies the input data set. Can also specify one, or two output data sets.

Requirement: An input data set is required.

Syntax

PROC G3GRID <DATA=*input-data-set*>
<OUT=*output-data-set*>
<OUTTRI=*output-data-set*>
;

Optional Arguments

DATA=*input-data-set*
specifies the SAS data set that contains the variables to process. By default, the procedure uses the most recently created SAS data set.

See "The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

"The Input Data Set" on page 1557

OUT=*output-data-set*
specifies the output data set. The data set contains any BY variables that you specify, the interpolated or smoothed values of the vertical variables (*z* through *z-n*), and the coordinates for all grid positions on the horizontal (*x-y*) plane. If you specify smoothing, the output data set also contains a variable named _SMTH_, whose value is a smoothing parameter. The observations in this data set are ordered by any variables that you specify with a BY statement. By default, the output of PROC G3GRID creates WORK.DATA1.

Depending on the shape of the original data, and the options that you use, the output data set can contain values for the vertical (*z* through *z-n*) values that are outside of the range of the original values in the data set.

Example "Example 1: Using the Default Interpolation Method" on page 1567

OUTTRI=*output-data-set*

specifies an additional output data set that contains triangular coordinates. The data set will contain any BY variables that you specify, the two horizontal variables giving the horizontal (*x* -*y*) plane coordinates of the input points, and a variable named TRIANGLE that uses the integer values to label the triangles. The observations in this data set are ordered by any variables that you specify with a BY statement.

The data set contains three observations for each value of the variable TRIANGLE. The three observations give the coordinates of the three vertices of the triangle. Points on the convex hull of the input data set of points are also assumed to lie in degenerate triangles, whose other vertices are at infinity. The points in the convex hull can be recovered by keeping only those triangles with exactly two missing vertices.

By default, no OUTTRI= data set is produced. OUTTRI= is not valid when you specify the SPLINE option in the GRID statement.

GRID Statement

Specifies the three numeric variables for interpolation or for smoothing. Can also specify the number of observations (*x* and *y* values), in the output data set; output values for the two horizontal variables *x*-*y*; and the interpolation method for the vertical variables.

Requirement: Exactly one grid request is required.

Syntax

GRID *grid-request* <*/options*>;

Summary of Optional Arguments

Grid options

AXIS1=*ascending-value-list*

specifies a list of numeric values to assign to the first (*y*) variable in the grid request for the output data set.

AXIS2=*ascending-value-list*

specifies a list of numeric values to assign to the second (*x*) variable in the grid request for the output data set.

NAXIS1=*n*

specifies the number of values for the first (*y*) variable in the grid request for the output data set.

NAXIS2=*n*

specifies the number of values for the second (*x*) variable in the grid request for the output data set.

Interpolation options

JOIN

uses a linear interpolation within a set of triangular regions that are formed from the input data set.

NEAR=*n*
specifies the number of the nearest data points to use for computing the estimates of the first derivative, and the second derivative.

NOSCALE
specifies that the *x* and *y* variables not be scaled to the same range before interpolation.

PARTIAL
specifies that a spline be used to estimate the derivatives for the biquintic polynomial interpolation.

SCALE
specifies that the *x* and *y* variables be scaled to the same range before interpolation.

SMOOTH=*ascending-value-list*
specifies a list of numbers for smoothing parameters.

SPLINE
specifies the use of a bivariate spline to interpolate, or to form a smoothed estimate, if you also use the SMOOTH= option.

Required Argument

grid-request
specifies three or more numeric variables from the input data set as `y*x=z(s)`. The variables are:

y
is one of the variables that form the horizontal (*x*-*y*) plane

x
is another of the variables that form the horizontal (*x*-*y*) plane

z(s)
is one or more of the vertical variables for the interpolation

Although the GRID statement can specify only two horizontal variables, it can include multiple vertical variables. Separate vertical variables with blanks:

```
grid x*y=z w u v;
```

Optional Arguments

AXIS1=*ascending-value-list*
specifies a list of numeric values to assign to the first (*y*) variable in the grid request for the output data set. Numbers that you specify with this option determine the number of values for *y*, and override a value that you specify with the NAXIS1= option. The *ascending-value-list* must be arranged in ascending order. The value list can be in any of the following forms:

- *n* <...*n*>
- *n* TO *n* <BY *increment*>
- *n* <...*n*> TO *n* <BY *increment* > <*n* <...*n*> >

Examples "Example 1: Using the Default Interpolation Method" on page 1567

"Example 4: Spline Interpolation" on page 1574

AXIS2=*ascending-value-list*

specifies a list of numeric values to assign to the second (*x*) variable in the grid request for the output data set. Numbers that you specify with this option determine the number of values for *x* and override a value that you specify with the NAXIS2= option. The *ascending-value-list* must be arranged in ascending order. The value list can be in any of the following forms:

- *n* <...*n*>
- *n* TO *n* <BY *increment*>
- *n* <...*n*> TO *n* <BY *increment* > <*n* <...*n*> >

Examples "Example 1: Using the Default Interpolation Method" on page 1567

"Example 4: Spline Interpolation" on page 1574

JOIN

uses a linear interpolation within a set of triangular regions that are formed from the input data set. This interpolation method creates values in the range of the initial values of the vertical variable, but the resulting interpolated surface might not be smooth.

NAXIS1=*n*

specifies the number of values for the first (*y*) variable in the grid request for the output data set. You can determine the actual values used for *y* by taking the minimum and maximum values of *y* and dividing the range into *n*- one equal sections.

A value specified with NAXIS1= is ignored if values are also specified with AXIS1=.

Default 11

NAXIS2=*n*

specifies the number of values for the second (*x*) variable in the grid request for the output data set. You can determine the actual values that are used for *x* by taking the minimum value and the maximum value of *x*, and dividing the range into *n*- one equal sections.

A value specified with NAXIS2= is ignored if values are also specified with AXIS2=.

Default 11

NEAR=*n*

specifies the number of the nearest data points to use for computing the estimates of the first derivative, and the second derivative. As NEAR= values become larger, time and computation costs increase significantly. NEAR= is ignored if you specify SPLINE. The value of *n* must be greater than or equal to 3.

If the number of input data points is insufficient for the number that you specify with NEAR=, a smaller number of data points is used.

Default 3

Example "Example 3: Partial Spline Interpolation" on page 1573

NOSCALE

specifies that the *x* and *y* variables not be scaled to the same range before interpolation. By default, the interpolation is performed after both variables are similarly scaled because the interpolation methods assume that the scales of *x* and *y* are comparable.

Default SCALE

PARTIAL

specifies that a spline be used to estimate the derivatives for the biquintic polynomial interpolation. A bivariate spline is fit to the nearest neighbors, and is used to estimate the needed derivatives. This option produces results that are less smooth than those produced by the SPLINE option and uses fewer computer resources. However, the results produced by PARTIAL are smoother than those that are produced by the default. If you use both the PARTIAL option and the SPLINE option, the PARTIAL option is ignored.

Example "Example 3: Partial Spline Interpolation" on page 1573

SCALE

specifies that the *x* and *y* variables be scaled to the same range before interpolation. The interpolation is performed after both variables are similarly scaled because the interpolation methods assume that the scales of *x* and *y* are comparable.

Default SCALE

SMOOTH=*ascending-value-list*

specifies a list of numbers for smoothing parameters. Use the SMOOTH= option only when you also use the SPLINE option. The *ascending-value-list* must be arranged in ascending order. The value list can be in any of the following forms:

- *n* <...*n*>
- *n* TO *n* <BY *increment*>
- *n* <...*n*> TO *n* <BY *increment* > <*n* <...*n*> >

For each value λ of the smoothing parameter, a function *u* (*x*, *y*) is formed that minimizes

$$\frac{1}{n}\sum_{j=1}^{n}\left(\mathrm{u}\left(x_j,\ y_j\right)-z_j\right)^2+\ \lambda\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(\left(\frac{\partial^2\,\mathrm{u}}{\partial^2\,\mathrm{x}}\right)^2+2\left(\frac{\partial^2\,\mathrm{u}}{\partial\mathrm{x}\partial\,\mathrm{y}}\right)^2+\left(\frac{\partial^2\,\mathrm{u}}{\partial^2\,\mathrm{y}}\right)^2\right)\mathrm{dxdy}$$

where *n* is the number of datapoints, and the pairs (x_j, y_j) are the available points, with corresponding function values z_j (Wahba 1990).

The higher the value of the smoothing parameter, the smoother the resulting interpolation. The lower the smoothing parameter, the closer the resulting surface is to the original data points. A smoothing parameter of 0 produces the same results as the SPLINE option without the SMOOTH= option.

This procedure repeats for each value of the smoothing parameter. The output data set that you specify in the OUT= option contains:

- the interpolated values
- the values of the grid points
- the values of the smoothing parameter in the variable _SMTH_
- a separate grid for each value of the smoothing parameter

Example "Example 2: Spline and Smoothing Interpolations" on page 1570

SPLINE
specifies the use of a bivariate spline to interpolate, or to form a smoothed estimate, if you also use the SMOOTH= option. The SPLINE option results in the use of an order n^3 algorithm, where n is the number of input data points. Consequently, this method can be time-consuming. If you use more than 100 input points, the procedure can use excessive time.

See Harder and Desmarais (1972)

Meinguet (1979)

Green and Silverman (1994)

Examples "Example 2: Spline and Smoothing Interpolations" on page 1570

"Example 4: Spline Interpolation" on page 1574

Details

Controlling Observations in the Output Data Set

The G3GRID procedure produces a data set with 121 observations for combinations of eleven values for each of the horizontal variables, *x* and *y*. To create a data set with a different number of observations, use the GRID statement's NAXIS1= option, or the NAXIS2= option to specify the number of the values of *y* or *x*, respectively. You can use the GRID statement's AXIS1= option or the AXIS2= option to specify the actual values for *y* or *x*, respectively.

The following table shows the number of observations that will be in the output data set if you use any of these options.

If you specify multiple smoothing parameters, the number of observations in the output data set will be the number shown in the table, multiplied by the number of smoothing values that you specify in the SMOOTH= option. If you use BY-group processing, multiply the number in the table by the number of BY groups.

Table 48.1 *Number of Observations Contained in the Output Data Set*

Options Specified	Number of Observations in Output Data Set
None	121
AXIS1=	(number of values for AXIS1=) * 11
AXIS2=	(number of values for AXIS2=) * 11
NAXIS1=	(value of NAXIS1=) * 11
NAXIS2=	(value of NAXIS2=) * 11

Options Specified	Number of Observations in Output Data Set
AXIS1=, AXIS2=	(number of values for AXIS1=) * (number of values for AXIS2=)
AXIS1=, NAXIS1=	(number of values for AXIS1=) * 11
AXIS1=, NAXIS2=	(number of values for AXIS1=) * (value of NAXIS2=)
AXIS2=, NAXIS1=	(number of values for AXIS2=) * (value of NAXIS1=)
AXIS2=, NAXIS2=	(number of values for AXIS2=) * 11
NAXIS1=, NAXIS2=	(value of NAXIS1=) * (value of NAXIS2=)

Depending on the shape of the original data, and the options that you specify, the output data set can contain values for the vertical (*z*) values that are outside of the range of the original values in the data set.

Examples: G3GRID Procedure

Example 1: Using the Default Interpolation Method

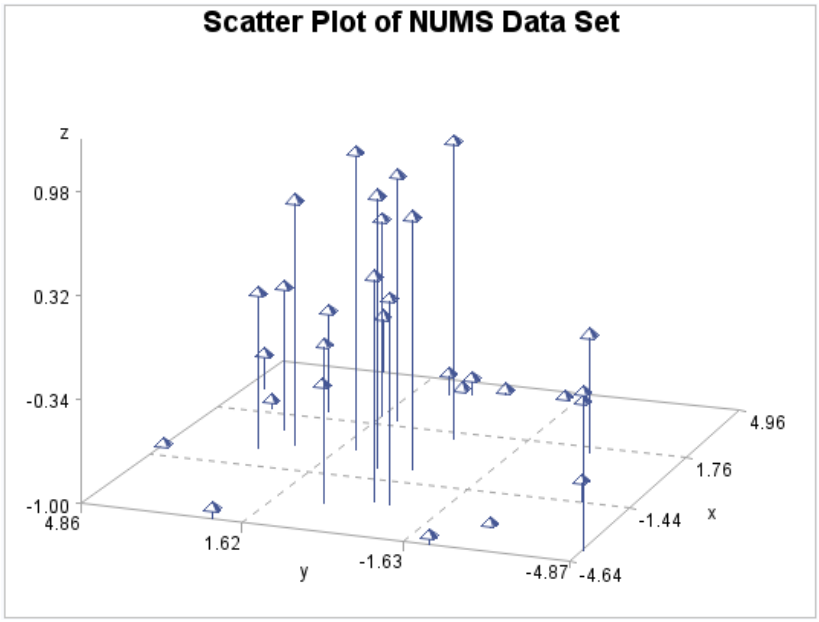
Features: G3GRID statement option OUT=
GRID statement options AXIS1= and AXIS2=
DATA step and the G3D procedure

Sample library member: GTGDEFIN

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

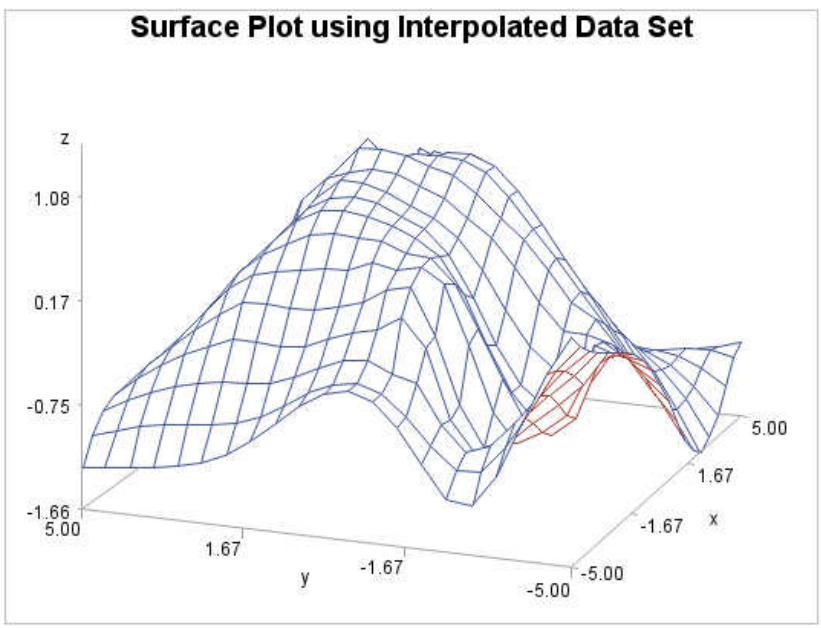
This example demonstrates the default interpolation method that is used by the GRID statement. The example first generates a scatter plot of random data to show the concentration of data values before processing the data set with the G3GRID procedure. The original data does not contain enough combinations of *x*, *y*, and *z* values to generate a surface plot with the G3D procedure or generate a contour plot with the GCONTOUR procedure.

Output 48.1 *Scatter Plot of NUMS Data Set (gtgdefin)*



The example then runs the G3GRID procedure to interpolate additional *x*, *y*, and *z* values. Because no interpolation method is specified, the default interpolation method is used. The resulting output data set is used as input to the G3D procedure, which generates the surface plot shown in the following output.

Output 48.2 *Surface Plot Using Interpolated Data Set (gtgdefin)*



Program

```
goptions reset=all border;

data nums;
```

```
   keep x y z;
   do i=1 to 30;
      x=10*ranuni(33)-5;
      y=10*ranuni(33)-5;
      z=sin(sqrt(x*x+y*y));
      output;
   end;
run;

title "Scatter Plot of NUMS Data Set";

proc g3d data=nums;
   scatter y*x=z;
   run;
quit;

proc g3grid data=nums out=default;
   grid y*x=z /
      axis1=-5 to 5 by .5
      axis2=-5 to 5 by .5;
   run;
quit;

title "Surface Plot using Interpolated Data
Set";

proc g3d data=default;
   plot y*x=z;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create data set. NUMS uses a set of randomly sampled points to create the data used in this, and all remaining examples in this chapter.

```
data nums;
   keep x y z;
   do i=1 to 30;
      x=10*ranuni(33)-5;
      y=10*ranuni(33)-5;
      z=sin(sqrt(x*x+y*y));
      output;
   end;
run;
```

Define the title for the plot.

```
title "Scatter Plot of NUMS Data Set";
```

Generate the scatter plot with PROC G3D.

```
proc g3d data=nums;
   scatter y*x=z;
   run;
quit;
```

Grid the data with PROC G3GRID. The OUT= option on PROC G3GRID specifies a name for the temporary output data set. The GRID option specifies the variables Y*X=Z for the output data set. The AXIS statements define axes ranges.

```
proc g3grid data=nums out=default;
   grid y*x=z /
      axis1=-5 to 5 by .5
      axis2=-5 to 5 by .5;
   run;
quit;
```

Define the title for the plot.

```
title "Surface Plot using Interpolated Data
Set";
```

Generate the surface plot. The G3D procedure using the G3GRID procedure's output data set as the input data set.

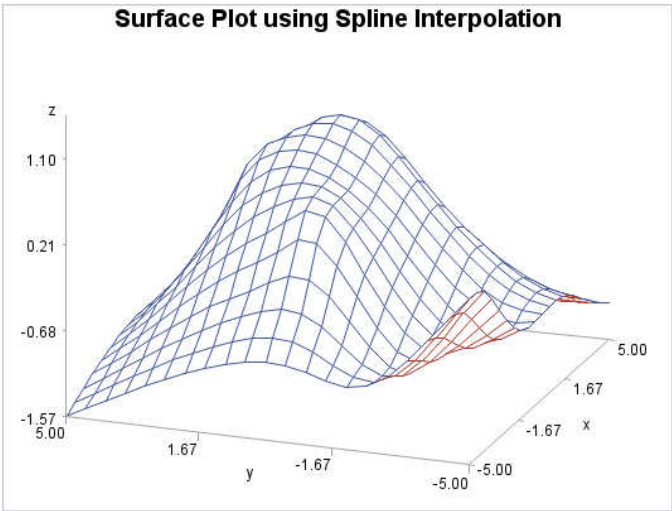
```
proc g3d data=default;
   plot y*x=z;
run;
quit;
```

Example 2: Spline and Smoothing Interpolations

Features:	GRID statement options SMOOTH= and SPLINE
Data set:	NUMS
Sample library member:	GTGSISS
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

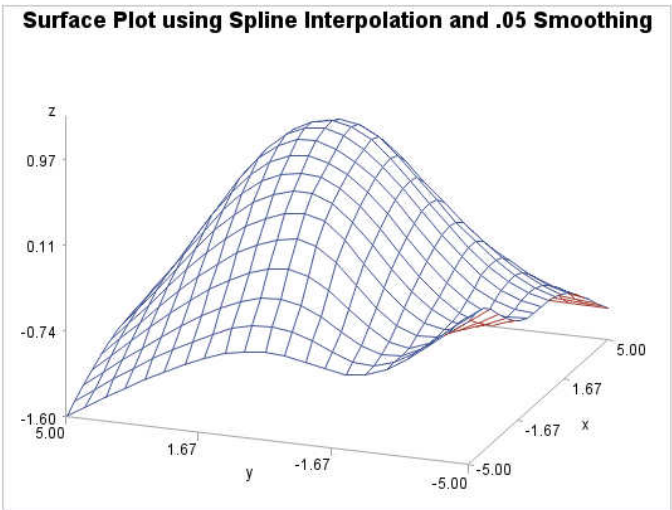
This example extends “Example 1: Using the Default Interpolation Method” on page 1567 to specify the SPLINE option in the GRID statement. The output data set, when used in PROC G3D, generates a smoother surface plot.

Output 48.3 *Surface Plot Using Spline Interpolation (gtgsiss)*



The following plot extends "Example 1: Using the Default Interpolation Method" on page 1567 to specify the SPLINE option, and the SMOOTH= option in the GRID statement. The SMOOTH= option is set to .05 for additional smoothing. The output data set, when used in PROC G3D, generates a smoother surface plot.

Output 48.4 *Surface Plot Using Spline Interpolation and .05 Smoothing (gtgsiss)*



Program

```
goptions reset=all  border;

title "Surface Plot using Spline
Interpolation";

proc g3grid data=nums out=spline;
   grid y*x=z / spline
                axis1=-5 to 5 by .5
                axis2=-5 to 5 by .5;
run;
```

```
proc g3d data=spline;
   plot y*x=z ;
run;
quit;

title "Surface Plot using Spline Interpolation and
.05 Smoothing";

proc g3grid data=nums out=smoothed;
   grid y*x=z / spline
                smooth=.05
                axis1=-5 to 5 by .5
                axis2=-5 to 5 by .5;
run;
quit;

proc g3d data=smoothed;
   plot y*x=z;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all  border;
```

Define the title for the plot.

```
title "Surface Plot using Spline
Interpolation";
```

Process points with PROC G3GRID. The SPLINE option specifies the bivariate spline method for the data set interpolation.

```
proc g3grid data=nums out=spline;
   grid y*x=z / spline
                axis1=-5 to 5 by .5
                axis2=-5 to 5 by .5;
run;
```

Generate the surface plot.

```
proc g3d data=spline;
   plot y*x=z ;
run;
quit;
```

Define the title for the plot.

```
title "Surface Plot using Spline Interpolation and
.05 Smoothing";
```

Process the data with PROC G3GRID. The SMOOTH=.05 option specifies the smoothing parameter to use during spline interpolation.

```
proc g3grid data=nums out=smoothed;
   grid y*x=z / spline
                smooth=.05
                axis1=-5 to 5 by .5
                axis2=-5 to 5 by .5;
```

```
run;
quit;
```

Generate the surface plot.

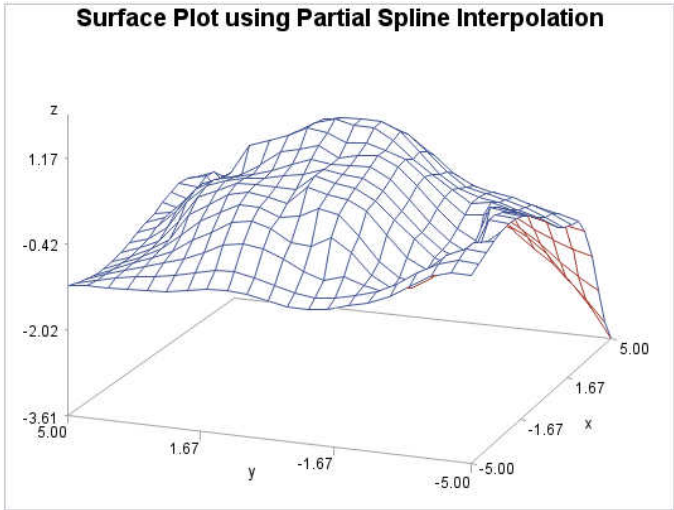
```
proc g3d data=smoothed;
   plot y*x=z;
run;
quit;
```

Example 3: Partial Spline Interpolation

Features:	GRID statement options NEAR and PARTIAL
Data set:	NUMS
Sample library member:	GTGPART
Note:	The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

This example specifies a partial spline interpolation in the GRID statement, using the eight nearest neighbors for computing the estimates of the first, and second derivatives. The output data set, when used in PROC G3D, generates a more smooth surface plot than the surface plot that results from the default interpolation shown in “Example 1: Using the Default Interpolation Method” on page 1567 but does not generate the smoothness of the surface plot that results from the spline interpolation shown in “Example 2: Spline and Smoothing Interpolations” on page 1570 .

Output 48.5 *Surface Plot Using Partial Spline Interpolation (gtgpart)*



Program

```
goptions reset=all border;

proc g3grid data=nums out=partial;
   grid y*x=z / partial
                near=8
                axis1=-5 to 5 by .5
                axis2=-5 to 5 by .5;
run;

title "Surface Plot using Partial Spline
Interpolation";

proc g3d data=partial;
   plot y*x=z;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Process data with PROC G3GRID. The PARTIAL option specifies that a spline be used to estimate the derivatives for the biquintic polynomial interpolation. The NEAR= option specifies the number of nearest neighbors to be used for computing the estimates of the first, and the second derivatives.

```
proc g3grid data=nums out=partial;
   grid y*x=z / partial
                near=8
                axis1=-5 to 5 by .5
                axis2=-5 to 5 by .5;
run;
```

Define title for the plot.

```
title "Surface Plot using Partial Spline
Interpolation";
```

Generate the surface plot.

```
proc g3d data=partial;
   plot y*x=z;
run;
quit;
```

Example 4: Spline Interpolation

Features: GRID statement options AXIS1=, AXIS2=, and SPLINE

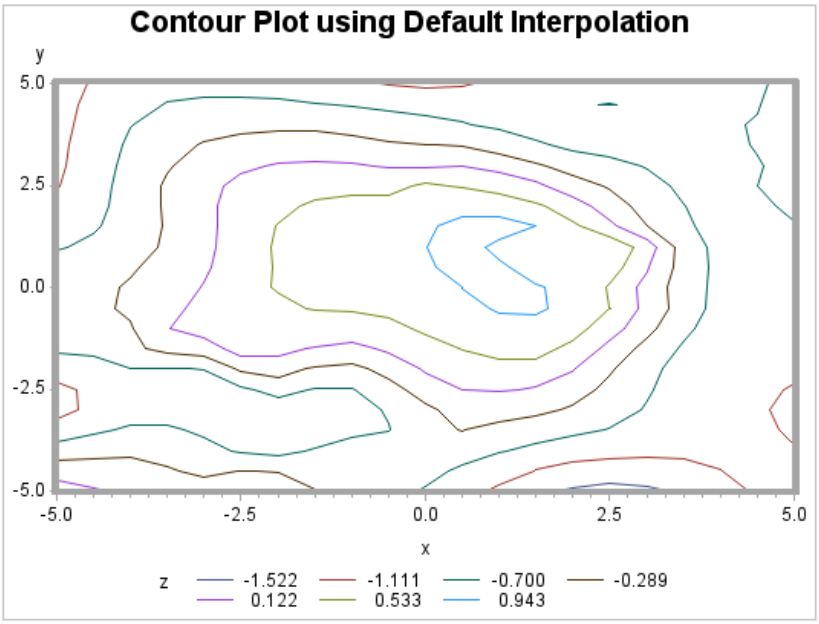
Data set: NUMS

Sample library member: GTGSPLIN

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com, under the Related Documentation heading.

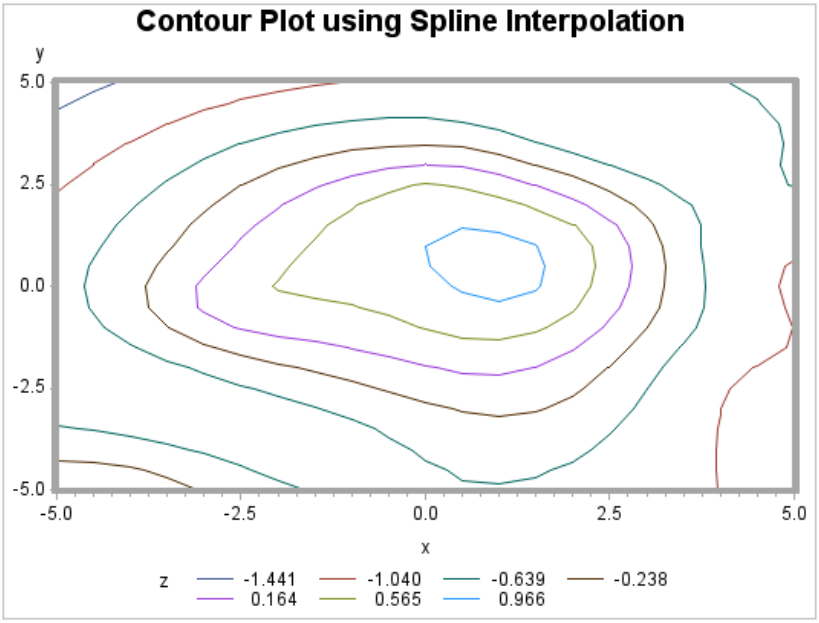
This example demonstrates the default interpolation method when used by the GCONTOUR procedure to generate a contour plot from the resulting output data set.

Output 48.6 *Contour Plot Using Default Interpolation (gtgsplin)*



The second plot, demonstrates the spline interpolation method when used by the GCONTOUR procedure to generate a contour plot from the resulting output data set.

Output 48.7 Contour Plot Using Spline Interpolation (gtgsplin)



Program

```
goptions reset=all border;

title "Contour Plot using Default
Interpolation";

axis1 width=3;

proc g3grid data=nums out=numdef;
   grid y*x=z /
      axis1=-5 to 5 by .5
      axis2=-5 to 5 by .5;
   run;

proc gcontour data=numdef;
   plot y*x=z /
      haxis=axis1
      vaxis=axis1;
   run;
quit;

title "Contour Plot using Spline
Interpolation";

proc g3grid data=nums out=numspl;
   grid y*x=z / spline
                axis1=-5 to 5 by .5
                axis2=-5 to 5 by .5;
run;

proc gcontour
data=numspl;
   plot y*x=z /
      haxis=axis1
```

```
      vaxis=axis1;
   run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the plot.

```
title "Contour Plot using Default
Interpolation";
```

Define the axis characteristics.

```
axis1 width=3;
```

Process data with PROC G3GRID.

```
proc g3grid data=nums out=numdef;
   grid y*x=z /
      axis1=-5 to 5 by .5
      axis2=-5 to 5 by .5;
   run;
```

Generate the contour after default interpolation.

```
proc gcontour data=numdef;
   plot y*x=z /
      haxis=axis1
      vaxis=axis1;
   run;
quit;
```

Define the title for the plot.

```
title "Contour Plot using Spline
Interpolation";
```

Process data with PROC G3GRID. The SPLINE option specifies the bivariate spline method for the interpolation.

```
proc g3grid data=nums out=numspl;
   grid y*x=z / spline
                axis1=-5 to 5 by .5
                axis2=-5 to 5 by .5;
run;
```

Generate the contour plot using the spline interpolation.

```
proc gcontour
data=numspl;
   plot y*x=z /
      haxis=axis1
      vaxis=axis1;
   run;
quit;
```

References

Akima, Hiroshi 1978. “A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points.” *ACM Transaction on Mathematical Software* : 4, 148–159.

Eubank, R.L. 1988. *Spline Smoothing and Nonparametric Regression*. New York, NY: Marcel Dekker.

Green, P.J. and Silverman, B.W. 1994. *Nonparametric Regression and Generalized Linear Models*. London, : Chapman & Hall.

Harder, R.L. and Desmarais, R.N. 1972. “Interpolation Using Surface Splines.” *Journal of Aircraft* : 9, 189–191.

Meinguet, Jean 1979. “Multivariate Interpolation at Arbitrary Points Made Simple.” *Journal of Applied Mathematics and Physics* : 30, 292–304.

Ripley, B.D. 1981. *Spatial Statistics*. New York, NY: John Wiley & Sons, Inc.

Wahba, G. 1990. *Spline Models for Observational Data*. Philadelphia, PA: SIAM.

PART 8

Appendixes

Appendix 1

Summary of ActiveX and Java Support

Introduction

The following tables summarize which options and annotate variables are supported or partially supported by the ACTIVEX, ACTXIMG, and JAVAIMG devices. Partial support for options that refer to global statements, such as the GAXIS= option, indicates that some but not all AXIS statement options are supported. Partial support can also indicate that an option works differently for the other devices than it does for the ACTIVEX, ACTXIMG, and JAVAIMG device drivers. For a complete description of each option or variable, refer to the documentation for the option or variable.

Note: Starting with SAS 9.4M8, the ACTIVEX and ACTXIMG graphics devices are obsolete. If you specify the ACTIVEX or ACTXIMG device, the JAVAIMG device is used instead.

SAS/GRAPH Statements

AXIS Statement

Table A24.1 *ActiveX and Java Support for the AXIS Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR= C=	Yes	Yes
INTERVAL=	No	No
LABEL=	Yes (partial)	Yes (partial)
LENGTH=	Yes	No
LOGBASE=	Yes	No
LOGSTYLE=	Yes	No
MAJOR=	Yes (partial)	Yes (partial)
MINOR=	Yes (partial)	Yes (partial)
NOBRACKETS	No	No
NOPLANE	Yes	Yes
OFFSET=	Yes	No
ORDER=	Yes (partial)	Yes (partial)
ORIGIN=	No	No
REFLABEL=	No	No
SPLIT=	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
STYLE=	Yes	Yes
VALUE=	Yes	Yes (partial)
WIDTH=	Yes (partial)	No

AXIS Statement Text Description Suboptions

Text description suboptions are used by the LABEL=, REFLABEL=, and VALUE= options.

Table A24.2 *ActiveX and Java Support for the AXIS Text Description Suboptions*

Suboption	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANGLE= A=	Yes	Yes (partial)
AUTOREF	No	No
COLOR= C=	Yes	Yes
FONT= F=	Yes	Yes (partial)
HEIGHT= H=	Yes	Yes
JUSTIFY= J=	Yes	No
POSITION=	No	No
ROTATE= R=	Yes	Yes (partial)
TICK= T=	No	No

AXIS Statement Tick Mark Description Suboptions

Tick mark description suboptions are used by the MAJOR= and MINOR= options to change the color, height, width, and number of the tick marks to which they apply.

Table A24.3 ActiveX and Java Support for the Tick Mark Description Suboptions

Suboption	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR= C=	Yes	Yes
HEIGHT= H=	No	No
NUMBER= N=	Yes	Yes
WIDTH= W=	Yes	Yes (partial)

GOPTIONS Statement

You must specify the ODS USEGOPT statement for the CTEXT=, CTITLE=, FTEXT=, FTITLE=, HTEXT=, and HTITLE= options to work for the Java and ActiveX devices. See "Using Graphics Options with ODS (USEGOPT)" on page 116 for more information.

Table A24.4 ActiveX and Java Support for the GOPTIONS Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ACCESSIBLE	Yes	Yes
ASPECT=	No	No
AUTOCOPY NOAUTOCOPY	No	No
AUTOFEED NOAUTOFEED	No	No
AUTOSIZE=	No	No
BINDING=	No	No
BORDER	Yes	Yes
CBACK=	Yes	Yes
CBY=	No	No
CELL	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CHARACTERS NOCHARCTERS	No	No
CHARTYPE=	No	No
CIRCLEARC NOCIRCLEARC	No	No
COLLATE NOCOLLATE	No	No
COLORS=	Yes	Yes
CPATTERN=	No	No
CSYMBOL=	No	No
CTEXT=	Yes	Yes (partial)
CTITLE=	Yes	Yes
DASH NODASH	No	No
DASHSCALE=	No	No
DELAY=	No	No
DEVADDR=	No	No
DEVICE=	Yes	Yes
DEVMAP=	No	No
DISPLAY NODISPLAY	No	No
DISPOSAL=	No	No
DRVINIT=	No	No
DRVTERM=	No	No
DUPLEX NODUPLEX	No	No
ERASE NOERASE	No	No
EXTENSION	No	No
FASTTEXT NOFASTTEXT	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
FBY=	No	No
FCACHE=	No	No
FILECLOSE=	No	No
FILEONLY NOFILEONLY	No	No
FILL NOFILL	No	No
FILLINC=	No	No
FONTRES=	No	No
FTEXT=	Yes (partial)	Yes (partial)
FTITLE=	Yes	Yes
FTRACK=	No	No
GACCESS=	No	No
GCOPIES=	No	No
GEND=	No	No
GEPILOG=	No	No
GOUTMODE=	No	No
GPROLOG=	No	No
GPROTOCOL=	No	No
GRAPHRC NOGRAPHRC	No	No
GSFLEN=	No	No
GSFMODE=	No	No
GSFNAME=	No	No
GSFPROMPT NOGSFPROMPT	No	No
GSIZE=	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
GSTART=	No	No
GUNIT=	Yes (partial)	Yes (partial)
GWAIT=	No	No
HANDSHAKE=	No	No
HBY=	No	No
HORIGIN=	No	No
HPOS=	No	No
HSIZE=	Yes (partial)	Yes (partial)
HTEXT=	Yes	Yes (partial)
HTITLE=	Yes	Yes
IBACK=	Yes	Yes (partial)
IMAGEPRINT NOIMAGEPRINT	No	No
IMAGESTYLE=	Yes	No
INTERLACED NOINTERLACED	No	No
INTERPOL=	No	No
ITERATION=	No	No
KEYMAP=	No	No
LFACTOR=	No	No
OFFSHADOW=	No	No
PAPERDEST=	No	No
PAPERFEED=	No	No
PAPERLIMIT=	No	No
PAPERSIZE=	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
PAPERSOURCE=	No	No
PAPERTYPE=	No	No
PCLIP NOPCLIP	No	No
PENMOUNTS=	No	No
PENSORT NOPENSORT	No	No
PIEFILL NOPIEFILL	No	No
POLYGONCLIP NOPOLYGONCLIP	No	No
POLYGONFILL NOPOLYGONFILL	No	No
POSTGEPILOG=	No	No
POSTGPROLOG=	No	No
POSTGRAPH=	No	No
PPDFILE=	No	No
PREGEPILOG=	No	No
PREGPROLOG=	No	No
PREGRAPH=	No	No
PROMPT NOPROMPT	No	No
PROMPTCHARS=	No	No
RENDER=	No	No
RENDERLIB=	No	No
REPAINT=	No	No
RESET	Yes	Yes
REVERSE NOREVERSE	No	No
ROTATE=	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ROTATE NOROTATE	No	No
SIMFONT=	No	No
SPEED=	No	No
SWAP NOSWAP	No	No
SWFONTRENDER	No	No
SYMBOL NOSYMBOL	No	No
TARGETDEVICE=	No	No
TRANSPARENCY NOTRANSPARENCY	Yes (partial)	No
TRANTAB=	No	No
UCC=	No	No
USERINPUT NOUSERINPUT	No	No
VORIGIN=	No	No
VPOS=	No	No
VSIZE=	Yes (partial)	Yes (partial)
V6COMP NOV6COMP	Yes (partial)	Yes (partial)
XMAX=	No	No
XPIXELS=	Yes (partial)	Yes (partial)
YMAX=	No	No
YPIXELS=	Yes (partial)	Yes (partial)

LEGEND Statement

Table A24.5 ActiveX and Java Support for the LEGEND Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ACROSS=	Yes	Yes
CBLOCK=	Yes	No
CBORDER=	Yes	Yes
CFRAME=	Yes	Yes
CSHADOW=	Yes	Yes
DOWN=	Yes	Yes
FRAME	Yes	Yes
FWIDTH=	No	No
LABEL=	Yes (partial)	Yes (partial)
MODE=	No	No
OFFSET=	No	No
ORDER=	No	No
ORIGIN=	No	No
POSITION=	Yes	Yes (partial)
REPEAT=	No	No
SHAPE=	No	No
VALUE=	Yes (partial)	Yes (partial)

LEGEND Statement Text Description Suboptions

Text description suboptions are used by the LABEL= and VALUE= options to change the color, height, justification, font, and angle of either default text or specified text strings. See LABEL= and VALUE=.

Table A24.6 *ActiveX and Java Support for the LEGEND Statement Text Description Suboptions*

Suboption	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR= C=	Yes	Yes
FONT= F=	Yes (partial)	Yes
HEIGHT= H=	Yes	Yes
JUSTIFY= J=	Yes	Yes
POSITION=	Yes (partial)	No
TICK= T=	Yes	Yes

PATTERN Statement

Table A24.7 *ActiveX and Java Support for the PATTERN Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR= C=	Yes (partial)	Yes (partial)
IMAGE=	Yes (partial)	Yes (partial)
IMAGESTYLE=	Yes (partial)	Yes (partial)
REPEAT= R=	Yes (partial)	Yes (partial)
VALUE=*bar/block-pattern* V=*bar/block-pattern*	Yes (partial)	Yes (partial)
	Yes (partial)	Yes (partial)

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
VALUE=*map/plot-pattern* V=*map/plot-pattern*		
VALUE=*pie/star-pattern* V=*pie/star-pattern*	Yes (partial)	Yes (partial)

SYMBOL Statement

Table A24.8 *ActiveX and Java Support for the SYMBOL Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
BWIDTH=	Yes	Yes
CI=	Yes	Yes
CO=	Yes	Yes
COLOR= C=	Yes (GPLOT and GBARLINE) No (GCONTOUR)	Yes (GPLOT and GBARLINE) No (GCONTOUR)
CV=	Yes (GPLOT) No (GCONTOUR)	Yes (GPLOT) No (GCONTOUR)
FONT=	No	No
HEIGHT= H=	Yes (GPLOT) No (GCONTOUR)	Yes (GPLOT) No (GCONTOUR)
INTERPOL=BOX I=BOX	Yes	Yes (partial)
INTERPOL=HILO I=HILO	Yes	Yes (partial)
INTERPOL=JOIN I=JOIN	Yes	Yes
INTERPOL=L I=L	Yes	Yes
INTERPOL=*map/plot-pattern* I=*map/plot-pattern*	Yes	Yes (partial)
INTERPOL=NEEDLE I=NEEDLE	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
INTERPOL=NONE I=NONE	Yes	Yes
INTERPOL=R I=R	Yes	Yes (partial)
INTERPOL=SM I=SM	Yes	No
INTERPOL=SPLINE I=SPLINE	Yes	Yes
INTERPOL=STD I=STD	Yes	Yes (partial)
INTERPOL=STEP I=STEP	Yes	Yes
LINE= L=	Yes (GPLOT) No (GCONTOUR)	Yes (GPLOT) No (GCONTOUR)
MODE=	Yes (partial)	Yes (partial)
POINTLABEL=	Yes (partial)	Yes (partial)
REPEAT= R=	Yes (GPLOT) No (GCONTOUR)	Yes (GPLOT) No (GCONTOUR)
STEP= S=	No	No
VALUE= V=	Yes (partial for GPLOT) No (GCONTOUR)	Yes (partial for GPLOT) No (GCONTOUR)
WIDTH= W=	Yes (partial for GPLOT) No (GCONTOUR)	Yes (partial for GPLOT) No (GCONTOUR)

SYMBOL Statement POINTLABEL= Label Description Suboptions

Table A24.9 *ActiveX and Java Support for POINTLABEL Description Suboptions*

Suboption	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR= C=	Yes	No

Suboption	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
FONT= F=	Yes	No
HEIGHT= H=	Yes	No
JUSTIFY= J=	No	No
POSITION=	No	No
"#var" | "#x:#y <$char>" "#y:#x $<char>"	Yes (partial)	Yes (partial)

TITLE and FOOTNOTE Statements (NOTE Statement is not supported)

Table A24.10 *ActiveX and Java Support for the TITLE and FOOTNOTE Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANGLE=	No	No
BCOLOR=	Yes	Yes
BLANK=	No	No
BOX=	No	No
BSPACE=	No	No
COLOR=	Yes	Yes
DRAW=	No	No
FONT=	Yes	Yes
HEIGHT=	Yes (partial)	Yes (partial)
JUSTIFY=	Yes	Yes
LANGLE=	No	No
LINK=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
LSPACE=	No	No
MOVE=	No	No
ROTATE=	No	No
UNDERLIN=	Yes (partial)	Yes (partial)

PROC GAREABAR

Table A24.11 ActiveX and Java Support for PROC GAREABAR

Statement	Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
PROC GAREABAR	DATA=	Yes	No
HBAR and VBAR	CFRFRAME=	Yes	No
	COUTLINE=	Yes	No
	CTEXT=	Yes	No
	DISCRETE	Yes	No
	FRAME NOFRAME	Yes	No
	NAME=	Yes	No
	RSTAT= RESPSTAT= RESPONSESTAT=	Yes	No
	SUBGROUP=	Yes	No
	SUMVAR=	Yes	No
	WSTAT= WIDTHSTAT=	Yes	No

PROC GBARLINE

PROC GBARLINE Statement

Table A24.12 ActiveX and Java Support for the PROC GBARLINE Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	No
DATA=	Yes	No
IMAGEMAP=	No	No

BAR Statement

Table A24.13 ActiveX and Java Support for the PROC GBARLINE BAR Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	No
ASCENDING	Yes	No
AUTOREF	Yes	No
AXIS=	Yes	No
CAUTOREF=	Yes	No
CAXIS=	Yes	No
CERROR=	Yes	No
CFRAME= CFR=	Yes	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CFREQ	Yes	No
CLIPREF	Yes	No
CLM=	Yes	No
COUTLINE=	Yes	No
CPERCENT CPCT	Yes	No
CREF=	Yes	No
CTEXT=	Yes	No
DESCENDING	Yes	No
DESCRIPTION= DES=	Yes	No
DISCRETE	Yes	No
ERRORBAR=	Yes	No
FRAME NOFRAME FR NOFR	Yes	No
FREQ	Yes	No
FREQ=*numeric-variable*	No	No
FRONTREF	Yes	No
HTML=	Yes	No
HTML_LEGEND	No	No
INSIDE=	Yes	No
LAUTOREF=	Yes	No
LEGEND	Yes (partial)	No
LEVELS=	Yes	No
LREF= LR=	Yes	No
MAXIS=	Yes	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
MEAN	Yes	No
MIDPOINTS=*value-list*	Yes	No
MIDPOINTS=OLD	Yes	No
MINOR=	Yes	No
MISSING	Yes	No
NAME=	Yes	No
NOAXIS	Yes	No
NOBASEREF	Yes	No
NOZERO	Yes	No
OUTSIDE=	Yes	No
PATTERNID=	Yes	No
PERCENT PCT	Yes	No
RANGE	Yes	No
RAXIS= AXIS=	Yes (partial)	No
REF=	Yes	No
SPACE=	Yes	No
SUM	Yes	No
SUMVAR=	Yes	No
TYPE=	Yes	No
URL=	No	Yes
WIDTH=	Yes	No
WOUTLINE=	Yes	No

PLOT Statement

Table A24.14 ActiveX and Java Support for the PROC GBARLINE PLOT Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ASCENDING	Yes	No
AXIS=	Yes	No
FREQ=*numeric-variable*	No	No
HTML=	No	No
MINOR=	Yes	No
NOLINE	Yes	No
NOMARKER	Yes	No
RAXIS=AXIS=	Yes	No
SUMVAR=	Yes	No
TYPE=	Yes	No

PROC GCHART

PROC GCHART Statement

Table A24.15 ActiveX and Java Support for the PROC GCHART Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
DATA=	Yes	Yes
GOUT=	Yes	Yes
IMAGEMAP=	No	No

BLOCK Statement

Table A24.16 *ActiveX and Java Support for the PROC GCHART BLOCK Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes
BLOCKMAX=	No	No
CAXIS=	Yes	Yes
COUTLINE=	Yes (partial)	Yes (partial)
CTEXT=	Yes	Yes
DESCRIPTION= DES=	Yes	Yes
DISCRETE	Yes	Yes
FREQ=	Yes	Yes
G100	Yes	Yes
GROUP=	Yes	Yes
HTML=	Yes	Yes
HTML_LEGEND=	No	No
LEGEND=	Yes (partial)	Yes (partial)
LEVELS=	Yes	Yes
MIDPOINTS=*value-list*	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
MIDPOINTS=OLD	Yes	Yes
MISSING	Yes	Yes
NAME=	Yes	Yes
NOHEADING	No	No
NOLEGEND	Yes	Yes
PATTERNID=	Yes	Yes
SUBGROUP=	Yes	Yes
SUMVAR=	Yes	Yes
TYPE=	Yes	Yes
URL=	Yes	Yes
WOUTLINE=	Yes	No

HBAR, HBAR3D, VBAR, and VBAR3D Statements

Table A24.17 *ActiveX and Java Support for PROC GCHART HBAR, HBAR3D, VBAR, and VBAR3D Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes
ASCENDING	Yes	Yes
AUTOREF	Yes	Yes
AXIS=	Yes	Yes
CAUTOREF=	Yes	Yes
CAXIS=	Yes	Yes
CFRAME= CFR=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CERROR=	Yes	Yes
CFREQ	Yes	Yes
CFREQLABEL=	No	No
CLIPREF	Yes	Yes
CLM=	Yes	Yes
COUTLINE=	Yes	Yes
CPERCENT CPCT	Yes	Yes
CPERCENTLABEL=	No	No
CREF=	Yes	Yes
CTEXT=	Yes	Yes
DESCENDING	Yes	Yes
DESCRIPTION= DES=	Yes	Yes
DISCRETE	Yes	Yes
ERRORBAR=	Yes	Yes
FRAME NOFRAME FR NOFR	Yes	Yes
FREQ	Yes	Yes
FREQLABEL=	No	No
FREQ=*numeric-variable*	Yes	Yes
FRONTREF	Yes	Yes
G100	Yes	Yes
GAXIS=	Yes (partial)	Yes (partial)
GROUP=	Yes	Yes
GSPACE=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
HTML=	Yes	Yes
HTML_LEGEND=	No	No
IFRAME=	Yes	No
IMAGESTYLE=	Yes	No
INSIDE=	Yes	Yes
LAUTOREF=	Yes	Yes
LEGEND=	Yes	Yes
LEVELS=	Yes	Yes
LREF= LR=	Yes	No
MAXIS=	Yes (partial)	Yes (partial)
MEAN	Yes	Yes
MEANLABEL=	No	No
MIDPOINTS=*value-list*	Yes	Yes
MIDPOINTS=OLD	Yes	Yes
MINOR=	Yes	Yes
MISSING	Yes	Yes
NAME=	Yes	Yes
NOAXIS	Yes	Yes
NOBASEREF	Yes	Yes
NOLEGEND	Yes	Yes
NOSTATS	Yes	No
NOZERO	Yes	Yes
OUTSIDE=	Yes	Yes
PATTERNID=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
PERCENT PCT	Yes	Yes
PERCENTLABEL=	No	No
PERCENTSUM PCTSUM	Yes (partial)	No
RANGE	Yes	Yes
RAXIS= AXIS=	Yes (partial)	Yes (partial)
REF=	Yes	Yes
SHAPE=	Yes	Yes
SPACE=	Yes	Yes
SUBGROUP=	Yes	Yes
SUM	Yes	Yes
SUMLABEL=	No	No
SUMVAR=	Yes	Yes
TYPE=	Yes	Yes
URL=	Yes	Yes
WIDTH=	Yes	Yes
WOUTLINE=	Yes	No

PIE, PIE3D, and DONUT Statements

Table A24.18 *ActiveX and Java Support for the PROC GCHART PIE, PIE3D, and DONUT Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ACROSS=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANGLE=	Yes	Yes
ANNOTATE= ANNO=	Yes	Yes
ASCENDING	Yes	Yes
CFILL=	Yes	Yes
CLOCKWISE	Yes	Yes
COUTLINE=	Yes	Yes
CTEXT=	Yes	Yes
DESCENDING	Yes	Yes
DESCRIPTION= DES=	Yes	Yes
DETAIL=	Yes	Yes
DETAIL_PERCENT=	Yes	Yes
DETAIL_RADIUS=	Yes	Yes
DETAIL_SLICE=	Yes	Yes
DETAIL_THRESHOLD=	Yes	Yes
DETAIL_VALUE=	Yes	Yes
DISCRETE	Yes	Yes
DONUTPCT=	Yes	Yes
DOWN=	Yes	Yes
EXPLODE=	Yes	Yes
FILL=	Yes (partial)	Yes (partial)
FREQ=	Yes	Yes
GROUP=	Yes	Yes
HTML=	Yes	Yes
HTML_LEGEND=	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
INVISIBLE=	Yes	Yes
JSTYLE	Yes	Yes
LABEL=	Yes (partial)	Yes (partial)
PLABEL=	Yes (partial)	Yes (partial)
PPERCENT=	Yes (partial)	Yes (partial)
LEGEND=	Yes (partial)	Yes (partial)
LEVELS=	Yes	Yes
MATCHCOLOR	Yes	Yes
MIDPOINTS=*value-list*	Yes	Yes
MIDPOINTS=OLD	Yes	Yes
MISSING	Yes	Yes
NAME=	Yes	Yes
NOGROUPHEADING	Yes	Yes
NOHEADING	No	No
NOLEGEND	Yes	Yes
OTHER=	Yes	Yes
OTHERCOLOR=	Yes	Yes
OTHERLABEL=	Yes	Yes
PERCENT=	Yes	Yes
SLICE=	Yes	Yes
SUBGROUP=	Yes	Yes
SUMVAR=	Yes	Yes
TYPE=	Yes	Yes
URL=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
VALUE=	Yes	Yes
WOUTLINE=	Yes	No

DONUT Statement Text Description Suboptions

Text description suboptions are used by the LABEL= option in the DONUT statement.

Table A24.19 *ActiveX and Java Support for PROC GCHART DONUT Statement Text Description Suboptions*

Suboption	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANGLE= A=	Yes	No
COLOR= C=	Yes	Yes
FONT= F=	Yes (partial)	Yes (partial)
HEIGHT= H=	Yes	Yes
JUSTIFY= J=	No	No
ROTATE= R=	Yes	No

STAR Statement

The GCHART procedure STAR statement is not supported by ActiveX and Java.

PROC GCONTOUR

PROC GCONTOUR Statement

Table A24.20 ActiveX and Java Support for the PROC GCONTOUR Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes (partial)	Yes (partial)
DATA=	Yes	Yes
GOUT=	No	No
INCOMPLETE	No	No

PLOT Statement

Table A24.21 ActiveX and Java Support for the PROC GCONTOUR PLOT Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes (partial)	Yes (partial)
AUTOHREF	Yes	No
AUTOLABEL=	No	No
AUTOVREF	Yes	No
CAUTOHREF=	Yes	No
CAUTOVREF=	Yes	No
CAXIS=	Yes	Yes (partial)

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CFRAME= CFR=	No	No
CHREF= CH=	Yes	No
CLEVELS=	Yes (partial)	No
COUTLINE=	No	No
CTEXT=	Yes	Yes
CVREF= CV=	Yes	No
DESCRIPTION= DES=	Yes	Yes
GRID	Yes	No
HAXIS=	Yes (partial)	Yes (partial)
HMINOR= HM=	Yes	No
HREF=	Yes	No
HREVERSE=	Yes	No
JOIN	Yes (partial)	Yes (partial)
LAUTOHREF=	Yes	No
LAUTOVREF=	Yes	No
LEGEND=	Yes (partial)	Yes (partial)
LEVELS=	Yes	Yes
LHREF= LH=	Yes (partial)	No
LLEVELS=	Yes	No
LVREF= LV=	Yes (partial)	Yes (partial)
NAME=	Yes	Yes
NLEVELS=	Yes	Yes
NOAXIS NOAXES	Yes	Yes
NOFRAME	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
NOLEGEND	Yes	Yes
PATTERN	Yes (partial)	Yes (partial)
VAXIS=	Yes (partial)	Yes (partial)
VMINOR= VM=	Yes	Yes
VREF=	Yes	No
VREVERSE	Yes	No
XTICKNUM= YTICKNUM=	Yes	Yes

PROC GMAP

PROC GMAP Statement

Table A24.22 *ActiveX and Java Support for the PROC GMAP Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
MAP=	Yes	Yes
ALL	Yes	Yes
ANNOTATE=	Yes	Yes
DATA=	Yes	Yes
GOUT=	No	No
IMAGEMAP=	No	No
STRETCH	No	No
UNIFORM	Yes	No

AREA Statement

Table A24.23 ActiveX and Java Support for the PROC GMAP AREA Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
DISCRETE	Yes	Yes
LEGEND=	Yes (partial)	Yes (partial)
LEVELS=	Yes	Yes
MIDPOINTS=	Yes	Yes (partial)
MISSING	Yes	Yes
NOLEGEND	Yes	Yes
PERCENT	Yes	Yes
RANGE	Yes	Yes
STATFMT=	Yes	Yes
STATISTIC=	Yes	Yes
UNIFORM	Yes	No

ID Statement

The GMAP procedure ID statement is supported by ActiveX and Java. It has no options.

BLOCK Statement

Table A24.24 ActiveX and Java Support for the PROC GMAP BLOCK Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE=	Yes	Yes
AREA=	Yes	Yes
BLOCKSIZE=	Yes	Yes
CBLKOUT=	Yes	Yes
CDEFAULT=	Yes	No
CEMPTY=	Yes	No
COUTLINE=	Yes	Yes
CTEXT=	Yes	Yes
DESCRIPTION=	Yes	Yes
DISCRETE	Yes	Yes
HTML=	Yes	Yes
HTML_LEGEND=	No	No
LEGEND=	Yes (partial)	Yes (partial)
LEVELS=	Yes	Yes
MIDPOINTS=	Yes	Yes (partial)
MISSING	Yes	Yes
NAME=	Yes	Yes
NOLEGEND	Yes	Yes
PERCENT	Yes	Yes
RANGE	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
RELZERO	Yes	No
SHAPE=	Yes	Yes
STATISTIC=	Yes	Yes
STRETCH	No	No
UNIFORM	Yes	No
URL=	Yes	Yes
WOUTLINE=	Yes	Yes
XSIZE= YSIZE=	No	No
XVIEW= YVIEW= ZVIEW=	Yes	Yes (partial)

CHORO Statement

Table A24.25 *ActiveX and Java Support for the PROC GMAP CHORO Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE=	Yes	Yes
CDEFAULT=	Yes	No
CEMPTY=	Yes	No
COUTLINE=	Yes	Yes
CTEXT=	Yes	Yes
DESCRIPTION=	Yes	Yes
DISCRETE	Yes	Yes
HTML=	Yes	Yes
HTML_LEGEND=	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
LEGEND=	Yes (partial)	Yes (partial)
LEVELS=	Yes	Yes
MIDPOINTS=	Yes	Yes (partial)
MISSING	Yes	Yes
NAME=	Yes	Yes
NOLEGEND	Yes	Yes
OSM	No	Yes
PERCENT	Yes	Yes
RANGE	Yes	Yes
STATFMT=	Yes	Yes
STATISTIC=	Yes	Yes
STRETCH	No	No
UNIFORM	Yes	No
URL=	Yes	Yes
WOUTLINE=	Yes	Yes
XSIZE= YSIZE=	No	No

PRISM Statement

Table A24.26 *ActiveX and Java Support for the PROC GMAP PRISM Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE=	Yes	Yes
AREA=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CDEFAULT=	Yes	No
CEMPTY=	Yes	No
COUTLINE=	Yes	Yes
CTEXT=	Yes	Yes
DESCRIPTION=	Yes	Yes
DISCRETE	Yes	Yes
HTML=	Yes	Yes
HTML_LEGEND=	No	No
LEGEND=	Yes (partial)	Yes (partial)
LEVELS=	Yes	Yes
MIDPOINTS=	Yes	Yes (partial)
MISSING	Yes	Yes
NAME=	Yes	Yes
NOLEGEND	Yes	Yes
PERCENT	Yes	Yes
RANGE	Yes	Yes
RELZERO	Yes	No
STATFMT=	Yes	Yes
STATISTIC=	Yes	Yes
STRETCH	No	No
UNIFORM	Yes	No
URL=	Yes	Yes
WOUTLINE=	No	Yes
XLIGHT= YLIGHT=	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
XSIZE= YSIZE=	No	No
XVIEW= YVIEW= ZVIEW=	Yes	Yes (partial)

SURFACE Statement

The GMAP procedure SURFACE statement is not supported by ActiveX and Java.

PROC GPLOT

PROC GPLOT Statement

Table A24.27 *ActiveX and Java Support for the PROC GPLOT Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes
DATA=	Yes	Yes
GOUT=	Yes	Yes
IMAGEMAP=	No	No
UNIFORM	Yes (partial)	No

BUBBLE Statement

Table A24.28 ActiveX and Java Support for the PROC GPLOT BUBBLE Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes
AUTOHREF	Yes	Yes
AUTOVREF	Yes	Yes
BCOLOR=	Yes	Yes
BFILL=	No	No
BFONT=	No	No
BLABEL	Yes	Yes
BSCALE=	No	No
BSIZE=	Yes (partial)	Yes (partial)
CAUTOHREF=	Yes	Yes
CAUTOVREF=	Yes	Yes
CAXIS= CA=	Yes	Yes
CFRAME= CFR=	Yes	Yes
CHREF= CH=	Yes	Yes
CTEXT= C=	Yes	Yes
CVREF= CV=	Yes	Yes
DESCRIPTION= DES=	Yes	Yes
FRAME NOFRAME FR NOFR	Yes	Yes
GRID	Yes	Yes
HAXIS=	Yes (partial)	Yes (partial)

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
HMINOR= HM=	Yes	Yes
HREF=	Yes	Yes
HTML=	Yes	Yes
HREVERSE	Yes (partial)	Yes (partial)
HZERO	Yes	Yes
IFRAME=	Yes	No
IMAGESTYLE=	Yes	Yes
LAUTOHREF=	Yes	Yes
LAUTOVREF=	Yes	Yes
LHREF= LH=	Yes	Yes
LVREF= LV=	Yes	Yes
NAME=	Yes	Yes
NOAXIS NOAXES	Yes	Yes
URL=	Yes	Yes
VAXIS=	Yes (partial)	Yes (partial)
VMINOR= VM=	Yes	Yes
VREF=	Yes	Yes
VREVERSE	Yes	Yes
VZERO	Yes	Yes

BUBBLE2 Statement

Table A24.29 ActiveX and Java Support for the PROC GPLOT BUBBLE2 Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes
AUTOVREF	Yes	Yes
BCOLOR=	Yes	Yes
BFILL=	No	No
BFONT=	No	No
BLABEL	Yes	Yes
BSCALE=	No	No
BSIZE=	Yes (partial)	Yes (partial)
CAUTOVREF=	Yes	Yes
CAXIS= CA=	Yes	Yes
CFRAME= CFR=	Yes	Yes
CTEXT= C=	Yes	Yes
CVREF= CV=	Yes	Yes
FRAME NOFRAME FR NOFR	Yes	Yes
GRID	Yes	Yes
HAXIS	Yes (partial)	Yes (partial)
HREVERSE	Yes (partial)	Yes (partial)
HTML=	Yes (partial)	No
IFRAME	Yes	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
LAUTOVREF=	Yes	Yes
LVREF= LV=	Yes	Yes
NOAXIS NOAXES	Yes	Yes
URL=	Yes (Partial)	Yes (Partial)
VAXIS=	Yes (partial)	Yes (partial)
VMINOR= VM=	Yes	Yes
VREF=	Yes	Yes
VREVERSE	Yes	Yes
VZERO	Yes	Yes

PLOT Statement

Table A24.30 *ActiveX and Java Support for the PROC GPLOT PLOT Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE=	Yes	Yes
AREAS=	Yes	Yes (partial)
AUTOHREF	Yes	Yes
AUTOVREF	Yes	Yes
CAUTOHREF=	Yes	Yes
CAUTOVREF=	Yes	Yes
CAXIS= CA=	Yes	Yes
CBASELINE= CB=	No	No
CFRAME= CFR=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CHREF= CH=	Yes	Yes
COUTLINE=	Yes	No
CTEXT= C=	Yes	Yes
CVREF= CV=	Yes	Yes
DESCRIPTION= DES=	Yes	Yes
FRAME NOFRAME FR NOFR	Yes	Yes
GRID	Yes	Yes
HAXIS=	Yes (partial)	Yes (partial)
HMINOR= HM=	Yes	Yes
HREF=	Yes	Yes
HREVERSE	Yes (partial)	Yes (partial)
HTML=	Yes (partial)	Yes (partial)
HTML_LEGEND=	No	No
HZERO	Yes	Yes
IFRAME=	Yes	No
IMAGESTYLE=	Yes	No
LAUTOHREF=	Yes	Yes
LAUTOVREF=	Yes	Yes
LEGEND=	Yes	Yes
LHREF= LH=	Yes	Yes
LVREF= LV=	Yes (partial)	Yes (partial)
NAME=	Yes	Yes
NOAXIS NOAXES	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
NOLEGEND	Yes	Yes
OVERLAY	Yes	Yes (partial)
REGEQN	No	Yes
SKIPMISS	Yes	Yes
URL=	Yes	Yes
VAXIS=	Yes (partial)	Yes (partial)
VMINOR= VM=	Yes	Yes
VREF=	Yes	Yes
VREVERSE	Yes	Yes
VZERO	Yes	Yes

PLOT2 Statement

Table A24.31 *ActiveX and Java Support for the PROC GPLOT PLOT2 Statement Options*

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
With INTERPOL= BOX, HILO, or STD	No	No
ANNOTATE= ANNO=	Yes	Yes
AREAS=	Yes	Yes (partial)
AUTOVREF	Yes	Yes
CAUTOVREF=	Yes	Yes
CAXIS= CA=	Yes	Yes
CBASELINE= CB=	No	No
CFRAME= CFR=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COUTLINE=	Yes	No
CTEXT= C=	Yes	Yes
CVREF= CV=	Yes	Yes
FRAME NOFRAME FR NOFR	Yes	Yes
GRID	Yes	Yes
HTML=	Yes (partial)	No
HTML_LEGEND=	No	No
LAUTOVREF=	Yes	Yes
LEGEND=	Yes	Yes
LVREF= LV=	Yes	Yes
NAME=	Yes	Yes
NOAXIS NOAXES	Yes	Yes
NOLEGEND	Yes	Yes
OVERLAY	Yes	Yes (partial)
REGEQN	No	Yes
SKIPMISS	Yes	Yes
URL=	Yes (Partial)	Yes (Partial)
VAXIS=	Yes (partial)	Yes (partial)
VMINOR= VM=	Yes	Yes
VREF=	Yes	Yes
VREVERSE	Yes	Yes
VZERO	Yes	Yes

PROC GRADAR

PROC GRADAR Statement

Table A24.32 ActiveX and Java Support for the PROC GRADAR Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE=	Yes	No
DATA=	Yes	No
GOUT=	Yes	No

CHART Statement

Table A24.33 ActiveX and Java Support for the PROC GRADAR CHART Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ACROSS= ACROSSVAR=	Yes	No
ANNOTATE= ANNO=	No	No
CALENDAR=	Yes	No
CAXIS= CAXES= CA=	No	No
CFRAME= CFR=	Yes	No
CFRAMESIDE=	Yes	No
CFRAMETOP=	Yes	No
CSPOKES= CSPOKE=	Yes	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CSTARCIRCLES= CSTARCIRCLE=	Yes	No
CSTARFILL=	Yes	No
CSTARS= CSTAR=	Yes	No
CTEXT=	Yes	No
CTILES= CTILE=	No	No
DESCRIPTION= DES=	Yes	No
DOWN= DOWNVAR=	Yes	No
FONT=	Yes	No
FRAME	Yes	No
FREQ=	Yes	No
HEIGHT= HLABEL=	Yes	No
HTML=	Yes	No
HTML_LEGEND=	Yes	No
IFRAME=	No	No
IMAGESTYLE=	Yes	No
INBORDER	No	No
INHEIGHT=	No	No
INTERTILE= INTERCHART=	Yes	No
LSPOKEs=	Yes	No
LSTARCIRCLES= LSTARCIRCLE=	Yes	No
LSTARS= LSTAR=	Yes	No
MAXNVERT= MAXVERT=	Yes	No
MISSING	No	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
MODE=	Yes	No
NAME=	Yes	No
NCOLS= NCOL=	No	No
NLEVELS=	Yes	No
NOLEGEND	Yes	No
NOZEROREF	Yes	No
NROWS= NROW=	No	No
ORDERACROSS=	No	No
OTHER=	Yes	No
OVERLAY=OVERLAYVAR =	Yes	No
SPEED	Yes	No
SPIDERWEB SPIDER	Yes	No
SPKLABEL=	No	No
SPOKESCALE=	Yes	No
STARAXIS= STARAXES=	Yes	No
STARCIRCLES= STARCIRCLE=	Yes	No
STARFILL=	Yes	No
STARINRADIUS=	No	No
STARLEGEND=	Yes	No
STARLEGENDLAB=	Yes	No
STAROUTRADIUS=	No	No
STARSTART=	Yes	No
STARTYPE=	Yes	No

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
SUMVAR=	Yes	No
TILELEGEND=	No	No
TILELEGLABEL=	No	No
URL=	Yes (partial)	No
WFRAME= WAXIS=	No	No
WINDROSE	Yes	No
WINDROSECIRCLES=	Yes	No
WSPOKES= WSPOKE=	Yes	No
WSTARCIRCLES= WSTARCIRCLE=	Yes	No
WSTARS= WSTAR=	Yes	No

PROC GTILE

Table A24.34 *ActiveX and Java Support for PROC GTILE*

Statement	Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
PROC GTILE	DATA=	Yes	Yes
FLOW, TILE, and TOGGLE	TILEBY	Yes	Yes
	BASELINE	Yes	Yes
	CDEFAULT=	Yes (partial)	Yes
	COLORRAMP=	Yes	Yes
	COLORVAR=	Yes	Yes
	DESCRIPTION=	Yes	Yes

Statement	Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
	DETAILLEVEL=	Yes	Yes
	LABELLEVEL=	Yes	Yes
	NAME=	Yes	Yes

PROC G3D

PROC G3D Statement

Table A24.35 ActiveX and Java Support for the PROC G3D Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes
DATA=	Yes	Yes
GOUT=	Yes	Yes

PLOT Statement

Table A24.36 ActiveX and Java Support for the PROC G3D PLOT Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes
CAXIS=	Yes	Yes

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CBOTTOM=	Yes	No
CTEXT=	Yes	Yes
CTOP=	Yes	Yes
DESCRIPTION= DES=	Yes (partial)	Yes (partial)
GRID	Yes	No
NAME=	Yes	Yes
NOAXIS NOAXES	Yes	Yes
NOLABEL	Yes	Yes
ROTATE=	No	Yes (partial)
SIDE	Yes	Yes
TILT=	Yes (partial)	Yes (partial)
XAXIS	Yes (partial)	Yes (partial)
XTICKNUM=	Yes (partial)	No (partial)
XYTYPE	Yes	No
YAXIS=	Yes (partial)	Yes (partial)
YTICKNUM=	Yes	No
ZAXIS	Yes (partial)	No (partial)
ZMAX=	Yes	No
ZMIN=	Yes	No
ZTICKNUM=	Yes (partial)	No

SCATTER Statement

Table A24.37 ActiveX and Java Support for the PROC G3D SCATTER Statement Options

Option	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANNOTATE= ANNO=	Yes	Yes
CAXIS=	Yes	Yes
COLOR=	Yes	Yes
CTEXT=	Yes	Yes
DESCRIPTION= DES=	Yes	Yes
GRID	Yes	Yes
NAME=	Yes	Yes
NOAXIS NOAXES	Yes	Yes
NOLABEL	Yes	Yes
NONEEDLE	Yes	Yes
ROTATE=	No	No
SHAPE=	Yes	Yes
SIZE=	Yes	Yes
TILT=	Yes	No
XTICKNUM= YTICKNUM= ZTICKNUM=	Yes	Yes
ZMAX= ZMIN=	Yes	Yes (partial)

Annotate Functions

ARROW Function

Table A24.38 ActiveX and Java Support for the ARROW Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANGLE	Yes	Yes
COLOR	Yes	Yes
GROUP	Yes	Yes
HSYS	Yes	Yes
LINE	Yes	Yes
MIDPOINT	Yes	Yes
SIZE	Yes	Yes
STYLE	Yes	No
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

BAR Function

Table A24.39 ActiveX and Java Support for the BAR Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR	Yes	Yes
GROUP	Yes	Yes
HTML	Yes	No
LINE	Yes	Yes
MIDPOINT	Yes	Yes
SIZE	Yes	Yes
STYLE	Yes (Partial)	Yes (Partial)
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

DRAW Function

Table A24.40 ActiveX and Java Support for the DRAW Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR	Yes	Yes
GROUP	Yes	Yes
HSYS	Yes	Yes

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
LINE	Yes	Yes
MIDPOINT	Yes	Yes
SIZE	Yes	Yes
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

DRAW2TXT Function

Table A24.41 *ActiveX and Java Support for the DRAW2TXT Function Options*

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR	Yes	Yes
HSYS	Yes	Yes
LINE	Yes	Yes
SIZE	Yes	Yes
WHEN	Yes	Yes

FRAME Function

Table A24.42 ActiveX and Java Support for the FRAME Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR	Yes	No
HSYS	Yes	No
HTML	Yes	No
LINE	Yes	No
SIZE	Yes	No
STYLE	Yes (Partial)	No
WHEN	Yes	No
XSYS, YSYS	Yes	No

IMAGE Function

Table A24.43 ActiveX and Java Support for the IMAGE Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
HTML	Yes	Yes
IMGPATH	Yes	Yes
STYLE	Yes	Yes
WHEN	Yes	Yes
X, Y	Yes	Yes
XSYS, YSYS	Yes	Yes

LABEL Function

Table A24.44 ActiveX and Java Support for the LABEL Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANGLE	Yes	No
CBORDER	Yes	Yes
CBOX	Yes	Yes
COLOR	Yes	Yes
GROUP	Yes	Yes
HSYS	Yes	No
HTML	Yes	No
MIDPOINT	Yes	Yes
POSITION	Yes	Yes (Partial)
ROTATE	Yes	No
SIZE	Yes	Yes
STYLE	Yes	No
SUBGROUP	Yes	Yes
TEXT	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

MOVE Function

Table A24.45 ActiveX and Java Support for the MOVE Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
GROUP	Yes	Yes
MIDPOINT	Yes	Yes
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

PIE Function

Table A24.46 ActiveX and Java Support for the PIE Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANGLE	Yes	Yes
COLOR	Yes	Yes
GROUP	Yes	Yes
HSYS	Yes	Yes
HTML	Yes	No
LINE	Yes	Yes
MIDPOINT	Yes	Yes
ROTATE	Yes	Yes

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
SIZE	Yes	Yes
STYLE	Yes (Partial)	Yes (Partial)
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
WIDTH	Yes (Partial)	Yes (Partial)
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

PIECNTR Function

Table A24.47 *ActiveX and Java Support for the PIECNTR Function Options*

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
GROUP	Yes	Yes
HSYS	Yes	Yes
MIDPOINT	Yes	Yes
SIZE	Yes	Yes
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

PIEXY Function

Table A24.48 ActiveX and Java Support for the PIEXY Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
ANGLE	Yes	Yes
SIZE	Yes	Yes
WHEN	Yes	Yes

POINT Function

Table A24.49 ActiveX and Java Support for the POINT Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR	Yes	Yes
GROUP	Yes	Yes
MIDPOINT	Yes	Yes
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes (Partial)
XC, YC	Yes	Yes (Partial)
XSYS, YSYS, ZSYS	Yes	Yes (Partial)

POLY Function

Table A24.50 ActiveX and Java Support for the POLY Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR	Yes	Yes
GROUP	Yes	Yes
HTML	Yes	No
LINE	Yes	No
MIDPOINT	Yes	Yes
SUBGROUP	Yes	Yes
SIZE	Yes	Yes
STYLE	Yes (Partial)	Yes (Partial)
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

POLYCONT Function

Table A24.51 ActiveX and Java Support for the POLYCONT Function Options

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
COLOR	Yes	Yes
GROUP	Yes	Yes
MIDPOINT	Yes	Yes

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

SYMBOL Function

Table A24.52 *ActiveX and Java Support for the SYMBOL Function Options*

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
CBOX	Yes	No
CBORDER	Yes	No
COLOR	Yes	Yes
GROUP	Yes	Yes
SUBGROUP	Yes	Yes
HSYS	Yes	Yes
HTML	Yes	No
MIDPOINT	Yes	Yes
SIZE	Yes	Yes
STYLE	Yes(Partial)	Yes (Partial)
TEXT	Yes (Partial)	Yes (Partial)
WHEN	Yes	Yes
X, Y, Z	Yes	Yes

Variable	Supported by ACTIVEX and ACTXIMG?	Supported by JAVAIMG?
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

49

ActiveX Control Parameter Dictionary

Specifying Parameters and Attributes for ActiveX

About Specifying Parameters in ODS Statements

You can specify attributes and parameters in ODS to override default values in Java and ActiveX. No attributes or parameters are required. SAS provides workable defaults.

Attributes can be any HTML name/value pair that is valid inside the initial (opening) OBJECT tag. Parameters are values that appear in the body of the OBJECT tag to configure the appearance or functionality of the ActiveX control.

Attributes and parameters are specified as options of one of the available ODS statements, such as ODS HTML:

ODS HTML

<ATTRIBUTES=("*attr-name*"="*attr-value*")>

<PARAMETERS=("*param-name*"="*param-value*")>

<*other-options*>;

The preceding syntax applies to all applicable ODS statements, such as HTML, MARKUP, PDF, PS, and RTF.

You can specify more than one "*name*"="*value*" pair (separated by blank spaces) inside the parenthesis of an ATTRIBUTES= or PARAMETERS= option. You can also specify multiple ATTRIBUTES= and PARAMETERS= options in a single ODS statement. These options can be specified in any order in the ODS statement.

You can remove a parameter tag by specifying a $ for its value, or by setting it to None using the menu of the ActiveX Control. This removes the data and axis label that would otherwise be included in the graph.

You can also append `,n` to tags that reference variables whose values are URLs. Normally, the substitution string is URL-encoded for browsers that do not support embedded white space in URL strings. Use `,n` to prevent this encoding.

No intervening white space should be added between the primary tag and the appended `,f` or `,n` characters. For information about the `,f` characters, see "Formatting Data Values in Script Drill-Down Mode" on page 216.

For an example of how to specify parameters, see "Examples: Creating Interactive ActiveX Output" on page 217.

For information about other ODS statement options, see the *SAS Output Delivery System: User's Guide*.

In HTML output that runs the ActiveX Control, all values of the ATTRIBUTES= option appear in the opening OBJECT tag. For example, a SAS/GRAPH program can specify the WIDTH attribute as follows:

```
ods html file="C:\sashtml\piechart.htm"
```

```
   attributes=("width"="720");
```

In the HTML output file, the WIDTH attribute appears inside the beginning OBJECT tag as shown in the following:

```
<script language="javascript" type="text/javascript">
<!--
   document.writeln("<OBJECT");
   document.writeln('style=" width: 720px; height: 480px;
        background-color: #4E5056; border-width: 0px;"');
   document.writeln("ALIGN=\"baseline\" class=\"Graph\"");
   .
   .
   .
//-->
</script>
```

Valid attribute names are those that are enabled for the OBJECT tag in HTML. Valid attributes must also be specified as required by the ACTIVEX device driver that runs in the operating environment.

All of the name/value pairs that are specified in the ODS statement option PARAMETERS= appear in the body of the OBJECT tag. For example, a SAS program can disable the data tips and set the background color for a graph as follows:

```
ods html file="test.html" parameters=(
   "tips"="none"
   "backdropcolor"="CXff0000"
);
```

Valid parameter values for the ActiveX control are defined in “Attributes and Parameters That Apply to the ActiveX Control” on page 1646.

Specifying the Location of the ActiveX Control Files (CODEBASE= Option)

About the Control Files

When you generate web presentations with the ACTIVEX device driver, the SAS/GRAPH software generates HTML pages that automatically look for the ActiveX control file in the default installation location. If you install the ActiveX control .exe file in a location other than the default or if you want to publish Output Delivery System (ODS) output containing the SAS/GRAPH control in a web server, then you might need to specify the location of the .exe file when you generate your web presentation.

You can use the CODEBASE= option to specify the location of the ActiveX control.

Note: The ActiveX control must be installed locally on each PC where the web presentation is viewed.

Specifying the Location of the ActiveX Control

If you use the ACTIVEX device driver to generate output containing an ActiveX control, then specify the location and version of the .exe file with the CODEBASE= option in the ODS statement. Specify the directory and file name of the .exe file. The CODEBASE location can be specified as a pathname or as a URL. (See "Specifying the CODEBASE= URL" on page 1646 for more information.) If previous versions of the ActiveX control have been installed, then you also need to specify the version that you want to use. For example, if your .exe file is sasgraph.exe in directory **/sasweb/graph**, specify the following:

```
ods html file="/path/to/mygraph.html"
   codebase="/sasweb/graph/sasgraph.exe#version=9,4";
```

Specifying the CODEBASE= URL

If the value that you specify for CODEBASE= is a URL, it can be a full URL (for example, http://your_server/sasweb/graph), or it can be relative to your web server (**/sasweb/graph**). If you are publishing HTML only on web servers where the ActiveX Control is installed in a common location, then it is generally recommended that you use the shorter, relative URL. A relative URL enables you to move the HTML to any web server without modifying the HTML (assuming that the ActiveX Control is installed on that server).

Attributes and Parameters That Apply to the ActiveX Control

The following table lists the parameters that apply to the ActiveX Control.

Table A25.1 *Attributes and Parameters Enabled for the ActiveX Control*

Parameter	Description
AMBIENT (p. 1648)	Specifies the intensity of non-directional ambient light in relation to direct light.
BACKIMAGE (p. 1649)	Specifies the URL of an image that is to be displayed as the background of the image area.[1]

Parameter	Description
COLORSCHEME (p. 1649)	Specifies the name of the color scheme that is applied to the graph.
DIRECT (p. 1649)	Specifies the intensity of direct light (from a light source) in relation to the ambient (non-directional) light.
DRILLDOWNFUNCTION (p. 1650)	Specifies the name of the JavaScript function that is called in Script drill-down mode.
DRILLDOWNMODE (p. 1650)	Specifies the drill-down mode.
DRILLPATTERN (p. 1650)	Specifies how to construct the drill-down URL when the drill-down mode is HTML.
DRILLTARGET (p. 1651)	Specifies where the drill-down destination is displayed in the browser.
G_COLOR (p. 1651)	Specifies a new color variable for the current drill-down level.
G_COLORV (p. 1651)	Specifies that the current color variable is the same variable that was used to configure the previous drill-down level.
G_DEP (p. 1652)	Specifies a new dependent variable for the current drill-down level.
G_DEPTH (p. 1652)	Specifies a new depth variable for the current drill-down level.
G_DEPTHV (p. 1652)	Specifies that the depth variable for the current drill-down level is the same depth variable that was used in the previous drill-down level.
G_DEPV (p. 1653)	Specifies that the drill-down graph at the specified drill-down level is to use the same dependent variable that was used in the previous drill-down level.
G_GROUP (p. 1653)	Specifies a new group variable for the current drill-down level.
G_GROUPV (p. 1653)	Specifies that this group variable should be the same group variable that was used at the previous drill-down level.
G_INDEP (p. 1653)	Specifies a new independent variable for the current drill-down level.

Parameter	Description
G_INDEPV (p. 1654)	Specifies that an independent variable at the current drill-down level is the same variable that was used at the previous drill-down level.
G_SUBGR (p. 1654)	Specifies a new subgroup variable for the current drill-down level.
G_SUBGRV (p. 1654)	Specifies that a subgroup variable at this drill-down level is the same subgroup variable that was used at the previous drill-down level.
PATTERNSTRIP (p. 1655)	Removes preceding and trailing white space from drill-down substitution patterns before the substituted text is added into a dynamically generated drill-down URL.
TIPS (p. 1655)	Specifies whether to display data tips.
TIPMODE (p. 1655)	Specifies which of two types of data tips are to be displayed.
VIEW2D (p. 1656)	Indicates whether the view point should be locked to two dimensions.

1 This option works only if STYLE=MINIMAL is specified in the ODS destination statement.

Dictionary

AMBIENT Parameter

specifies the intensity of non-directional ambient light in relation to direct light.

Default: 0.4

Range: 0.0–1.0

Restriction: The sum of the AMBIENT= and DIRECT= parameter values cannot exceed 1.0.

Interaction: If the sum of the AMBIENT= and DIRECT= parameter values exceeds 1.0, the AMBIENT= value is reduced so that the sum of the two values equals 1.0.

Syntax

AMBIENT=*light-level*

BACKIMAGE Parameter

specifies the URL of an image that is to be displayed as the background of the image area.

Default: The background is drawn in a single solid color.

Requirement: The background image must be in GIF, JPEG, or BMP format.

Syntax

BACKIMAGE=*image-URL*

COLORSCHEME Parameter

specifies the name of the color scheme that is applied to the graph.

Default: No color scheme is applied to the graph.

Syntax

COLORSCHEME=*scheme-name*

DIRECT Parameter

specifies the intensity of direct light (from a light source) in relation to the ambient (non-directional) light.

Default: 0.6

Range: 0.0–1.0

Restriction: The sum of AMBIENT= and DIRECT= parameter values cannot exceed 1.0.

Interaction: If the sum of the AMBIENT= and DIRECT= parameter values exceeds 1.0, the AMBIENT= value is reduced so that the sum of the two values equals 1.0.

Syntax

DIRECT=*light-level*

DRILLDOWNFUNCTION Parameter

specifies the name of the JavaScript function that is called in Script drill-down mode.

Alias: DRILLFUNC

Syntax

DRILLDOWNFUNCTION=*function-name*
DRILLFUNC=*function-name*

DRILLDOWNMODE Parameter

specifies the drill-down mode.

Default: The ActiveX Control does not enable user-selectable drill-down modes.

Syntax

DRILLDOWNMODE=HTML | SCRIPT | URL

Argument Values

HTML
uses a substitution string to dynamically generate a URL based on the selected chart elements, and then passes the URL to the browser.

SCRIPT
invokes the JavaScript function specified in the DRILLDOWNFUNCTION= parameter, and passes into the function data from the selected graph element.

URL
provides static drill-down functionality, using an image map in the HTML file. The image map is generated by ODS when you use the ODS HTML destination with the URL= or HTML= option in SAS/GRAPH.

DRILLPATTERN Parameter

specifies how to construct the drill-down URL when the drill-down mode is HTML.

Tip: The substitution string is constructed with drill-down tags, which are expressed in parameters such as G_DEPV, as described in "Configuring Drill-Down Links with ActiveX" on page 207.You can also use variable substitution as described in "Using Variables as Substitution Strings" on page 211.

Syntax

DRILLPATTERN=*substitution-string*

DRILLTARGET Parameter

specifies where the drill-down destination is displayed in the browser.

Default: _BLANK, which displays the drill-down destination in a new browser window.

Tip: The target can be specified as another reserved target name or as the name of a window or frame in your web presentation.

Syntax

DRILLTARGET=*target*

G_COLOR Parameter

specifies a new color variable for the current drill-down level.

Syntax

G_COLOR=*variable-name*

G_COLORV Parameter

specifies that the current color variable is the same variable that was used to configure the previous drill-down level.

Syntax

G_COLORV=*variable-name*

G_DEP Parameter

specifies a new dependent variable for the current drill-down level.

Note: The value of the G_DEP tag cannot be set to *None* because it is always represented in the graph.

Syntax

G_DEP=*variable-name*

G_DEPTH Parameter

specifies a new depth variable for the current drill-down level.

Note: Drill-down graphs that use this variable can be vertical bar charts or scatter plots.

Syntax

G_DEPTH=*variable-name*

G_DEPTHV Parameter

specifies that the depth variable for the current drill-down level is the same depth variable that was used in the previous drill-down level.

Note: Drill-down graphs that use this variable can be vertical bar charts or scatter plots.

Syntax

G_DEPTHV=*variable-name*

G_DEPV Parameter

specifies that the drill-down graph at the specified drill-down level is to use the same dependent variable that was used in the previous drill-down level.

Syntax

G_DEPV=*variable-name*

G_GROUP Parameter

specifies a new group variable for the current drill-down level.

Note: Drill-down graphs that use this variable can be bar charts.

Syntax

G_GROUP=*variable-name*

G_GROUPV Parameter

specifies that this group variable should be the same group variable that was used at the previous drill-down level.

Note: Drill-down graphs that use this variable can be bar charts.

Syntax

G_GROUPV=*variable-name*

G_INDEP Parameter

specifies a new independent variable for the current drill-down level.

Restriction: The values of the G_INDEP tags cannot be set to *None* because it is always represented in the graph.

Note: Drill-down graphs that use this variable can be charts and maps.

Syntax

G_INDEP=*variable-name*

G_INDEPV Parameter

specifies that an independent variable at the current drill-down level is the same variable that was used at the previous drill-down level.

Note: Drill-down graphs that use this variable can be charts and maps.

Syntax

G_INDEPV=*variable-name*

G_SUBGR Parameter

specifies a new subgroup variable for the current drill-down level.

Note: Drill-down graphs that use this variable can be bar charts and scatter plots.

Syntax

G_SUBGR=*variable-name*

G_SUBGRV Parameter

specifies that a subgroup variable at this drill-down level is the same subgroup variable that was used at the previous drill-down level.

Note: Drill-down graphs that use this variable can be bar charts and scatter plots.

Syntax

G_SUBGRV=*variable-name*

PATTERNSTRIP Parameter

removes preceding and trailing white space from drill-down substitution patterns before the substituted text is added into a dynamically generated drill-down URL.

Default: FALSE

Syntax

PATTERNSTRIP=TRUE | FALSE

TIPS Parameter

specifies whether to display data tips.

Syntax

TIPS=TRUE | FALSE

Argument Values

FALSE
specifies that data tips are to be disabled.

TRUE
specifies that data tips are to be enabled.

TIPMODE Parameter

specifies which of two types of data tips are to be displayed.

Default: STANDARD

Syntax

TIPMODE=STANDARD | HTML | TABULAR | ALL

Argument Values

HTML
displays only the data tips that are indicated by the HTML= statement option.

TABULAR
display only the data tips that are indicated by the value of the TIPS parameter.

STANDARD | ALL
displays both sets of data tips.

Details

One set of data tips is specified with the TIPS parameter. The other set of data tips is specified with the HTML= statement option. The syntax of that option is HTML="TITLE='*text*' | *variable-name*". For more information about data tips, see "Data Tips for Web Presentations" on page 253.

VIEW2D Parameter

indicates whether the view point should be locked to two dimensions.

Default: FALSE

Interaction: This parameter might be overridden if you specify an ODS style template.

Syntax

VIEW2D=TRUE | FALSE

Appendix 3

Using SAS/GRAPH Fonts

Introduction

SAS/GRAPH fonts are the entries in the SASHELP.FONTS catalog. Information about these fonts is provided for special purposes only. For example, some specialized devices do not support system fonts. Or, you might want to use special symbols in the Marker font to display solid symbols for data points in a plot. If you specify the NOGSTYLE system option and one of the Z device drivers (see "Compatibility Device Drivers (Z Drivers)" on page 106), SAS/GRAPH uses SAS/GRAPH fonts.

In general, it is recommended that you use the system fonts supplied by SAS whenever possible. See "SAS/GRAPH, System, and Device-Resident Fonts" on page 350 and "TrueType Fonts Supplied by SAS" on page 350 for more information.

Note: The Java and ActiveX devices do not support SAS/GRAPH fonts. Also, SAS/GRAPH fonts cannot be used with template-based graphics.

Rendering Bitstream Fonts

SAS/GRAPH includes methods of storing rendered versions of Bitstream fonts, along with three graphics options to control how the fonts are rendered.

When your graphics output uses one of the Bitstream fonts that are provided with SAS/GRAPH, SAS/GRAPH must process information contained in corresponding FONTS catalog entries to determine how to draw characters of the specified size and typeface. The process of calculating the character shapes and sizes is known as *rendering* the font. Bitstream fonts that are available with SAS/GRAPH include the Century, Swiss, and Zapf font families.

SAS/GRAPH can store rendered versions of the Bitstream fonts in memory or in special SAS files. Using these rendered versions of the fonts can provide a speed improvement when characters of the same size and style are used again during the SAS session. SAS/GRAPH can read the rendered version of the characters from memory or from the rendered font file instead of performing the rendering calculations each time the characters are used. If you store the rendered fonts in files in a permanent SAS data set, SAS/GRAPH can use the rendered font files again in subsequent SAS sessions.

Note: Because the rendered font files use a special utility member type, they do not appear in the list of library members that is displayed in the DIRECTORY window.

You control whether and how rendered versions of fonts are stored using the FONTRES=, RENDER=, and RENDERLIB= graphics options. See “Graphics Options and Device Parameters Dictionary” on page 577 for details.

Listing or Displaying SAS/GRAPH Fonts on Your System

The SASHELP.FONTS catalog contains information about the fonts available on your system. To list the SAS/GRAPH fonts that can be used in your application, submit the following SAS code:

```
proc catalog catalog=sashelp.fonts entrytype=font;
   contents out=work.swfonts(keep=name);
run;
quit;
data work.swfonts;
   set work.swfonts;
   if name =:'HW' then delete;
run;
proc print data=work.swfonts;
run;
```

You can display these fonts with the GFONT procedure. See “Example 1: Displaying Fonts with Character Codes” on page 1214.

SAS/GRAPH Font Lists

The SAS/GRAPH fonts available with SAS/GRAPH are listed in the following tables.

All of the SAS/GRAPH fonts are stored in the catalog SASHELP.FONTS. For many fonts, the last letter or letters of the font name indicates weight or spacing of the font:

B
: bold (thicker)

E
: empty (outline) versions of their counterparts

I
: italic (slanted)

L
: light (thin)

U
: uniformly spaced versions of their counterparts; most of the SAS/GRAPH fonts that do not end in U are proportionately spaced. However, the kanji fonts are always uniform.

X
: expanded (wider characters and extra space between characters).

CAUTION
Empty and uniform versions of fonts cannot be used if you have deleted their filled or proportionally spaced counterparts.

If the label of a font in SASHELP.FONTS is “Depends on,” it is possible to delete it. However, empty and uniform versions of fonts are generated from their regular, bold, or italic counterparts. Therefore, if you delete any of these fonts, you cannot use the uniform or empty version of that font. For example, you must have the CENTB (Century Bold) font in order to use the CENTBE (Century Bold Empty) font.

Table A26.1 *Roman Alphabet Text Fonts*

Typeface	Font Name	Type Sample	Uniform Font
Brush	BRUSH	A B C a b c 1 2 3	
Century			
■ Bold	CENTB	A B C a b c 1 2 3	CENTBU
■ Bold Empty	CENTBE	A B C a b c 1 2 3	

Typeface	Font Name	Type Sample	Uniform Font
■ Bold Italic	CENTBI	A B C a b c 1 2 3	CENTBIU
■ Bold Italic Empty	CENTBIE	A B C a b c 1 2 3	
■ Expanded	CENTX	A B C a b c 1 2 3	CENTXU
■ Expanded Empty	CENTXE	A B C a b c 1 2 3	
■ Expanded Italic	CENTXI	A B C a b c 1 2 3	CENTXIU
■ Expanded Italic Empty	CENTXIE	A B C a b c 1 2 3	
German	GERMAN	𝔄 𝔅 ℭ a b c 1 2 3	GERMANU
German Italic	GITALIC	𝔄 𝔅 ℭ a b c 1 2 3	GITALICU
Hershey			
■ Sans Serif	SIMPLEX	A B C a b c 1 2 3	SIMPLEXU
■ Sans Serif Bold	DUPLEX	A B C a b c 1 2 3	DUPLEXU
■ Serif	COMPLEX	A B C a b c 1 2 3	COMPLEXU
■ Serif Bold	TRIPLEX	A B C a b c 1 2 3	TRIPLEXU
■ Serif Bold Italic	TITALIC	A B C a b c 1 2 3	TITALICU
■ Serif Italic	ITALIC	A B C a b c 1 2 3	ITALICU
Old English	OLDENG	𝔄 𝔅 ℭ a b c 1 2 3	OLDENGU
Script	SCRIPT	𝒜 ℬ 𝒞 a b c 1 2 3	
Cscript	CSCRIPT	𝒜 ℬ 𝒞 a b c 1 2 3	
Swiss	SWISS	A B C a b c 1 2 3	SWISSU
■ Empty	SWISSE	A B C a b c 1 2 3	
■ Bold	SWISSB	A B C a b c 1 2 3	SWISSBU
■ Bold Empty	SWISSBE	A B C a b c 1 2 3	
■ Bold Italic	SWISSBI	A B C a b c 1 2 3	SWISSBIU

Typeface	Font Name	Type Sample	Uniform Font
■ Bold Italic Empty	SWISSBIE	A B C a b c 1 2 3	
■ Expanded	SWISSX	A B C a b c 1 2 3	SWISSXU
■ Expanded Empty	SWISSXE	A B C a b c 1 2 3	
■ Expanded Bold	SWISSXB	A B C a b c 1 2 3	SWISSXBU
■ Expanded Bold Empty	SWISSXBE	A B C a b c 1 2 3	
■ Italic	SWISSI	A B C a b c 1 2 3	SWISSIU
■ Italic Empty	SWISSIE	A B C a b c 1 2 3	
■ Light	SWISSL	A B C a b c 1 2 3	SWISSLU
■ Light Empty	SWISSLE	A B C a b c 1 2 3	
Zapf	ZAPF	A B C a b c 1 2 3	ZAPFU
■ Empty	ZAPFE	A B C a b c 1 2 3	
■ Bold	ZAPFB	A B C a b c 1 2 3	ZAPFBU
■ Bold Empty	ZAPFBE	A B C a b c 1 2 3	
■ Bold Italic	ZAPFBI	A B C a b c 1 2 3	ZAPFBIU
■ Bold Italic Empty	ZAPFBIE	A B C a b c 1 2 3	
■ Italic	ZAPFI	A B C a b c 1 2 3	ZAPFIU
■ Italic Empty	ZAPFIE	A B C a b c 1 2 3	

Table A26.2 *Non-Roman Alphabet Fonts*

Typeface	Font Name	Uniform Font Name
Arabic	ARABIC	
Arabic Empty	ARABICE	
Cyrillic	CYRILLIC	CYRILLIU

Typeface	Font Name	Uniform Font Name
David	DAVID	
Davidb	DAVIDB	
Fsong	FSONG	FSONGU
Greek	GREEK	GREEKU
Greek (serif)	CGREEK	CGREEKU
Hebrew	HEBREW	
Hebrew	NHEBREW*	
Hebrewb	HEBREWB	
Hebrew Empty	HEBREWE	
Hei	HEI	HEIU
Hiragana	HIRA	
Hiragana	NHIRA*	
Kanji	KANJI	
Kanji	KANSJIS	
Kanji Subset		
Kanji 1	KAN1	
Kanji 2	KAN2	
Kanji 3	KAN3	
Kanji 4	KAN4	
Kanji 5	KAN5	
Kanji 6	KAN6	
Kanji 7	KAN7	
Kanji 8	KAN8	

Typeface	Font Name	Uniform Font Name
Katakana	KATA	
Katakana	NKATA*	
Korean	KGOTHB1	
Mincho	MINCHO	MINCHOE
*This font requires a special keyboard and is host-dependent. If you are not equipped to use this font, use the host-independent version listed directly above.		

Figure A26.1 *Greek (GREEK)*

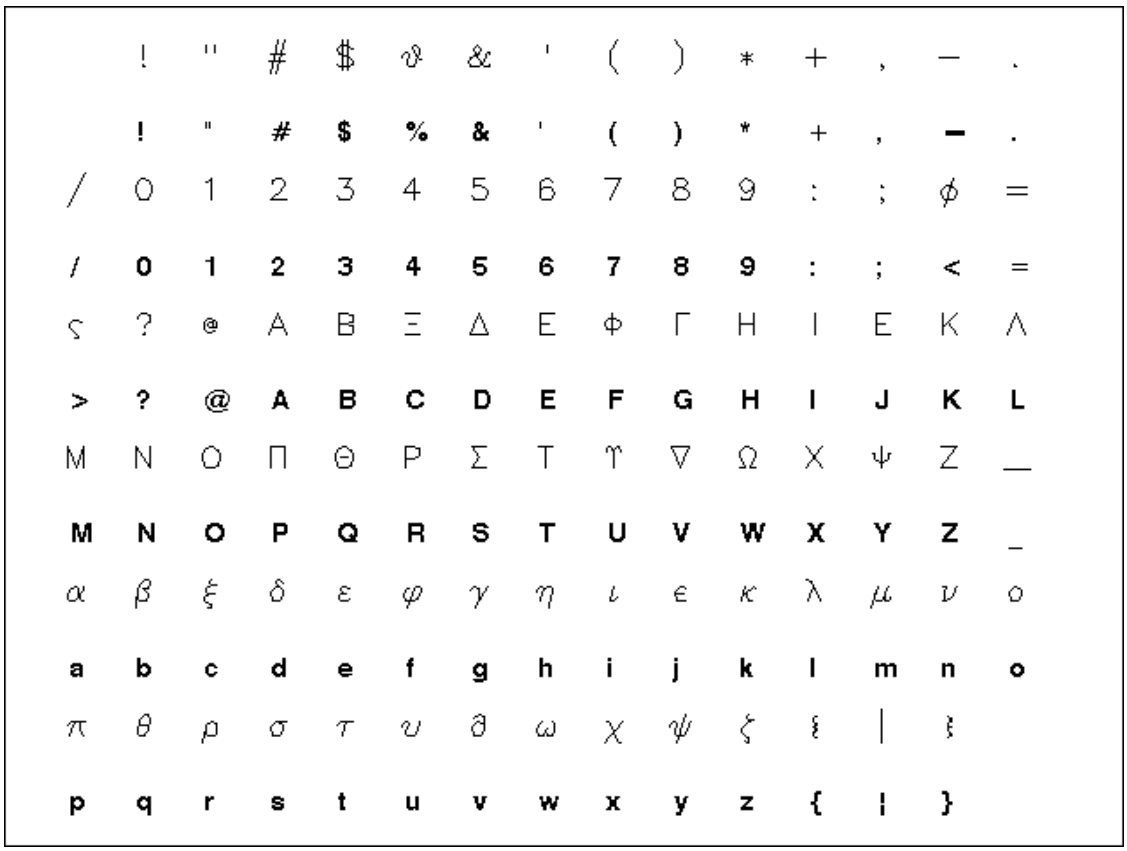


Figure A26.2 Greek Serif (CGREEK) Font

	!	″	#	$	ϑ	&	′	(	)	*	+	,	—	.
	!	"	#	$	%	&	'	(	)	*	+	,	−	.
/	0	1	2	3	4	5	6	7	8	9	:	;	ϕ	=
/	0	1	2	3	4	5	6	7	8	9	:	;	<	=
ς	?	@	A	B	Ξ	Δ	E	Φ	Γ	H	I	E	K	Λ
>	?	@	A	B	C	D	E	F	G	H	I	J	K	L
M	N	O	Π	Θ	P	Σ	T	ϒ	∇	Ω	X	Ψ	Z	___
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
α	β	ξ	δ	ε	φ	γ	η	ι	ϵ	κ	λ	μ	ν	ο
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
π	θ	ρ	σ	τ	υ	∂	ω	χ	ψ	ζ	{	|	}	
p	q	r	s	t	u	v	w	x	y	z	{	|	}	

Table A26.3 Symbol Fonts

Typeface	Font Name	Uniform Font Name
Cartographic	CARTOG	CARTOGU
Electronic	ELECTRON	ELECTROU
Marker	MARKER	
Marker Empty	MARKER E *	
Math	MATH	MATHU
Music	MUSIC	MUSICU
Special	SPECIAL	SPECIALU
Weather	WEATHER	WEATHERU

Typeface	Font Name	Uniform Font Name
*MARKERE is not displayed in the figures.		

Figure A26.3 *Cartographic Font*

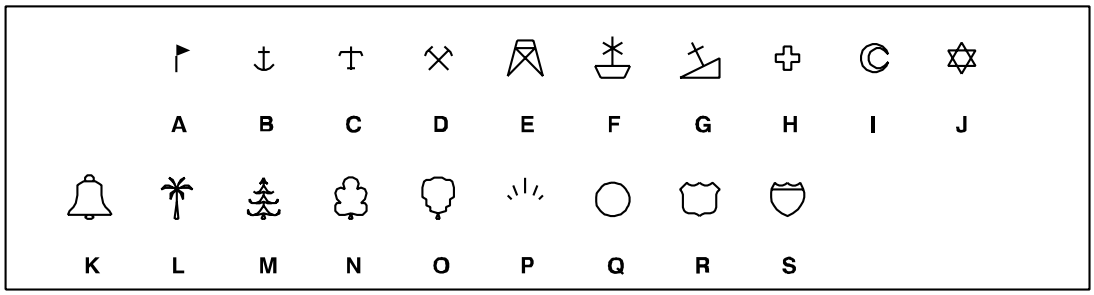
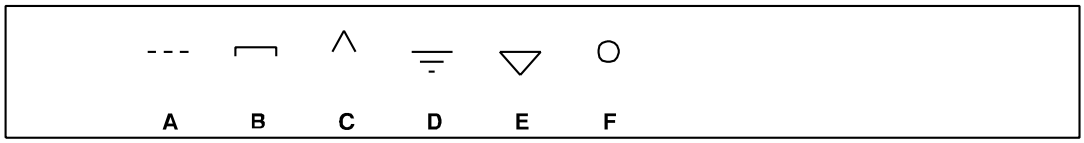


Figure A26.4 *Electronic Font*



Note: Figure A3.5 on page 1665 shows the MARKER font. The MARKERE font produces the same symbols but in empty (outline) form.

Figure A26.5 *Marker Font*

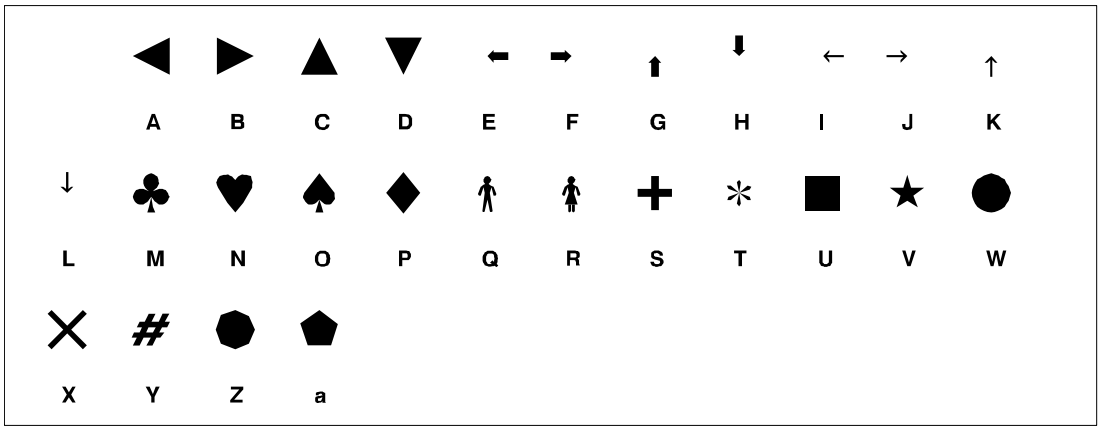


Figure A26.6 *Math Font*

	∥	⊥	∠	∴	<	>	±	∓	÷	≠	≡
	A	B	C	D	E	F	G	H	I	J	K
≦	≧	∝	∼	√	⊂	∪	⊃	∩	∈	→	↑
L	M	N	O	a	b	c	d	e	f	g	h
←	↓	∂	∇	∫	∮	∞	∃	∏	∑		
i	j	k	l	m	n	o	p	q	r		

Figure A26.7 *Music Font*

	𝅘𝅥	𝅘𝅥𝅮	𝅘𝅥𝅯	𝅝	𝅗𝅥	𝅘𝅥	♯	♮	♭	—
	A	B	C	D	E	F	G	H	I	J
𝄽	𝄾	𝄿	𝄞	𝄡	𝄪	𝄢	𝄫	𝄫		
K	L	M	N	O	P	Q	R	S		

Figure A26.8 Special Font

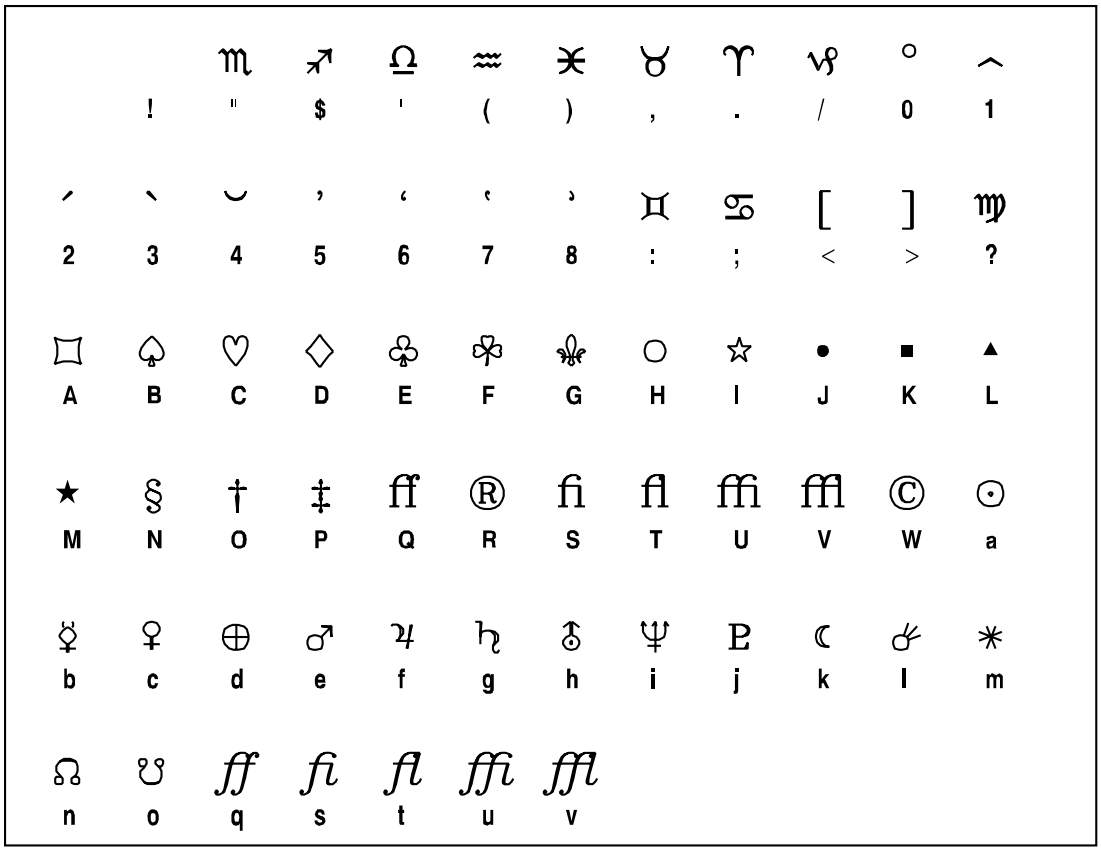
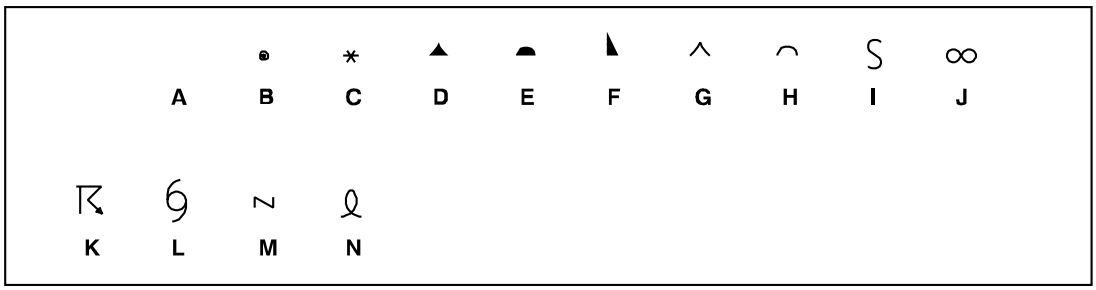


Figure A26.9 Weather Font



The SIMULATE Font

In some cases, the device's device-resident font cannot be used and the SIMULATE font is used instead. The SIMULATE font is a SAS/GRAPH font that simulates the device's resident characters by allowing the same amount of space for the text that the device-resident characters use. The SIMULATE font is used whenever the default device-resident font is unavailable, including the following situations:

- FONT=NONE or FONT=HW*xxxnnn* or no font is specified, **and** one of the following conditions or sets of conditions is also met:
  - GOPTIONS NOCHARACTERS is specified.
  - The device driver does not support device-resident text.
  - You request a device-resident font for a different device.
  - You specify an angle or rotation for the characters that the device does not support.
  - The device does not have a scalable font (characters can be generated only in the proportions specified with the font), **and** one of the following conditions is also met:
    - The values of the HPOS= and VPOS= graphics options do not match the values displayed in the LCOLS or PCOLS field or the LROWS or PROWS field in the Detail window of the device entry.
    - The HSIZE= or VSIZE= graphics option is set to values that are not the default.
    - You replay a graph in a template that is not the same size as the full size of the graphics output area, or you use a device driver other than the one you used to create the graph.
    - The target device and the display device have different values for the HPOS= and VPOS= graphics options.
    - You use any height specification, including the HEIGHT=, HTEXT=, HTITLE=, and HBY= graphics options, that is not equal to 1.

You should never delete the SIMULATE font from the fonts catalog.

Note: You can change the font that is used as the SIMULATE font with the SIMFONT= graphics option. If you use the SIMFONT= option, it is better to specify a uniform font. Do not specify a device-resident font as a substitute for SIMULATE. See "SIMFONT Graphics Option" on page 695 for more information about the SIMFONT= option.

Font Locations and the Default Search Path

SAS/GRAPH fonts are stored in catalogs. SAS/GRAPH looks only into catalogs with certain librefs and names to find fonts. By default, SAS/GRAPH searches for the font in the catalog SASHELP.FONTS, which contains SAS/GRAPH fonts, key maps, and device maps.

If you want to specify fonts that you have created locally, submit a LIBNAME statement that associates the libref GFONT0 with the location of your font catalog. If you have specified more than one libref in the sequence GFONT0 through GFONT9, SAS/GRAPH performs a sequential search of these catalogs when locating the font that you have specified.

When you specify a font name, SAS/GRAPH searches for the font in the following order:

1. If a SAS library with the libref GFONT0 exists, then SAS/GRAPH looks there for a catalog named FONTS. If GFONT0.FONTS exists, it is checked for the specified font. If the font is not there, then SAS/GRAPH looks next for a library with the libref GFONT1 and for a catalog named FONTS in that library. The search is repeated for the sequence of librefs through GFONT9.

2. SAS/GRAPH searches for the font in SASHELP.FONTS if the following situations occur.

   a. It fails to find the specified font in any FONTS catalog in the libraries GFONT0 to GFONT9.

   b. It finds a GFONT*n* libref without a FONTS catalog.

   c. It encounters an undefined libref in that sequence before locating the specified font.

   (SASHELP is one of the standard librefs defined automatically whenever you start your SAS session; you do not need to issue a LIBNAME statement to define it.)

3. If the specified font is not found in SASHELP.FONTS, then a warning is issued and the SIMULATE font is used. The SIMULATE font is the default SAS/GRAPH font and should never be deleted from the fonts catalog. See "The SIMULATE Font" on page 1667 for more information.

See Chapter 39, "GFONT Procedure," on page 1187 for additional information about specifying the libref GFONT0.

Appendix 4

Using Device-Resident Fonts

Introduction

You can use device-resident fonts with SAS/GRAPH output in four ways.

- by using the CHARTYPE= graphics option in a GOPTIONS statement to specify the default device-resident font. Assign the number of a font listed in the Chartype window of your device entry as the default device-resident font. See "Using a GOPTIONS Statement to Change the Default Device-Resident Font" on page 1673 for details.
- by using the GDEVICE procedure to specify the number of the font that you want to use as the default device-resident font. See "Using the GDEVICE Procedure to Change the Default Device-Resident Font" on page 1673 for details.
- by specifying the full font name as it appears on the Chartype window of the device driver entry. See "Specifying the Full Font Name" on page 1674 for details.
- by explicitly specifying a device-resident font name of the type HW*xxxnnn*. See "Specifying Alternative Device-Resident Fonts" on page 1674 for details.

There are several advantages to using device-resident fonts instead of SAS/GRAPH fonts. Device-resident fonts often are produced faster than SAS/GRAPH fonts and produce smaller output files. Also, some devices, such as laser printers with device-

resident fonts, might produce better quality output with device-resident fonts than with SAS/GRAPH fonts.

Default Device-Resident Fonts

When Are Device-Resident Fonts Used?

SAS/GRAPH uses a device's default device-resident font to draw characters when both of the following conditions are true:

- No font specification is made in the SAS/GRAPH program, or FONT=NONE is specified.
- The device-resident font can be used. See “Default Fonts” on page 351 for details about when device-resident fonts cannot be used.

The Chartype Value and Default Device-Resident Fonts

Every available device-resident font for a particular device has a number associated with it. This number and the corresponding font name are listed in the Chartype window of the device entry for your device. The default device-resident font is the font whose number is entered in the Chartype field in the Parameters window of the device entry. When FONT=NONE or no font is specified, SAS/GRAPH uses the font assigned to this field.

If your device has more than one device-resident font, you can assign a different default device-resident font in two ways:

- by specifying the font with the CHARTYPE= option in a GOPTIONS statement. See “Using a GOPTIONS Statement to Change the Default Device-Resident Font” on page 1673 .
- by using the GDEVICE procedure to modify the value of the Chartype field in the Parameters window of your device entry. See “Using the GDEVICE Procedure to Change the Default Device-Resident Font” on page 1673 for more details.

If your device has only one device-resident font (this is often the case), the Chartype field has a value of 0.

Using a GOPTIONS Statement to Change the Default Device-Resident Font

To assign the default device-resident font for your current SAS session, use the CHARTYPE= option in a GOPTIONS statement. Assign it the actual number of the device-resident font as listed in the Chartype field in the Chartype window of the device entry for your device.

Using the CHARTYPE= option changes the default font only for the duration of your SAS session; using the CHARTYPE= option does not change the value of the field in the device entry. (See "CHARTYPE Device Parameter and Graphics Option" on page 595 for a complete description of the CHARTYPE= option.)

When you specify a device-resident font by using the graphics option CHARTYPE=*n* and the font specification NONE, the size of the character cells is determined by the current values for the HPOS= and VPOS= options. This means that the font is drawn using the current cell size. As a result, the aspect ratio of the displayed font might be different and the height of the characters, if displayed in cells, might be affected.

CAUTION
Specifying a nonscalable device-resident font with the CHARTYPE= option might cause the SIMULATE font to be used.

In addition, the SIMULATE font is substituted if both of the following conditions are true.

- The font selected with CHARTYPE= is not scalable.
- The values of the HPOS= and VPOS= options do not match the values of the Rows and Cols fields in the Chartype window.

Using the GDEVICE Procedure to Change the Default Device-Resident Font

To change the default device-resident font with the GDEVICE procedure, change the Chartype field in the Parameters window for the device:

1. Invoke the GDEVICE procedure and select the entry for your device.
2. Go to the Chartype window and review the available fonts.
3. Note the number of the font that you want to use as the default font and go to the Parameters window.
4. Enter the number of the font in the Chartype field.
5. Close the window and exit the procedure.

Note: If you change the number in the Chartype field in the Parameters window of the device entry, the change remains in effect until you change the entry again.

(See Chapter 38, “GDEVICE Procedure,” on page 1163 for information about viewing device entries and changing device parameters.)

Specifying the Full Font Name

You can specify the full font name in any SAS statement where a font specification is valid. For example, you can specify the full font name in the FTEXT=*font* graphics option or the FONT=*font* specification in a TITLE statement. For the value font, specify the full font name exactly as it appears in the Chartype window of the device entry. For example, to specify the Times-Roman font in a TITLE statement when you use the PS300 device, use this code:

```
title font="Times-Roman" "Testing the Times-Roman font";
```

SAS allows up to 255 characters for the font name. The font name might contain spaces. If the font name is longer than 40 characters, PROC GDEVICE in fullscreen mode displays only the first 37 characters, followed by an ellipsis (...). To see the complete font name when the name is longer than 40 characters, use PROC GDEVICE with the NOFS (no fullscreen) option as follows:

```
proc gdevice c=sashelp.devices nofs;
   list driver-name;
run;
quit;
```

When a font is quoted, SAS first looks at the Chartype window of the device driver entry to determine whether it is a valid device-resident font. If the font is not found in the Chartype window, SAS then checks to determine whether the quoted font is a valid SAS/GRAPH font. If the font is not recognized as either a valid device-resident font or a valid SAS/GRAPH font, the SIMULATE font is used.

Specifying Alternative Device-Resident Fonts

An alternative device-resident font can be specified in any SAS statement where a font specification is valid. You can use more than one device-resident font in a single graph or even in a single statement. All of the fonts that you specify must exist on your device. If you specify a device-resident font, make sure that the font is available on the device and that there is a corresponding Chartype value for the font. If you request a device-resident font that does not have a Chartype defined, SAS/GRAPH substitutes the SIMULATE font.

These are the three ways to specify alternative device-resident fonts:

- In the font specification, explicitly assign a device-resident font using the following form:

  HW*xxxnnn*

  HW
  identifies the font as a device-resident font. The font name must begin with the characters HW.

  xxx
  are the last two or three characters of the module name in the Module field in the Detail window of your device entry. If the module name has eight characters (SASGDPSL, for example), use the last three characters (PSL). If the module name has only seven characters (SASGDVT, for example), use the last two characters (VT).

  nnn
  is the Chartype number of the device-resident font that you want to use as listed in the Chartype window in the device entry. This value should be a three-digit decimal number, with leading zeros if necessary.

- In the font specification, explicitly assign a device-resident font using the following form:

  device-resident-font-name
  identifies the name of the device-resident font that is listed in the Chartype window of the device entry. *Device-resident-font-name* must be enclosed in quotation marks and the maximum length is 256 characters. The specified font name is converted internally to the HW*xxxnnn* name. Note that in Annotate, the specified font name must be enclosed in both double quotation marks and single quotation marks. See "The Annotate Function, Variable, and Macro Dictionaries" on page 724 for details.

- Assign one of the fonts listed in the Chartype window of your device entry as the default device-resident font with the CHARTYPE= graphics option. You can also change the default device-resident font by modifying the value of the Chartype field in the Parameters window of your device entry. Then you can use FONT=NONE in your SAS/GRAPH procedure or statement to specify the new default device-resident font.

When you specify FONT=HW*xxxnnn* or *device-resident-font-name*, the size of the character cells is determined by the values in the Rows and Cols fields in the Chartype window of the device entry. The values of the HPOS= and VPOS= options are ignored for the font. Consequently, the font retains its original proportions. In addition, with this method the font catalog is checked for proportional spacing information. This information is used by SAS/GRAPH to determine how much space to reserve for proportional text. See "Graphics Options and Device Parameters Dictionary" on page 577 for additional information.

Appendix 5

Special Fonts and Symbols in JAVAIMG Output

Special Fonts and Symbols in JAVAIMG Output

The JAVAIMG device supports only system fonts. The following table shows the special symbols that the JAVAIMG device supports and the value that you must specify in the SYMBOL statement VALUE= option for each.

Table A28.1 *Symbols That Are Supported by Java*

Value	Symbol	Value	Symbol
* (asterisk)	♀	PLUS	+
+	⊕	SQUARE	□
>	♂	STAR	✳
CIRCLE	○	TRIANGLE	△
DIAMOND	◇	X	×

Value	Symbol	Value	Symbol
DOT	●		

See “SYMBOL Statement” on page 470 for more information.

Appendix 6

SAS Formats Supported in JAVAIMG Output

SAS Formats Supported in JAVAIMG Output

The JAVAIMG device supports the SAS character, numeric, and the date and time formats that are listed in the following tables. For a description of these formats, see *SAS System Options: Reference*.

Table A29.1 *Character Formats Supported by Java*

$	$ASCII	$BINARY	$CHAR
$F	$HEX	$OCTAL	

Table A29.2 *Numeric Formats Supported by Java*

BEST	BINARY	COMMA	COMMAX	COMMAX
D	DOLLAR	DOLLARX	E	EURO
EUROX	F	HEX	LOGPROB	NEGPAREN
NLBEST	NLD	NLMNIAED	NLMNIAUD	NLMNIBGN
NLMNIBRL	NLMNICAD	NLMNICHF	NLMNICNY	NLMNICZK

NLMNIDKK	NLMNIEEK	NLMNIEGP	NLMNIEUR	NLMNIGBP
NLMNIHKD	NLMNIHRK	NLMNIHUF	NLMNIIDR	NLMNIILS
NLMNIINR	NLMNIJPY	NLMNIKRW	NLMNILTL	NLMNILVL
NLMNIMOP	NLMNIMXN	NLMNIMYR	NLMNINOK	NLMNINZD
NLMNIPLN	NLMNIROL	NLMNIRUB	NLMNIRUR	NLMNISEK
NLMNISGD	NLMNISKK	NLMNITHB	NLMNITRY	NLMNITWD
NLMNIUSD	NLMNIZAR	NLMNLAED	NLMNLAUD	NLMNLBGN
NLMNLBRL	NLMNLCAD	NLMNLCHF	NLMNLCNY	NLMNLCZK
NLMNLDKK	NLMNLEEK	NLMNLEGP	NLMNLEUR	NLMNLGBP
NLMNLHKD	NLMNLHRK	NLMNLHUF	NLMNLIDR	NLMNLILS
NLMNLINR	NLMNLJPY	NLMNLKRW	NLMNLLTL	NLMNLLVL
NLMNLMOP	NLMNLMXN	NLMNLMYR	NLMNLNOK	NLMNLNZD
NLMNLPLN	NLMNLROL	NLMNLRUB	NLMNLRUR	NLMNLSEK
NLMNLSGD	NLMNLSKK	NLMNLTHB	NLMNLTRY	NLMNLTWD
NLMNLUSD	NLMNLZAR	NLMNY	NLMNYI	NLNUM
NLNUMI	NLPCT	NLPCTI	NLPVALUE	NUMX
OCTAL	PERCENT	PERCENTN	PVALUE	ROMAN
RSTDOCNY	RSTDOCYY	RSTDONYN	RSTDOPNY	RSTDOPYN
RSTDOPYY	YEN			

Table A29.3 *Date and Time Formats Supported by Java*

AFRDFDD	AFRDFDE	AFRDFDN	AFRDFDT	AFRDFDWN
AFRDFMN	AFRDFMY	AFRDFWDX	AFRDFWKX	CATDFDD
CATDFDE	CATDFDN	CATDFDT	CATDFDWN	CATDFMN
CATDFMY	CATDFWDX	CATDFWKX	CRODFDD	CRODFDE
CRODFDN	CRODFDT	CRODFDWN	CRODFMN	CRODFMY
CRODFWDX	CRODFWKX	CSYDFDD	CSYDFDE	CSYDFDN
CSYDFDT	CSYDFDWN	CSYDFMN	CSYDFMY	CSYDFWDX
CSYDFWKX	DANDFDD	DANDFDE	DANDFDN	DANDFDT

DANDFDWN	DANDFMN	DANDFMY	DANDFWDX	DANDFWKX
DATE	DATEAMPM	DATETIME	DAY	DDMMYY
DDMMYYN	DESDFDD	DESDFDE	DESDFDN	DESDFDT
DESDFDWN	DESDFMN	DESDFMY	DESDFWDX	DESDFWKX
DEUDFDD	DEUDFDE	DEUDFDN	DEUDFDT	DEUDFDWN
DEUDFMN	DEUDFMY	DEUDFWDX	DEUDFWKX	DOWNAME
DTDATE	DTMONYY	DTWKDATX	DTYEAR	DTYYQC
ENGDFDD	ENGDFDE	ENGDFDN	ENGDFDT	ENGDFDWN
ENGDFMN	ENGDFMY	ENGDFWDX	ENGDFWKX	ESPDFDD
ESPDFDE	ESPDFDN	ESPDFDT	ESPDFDWN	ESPDFMN
ESPDFMY	ESPDFWDX	ESPDFWKX	EURDFDD	EURDFDE
EURDFDN	EURDFDT	EURDFDWN	EURDFMN	EURDFMY
EURDFWDX	EURDFWKX	FINDFDD	FINDFDE	FINDFDN
FINDFDT	FINDFDWN	FINDFMN	FINDFMY	FINDFWDX
FINDFWKX	FRADFDD	FRADFDE	FRADFDN	FRADFDT
FRADFDWN	FRADFMN	FRADFMY	FRADFWDX	FRADFWKX
FRSDFDD	FRSDFDE	FRSDFDN	FRSDFDT	FRSDFDWN
FRSDFMN	FRSDFMY	FRSDFWDX	FRSDFWKX	HHMM
HOUR	HUNDFDD	HUNDFDE	HUNDFDN	HUNDFDT
HUNDFDWN	HUNDFMN	HUNDFMY	HUNDFWDX	HUNDFWKX
ITADFDD	ITADFDE	ITADFDN	ITADFDT	ITADFDWN
ITADFMN	ITADFMY	ITADFWDX	ITADFWKX	JDATEMD
JDATEMON	JDATEQRW	JDATEQTR	JDATESEM	JDATESMW
JULDATE	JULDAY	JULIAN	MACDFDD	MACDFDE
MACDFDN	MACDFDT	MACDFDWN	MACDFMN	MACDFMY
MACDFWDX	MACDFWKX	MMDDYY	MMDDYYN	MMSS
MMYY	MMYYN	MONNAME	MONTH	MONYY
NLDATE	NLDATEMD	NLDATEMN	NLDATEW	NLDATEWN
NLDATEYM	NLDATEYQ	NLDATEYR	NLDATEYW	NLDATM

NLDATMAP	NLDATMDT	NLDATMMD	NLDATMTM	NLDATMTZ
NLDATMW	NLDATMWN	NLDATMWZ	NLDATMYM	NLDATMYQ
NLDATMYR	NLDATMYW	NLDATMZ	NLDDFDD	NLDDFDE
NLDDFDN	NLDDFDT	NLDDFDWN	NLDDFMN	NLDDFMY
NLDDFWDX	NLDDFWKX	NLSTRMON	NLSTRQTR	NLSTRWK
NLTIMAP	NLTIME	NORDFDD	NORDFDE	NORDFDN
NORDFDT	NORDFDWN	NORDFMN	NORDFMY	NORDFWDX
NORDFWKX	POLDFDD	POLDFDE	POLDFDN	POLDFDT
POLDFDWN	POLDFMN	POLDFMY	POLDFWDX	POLDFWKX
PTGDFDD	PTGDFDE	PTGDFDN	PTGDFDT	PTGDFDWN
PTGDFMN	PTGDFMY	PTGDFWDX	PTGDFWKX	QTR
QTRR	RUSDFDD	RUSDFDE	RUSDFDN	RUSDFDT
RUSDFDWN	RUSDFMN	RUSDFMY	RUSDFWDX	RUSDFWKX
SLODFDD	SLODFDE	SLODFDN	SLODFDT	SLODFDWN
SLODFMN	SLODFMY	SLODFWDX	SLODFWKX	SVEDFDD
SVEDFDE	SVEDFDN	SVEDFDT	SVEDFDWN	SVEDFMN
SVEDFMY	SVEDFWDX	SVEDFWKX	TIME	TIMEAMPM
TOD	WEEKDATE	WEEKDATX	WEEKDAY	WEEKU
WEEKV	WEEKW	WORDDATE	WORDDATX	YEAR
YYMM	YYMMDD	YYMMDDN	YYMMN	YYMON
YYQ	YYQN	YYQR	YYQRN	YYWEEKU
YYWEEKV	YYWEEKW			

Note: The JAVAIMG device does not support nested formats that include user-defined formats. If you want to use a custom format with these devices and the custom format nests other formats, make sure that it nests only the formats that are provided by SAS.

Note: The JAVAIMG device does not honor the DATATYPE=DATE and NOEDIT options in the FORMAT procedure PICTURE statement.

Appendix 7

Predefined Colors

Predefined Colors

The following table shows the colors that are predefined in the SAS Registry, arranged by name. For each color, the equivalent CX color code (RGB), the equivalent HLS color code, and a color sample are shown. For a table of the SAS registry colors arranged by hue, see Table A7.2 on page 1690.

Table A30.1 *SAS Registry Colors by Name*

Color Name	Color Sample	CX Color Code (RGB)	HLS Color Code[1]
AliceBlue		CXF0F8FF	H148F8FF
AntiqueWhite		CXFAEBD7	H09AE9C6
Aqua		CX00FFFF	H12C80FF
Aquamarine		CX7FFDD4	H118BEF7
Azure		CXF0FFFF	H12CF8FF
Beige		CXF5F5DC	H0B4E98E
Bisque		CXFFE4C4	H098E2FF
Black		CX000000	H0000000

Color Name	Color Sample	CX Color Code (RGB)	HLS Color Code[1]
BlanchedAlmond		CXFFEBCD	H09CE6FF
Blue		CX0000FF	H00080FF
BlueViolet		CX8A2BE2	H01F87C2
Brown		CXA52A2A	H0786898
Burlywood		CXDEB887	H099B391
CadetBlue		CX5F9EA0	H12D8041
Chartreuse		CX7FFF00	H0D280FF
Chocolate		CXD2691E	H09178BF
Coral		CXFF7F50	H088A8FF
CornflowerBlue		CX6495ED	H152A9CA
Cornsilk		CXFFF8DC	H0A7EEFF
Crimson		CXDC143C	H06C78D5
Cyan		CX00FFFF	H12C80FF
DarkBlue		CX00008B	H00046FF
DarkCyan		CX008B8B	H12C46FF
DarkGoldenrod		CXB8860B	H0A262E2
DarkGray		CXA9A9A9	H000A900
DarkGreen		CX006400	H0F032FF
DarkKhaki		CXBDB76B	H0AF9462
DarkMagenta		CX8B008B	H03C46FF
DarkOliveGreen		CX556B2F	H0CA4D63
DarkOrange		CXFF8C00	H09880FF

Color Name	Color Sample	CX Color Code (RGB)	HLS Color Code[1]
DarkOrchid		CX9932CC	H0287F9B
DarkRed		CX8B0000	H07846FF
DarkSalmon		CXE9967A	H087B2B7
DarkSeaGreen		CX8FBC8F	H0F0A640
DarkSlateBlue		CX483D8B	H0086463
DarkSlateGray		CX2F4F4F	H12C3F41
DarkSlateGrey		CX2F2F2F	H0002F00
DarkTurquoise		CX00CED1	H12C69FF
DarkViolet		CX9400D3	H02A6AFF
DeepPink		CXFF1493	H0578AFF
DeepSkyBlue		CX00BFFF	H13B80FF
DimGray		CX696969	H0006900
DodgerBlue		CX1E90FF	H1498FFF
Firebrick		CXB22222	H0786AAD
FloralWhite		CXFFFAF0	H0A0F8FF
ForestGreen		CX228B22	H0F0579B
Fuchsia		CXFF00FF	H03C80FF
Gainsboro		CXDCDCDC	H000DC00
GhostWhite		CXF8F8FF	H000FCFF
Gold		CXFFD700	H0AA80FF
Goldenrod		CXDAA520	H0A27DBE
Gray or Grey		CX808080	H0008000

Color Name	Color Sample	CX Color Code (RGB)	HLS Color Code[1]
Green		CX008000	H0F040FF
GreenYellow		CXADFF2F	H0CB97FF
Honeydew		CXF0FFF0	H0F0F8FF
HotPink		CXFF69B4	H05AB4FF
IndianRed		CXCD5C5C	H0789587
Indigo		CX4B0082	H02241FF
Ivory		CXFFFFF0	H0B4F8FF
Khaki		CXF0E68C	H0AEBEC4
Lavender		CXE6E6FA	H000F0AA
LavenderBlush		CXFFF0F5	H064F8FF
LawnGreen		CX7CFC00	H0D27EFF
LemonChiffon		CXFFFACD	H0AEE6FF
LightBlue		CXADD8E6	H13ACA88
LightCoral		CXF08080	H078B8C9
LightCyan		CXE0FFFF	H12CF0FF
LightGoldenrodYellow		CXFAFAD2	H0B4E6CC
LightGray		CXD3D3D3	H000D300
LightGreen		CX90EE90	H0F0BFBB
LightPink		CXFFB6C1	H06EDBFF
LightSalmon		CXFFA07A	H089BDFF
LightSeaGreen		CX20B2AA	H12869B1
LightSkyBlue		CX87CEFA	H142C1EB

Color Name	Color Sample	CX Color Code (RGB)	HLS Color Code[1]
LightSlateGray		CX778899	H14A8824
LightSteelBlue		CXB0C4DE	H14DC769
LightYellow		CXFFFFE0	H0B4F0FF
Lime		CX00FF00	H0F080FF
LimeGreen		CX32CD32	H0F0809B
Linen		CXFAF0E6	H095F0AA
Magenta		CXFF00FF	H03C80FF
Maroon		CX800000	H07840FF
MediumAquamarine		CX66CDAA	H1179A81
MediumBlue		CX0000CD	H00067FF
MediumOrchid		CXBA55D3	H0309496
MediumPurple		CX9370DB	H013A698
MediumSeaGreen		CX3CB371	H10A787F
MediumSlateBlue		CX7B68EE	H008ABCB
MediumSpringGreen		CX00FA9A	H1147DFF
MediumTurquoise		CX48D1CC	H1298D99
MediumVioletRed		CXC71585	H0526ECE
MidnightBlue		CX191970	H00045A2
MintCream		CXF5FFFA	H10EFAFF
MistyRose		CXFFE4E1	H07EF0FF
Moccasin		CXFFE4B5	H09EDAFF
NavajoWhite		CXFFDEAD	H09BD6FF

Color Name	Color Sample	CX Color Code (RGB)	HLS Color Code[1]
Navy		CX000080	H00040FF
OldLace		CXFDF5E6	H09FF2D9
Olive		CX808000	H0B440FF
OliveDrab		CX6B8E23	H0C7599A
Orange		CXFFA500	H09E80FF
OrangeRed		CXFF4500	H08880FF
Orchid		CXDA70D6	H03EA596
PaleGoldenrod		CXEEE8AA	H0AECCAA
PaleGreen		CX98FB98	H0F0CAEC
PaleTurquoise		CXAFEEEE	H12CCFA6
PaleVioletRed		CXDB7093	H064A698
PapayaWhip		CXFFEFD5	H09DEAFF
Peachpuff		CXFFDAB9	H094DCFF
Peru		CXCD853F	H0958696
Pink		CXFFC0CB	H06DE0FF
Plum		CXDDA0DD	H03CBF79
PowderBlue		CXB0E0E6	H132CB84
Purple		CX800080	H03C40FF
Red		CXFF0000	H07880FF
RosyBrown		CXBC8F8F	H078A640
RoyalBlue		CX4169E1	H15991B9
SaddleBrown		CX8B4513	H0914FC2

Color Name	Color Sample	CX Color Code (RGB)	HLS Color Code[1]
Salmon		CXFA8072	H07EB6EE
SandyBrown		CXF4A460	H093AADE
SeaGreen		CX2E8B57	H10A5D80
Seashell		CXFFF5EE	H090F7FF
Sienna		CXA0522D	H08B678F
Silver		CXC0C0C0	H000C000
SkyBlue		CX87CEEB	H13DB9B6
SlateBlue		CX6A5ACD	H0089488
SlateGray		CX708090	H14A8020
Snow		CXFFFAFA	H078FDFF
SpringGreen		CX00FF7F	H10D80FF
SteelBlue		CX4682B4	H1477D70
Tan		CXD2B48C	H09AAF70
Teal		CX008080	H12C40FF
Thistle		CXD8BFD8	H03CCC3E
Tomato		CXFF6347	H081A3FF
Turquoise		CX40E0D0	H12690B8
Violet		CXEE82EE	H03CB8C2
Wheat		CXF5DEB3	H09FD4C4
White		CXFFFFFF	H000FF00
WhiteSmoke		CXF5F5F5	H000F500
Yellow		CXFFFF00	H0B480FF

Color Name	Color Sample	CX Color Code (RGB)	HLS Color Code[1]
YellowGreen		CX9ACD32	H0C7809B

1 The HLS color naming scheme in SAS follows the Tektronix Color Standard, which places blue at 0 degrees on the hue coordinate. For more information, see "HLS Color Codes" on page 372.

The following table shows the colors that are predefined in the SAS Registry, arranged by hue according to the Tektronix model. For each color, a color sample, and the equivalent RGB (CX) and HLS color codes are shown.

Table A30.2 *SAS Registry Colors by Hue*

Color Sample	Color Name	CX Code (RGB)	HLS Code[1]
	Black	CX000000	H0000000
	DarkSlateGrey	CX2F2F2F	H0002F00
	DimGray	CX696969	H0006900
	Gray or Grey	CX808080	H0008000
	DarkGray	CXA9A9A9	H000A900
	Silver	CXC0C0C0	H000C000
	LightGray	CXD3D3D3	H000D300
	Gainsboro	CXDCDCDC	H000DC00
	WhiteSmoke	CXF5F5F5	H000F500
	White	CXFFFFFF	H000FF00
	MidnightBlue	CX191970	H00045A2
	Lavender	CXE6E6FA	H000F0AA
	Navy	CX000080	H00040FF
	DarkBlue	CX00008B	H00046FF
	MediumBlue	CX0000CD	H00067FF
	Blue	CX0000FF	H00080FF

Color Sample	Color Name	CX Code (RGB)	HLS Code[1]
	GhostWhite	CXF8F8FF	H000FCFF
	SlateBlue	CX6A5ACD	H0089488
	DarkSlateBlue	CX483D8B	H0086463
	MediumSlateBlue	CX7B68EE	H008ABCB
	MediumPurple	CX9370DB	H013A698
	BlueViolet	CX8A2BE2	H01F87C2
	Indigo	CX4B0082	H02241FF
	DarkOrchid	CX9932CC	H0287F9B
	DarkViolet	CX9400D3	H02A6AFF
	MediumOrchid	CXBA55D3	H0309496
	Thistle	CXD8BFD8	H03CCC3E
	Plum	CXDDA0DD	H03CBF79
	Violet	CXEE82EE	H03CB8C2
	Purple	CX800080	H03C40FF
	DarkMagenta	CX8B008B	H03C46FF
	Fuchsia	CXFF00FF	H03C80FF
	Magenta	CXFF00FF	H03C80FF
	Orchid	CXDA70D6	H03EA596
	MediumVioletRed	CXC71585	H0526ECE
	DeepPink	CXFF1493	H0578AFF
	HotPink	CXFF69B4	H05AB4FF
	LavenderBlush	CXFFF0F5	H064F8FF

Color Sample	Color Name	CX Code (RGB)	HLS Code[1]
	PaleVioletRed	CXDB7093	H064A698
	Crimson	CXDC143C	H06C78D5
	Pink	CXFFC0CB	H06DE0FF
	LightPink	CXFFB6C1	H06EDBFF
	RosyBrown	CXBC8F8F	H078A640
	IndianRed	CXCD5C5C	H0789587
	Brown	CXA52A2A	H0786898
	Firebrick	CXB22222	H0786AAD
	LightCoral	CXF08080	H078B8C9
	Maroon	CX800000	H07840FF
	DarkRed	CX8B0000	H07846FF
	Red	CXFF0000	H07880FF
	Snow	CXFFFAFA	H078FDFF
	MistyRose	CXFFE4E1	H07EF0FF
	Salmon	CXFA8072	H07EB6EE
	Tomato	CXFF6347	H081A3FF
	DarkSalmon	CXE9967A	H087B2B7
	Coral	CXFF7F50	H088A8FF
	OrangeRed	CXFF4500	H08880FF
	LightSalmon	CXFFA07A	H089BDFF
	Sienna	CXA0522D	H08B678F
	Seashell	CXFFF5EE	H090F7FF

Color Sample	Color Name	CX Code (RGB)	HLS Code[1]
	Chocolate	CXD2691E	H09178BF
	SaddleBrown	CX8B4513	H0914FC2
	SandyBrown	CXF4A460	H093AADE
	Peachpuff	CXFFDAB9	H094DCFF
	Peru	CXCD853F	H0958696
	Linen	CXFAF0E6	H095F0AA
	Bisque	CXFFE4C4	H098E2FF
	DarkOrange	CXFF8C00	H09880FF
	Burlywood	CXDEB887	H099B391
	Tan	CXD2B48C	H09AAF70
	AntiqueWhite	CXFAEBD7	H09AE9C6
	NavajoWhite	CXFFDEAD	H09BD6FF
	BlanchedAlmond	CXFFEBCD	H09CE6FF
	PapayaWhip	CXFFEFD5	H09DEAFF
	Moccasin	CXFFE4B5	H09EDAFF
	Orange	CXFFA500	H09E80FF
	Wheat	CXF5DEB3	H09FD4C4
	OldLace	CXFDF5E6	H09FF2D9
	FloralWhite	CXFFFAF0	H0A0F8FF
	DarkGoldenrod	CXB8860B	H0A262E2
	Goldenrod	CXDAA520	H0A27DBE
	Cornsilk	CXFFF8DC	H0A7EEFF

Color Sample	Color Name	CX Code (RGB)	HLS Code[1]
	Gold	CXFFD700	H0AA80FF
	Khaki	CXF0E68C	H0AEBEC4
	LemonChiffon	CXFFFACD	H0AEE6FF
	PaleGoldenrod	CXEEE8AA	H0AECCAA
	DarkKhaki	CXBDB76B	H0AF9462
	Beige	CXF5F5DC	H0B4E98E
	LightGoldenrodYellow	CXFAFAD2	H0B4E6CC
	Olive	CX808000	H0B440FF
	Yellow	CXFFFF00	H0B480FF
	LightYellow	CXFFFFE0	H0B4F0FF
	Ivory	CXFFFFF0	H0B4F8FF
	OliveDrab	CX6B8E23	H0C7599A
	YellowGreen	CX9ACD32	H0C7809B
	DarkOliveGreen	CX556B2F	H0CA4D63
	GreenYellow	CXADFF2F	H0CB97FF
	Chartreuse	CX7FFF00	H0D280FF
	LawnGreen	CX7CFC00	H0D27EFF
	DarkSeaGreen	CX8FBC8F	H0F0A640
	ForestGreen	CX228B22	H0F0579B
	LimeGreen	CX32CD32	H0F0809B
	LightGreen	CX90EE90	H0F0BFBB
	PaleGreen	CX98FB98	H0F0CAEC

Color Sample	Color Name	CX Code (RGB)	HLS Code[1]
	DarkGreen	CX006400	H0F032FF
	Green	CX008000	H0F040FF
	Lime	CX00FF00	H0F080FF
	Honeydew	CXF0FFF0	H0F0F8FF
	SeaGreen	CX2E8B57	H10A5D80
	MediumSeaGreen	CX3CB371	H10A787F
	SpringGreen	CX00FF7F	H10D80FF
	MintCream	CXF5FFFA	H10EFAFF
	MediumSpringGreen	CX00FA9A	H1147DFF
	MediumAquamarine	CX66CDAA	H1179A81
	Aquamarine	CX7FFDD4	H118BEF7
	Turquoise	CX40E0D0	H12690B8
	LightSeaGreen	CX20B2AA	H12869B1
	MediumTurquoise	CX48D1CC	H1298D99
	DarkSlateGray	CX2F4F4F	H12C3F41
	PaleTurquoise	CXAFEEEE	H12CCFA6
	Teal	CX008080	H12C40FF
	DarkCyan	CX008B8B	H12C46FF
	Aqua	CX00FFFF	H12C80FF
	Cyan	CX00FFFF	H12C80FF
	LightCyan	CXE0FFFF	H12CF0FF
	Azure	CXF0FFFF	H12CF8FF

Color Sample	Color Name	CX Code (RGB)	HLS Code[1]
	DarkTurquoise	CX00CED1	H12C69FF
	CadetBlue	CX5F9EA0	H12D8041
	PowderBlue	CXB0E0E6	H132CB84
	LightBlue	CXADD8E6	H13ACA88
	DeepSkyBlue	CX00BFFF	H13B80FF
	SkyBlue	CX87CEEB	H13DB9B6
	LightSkyBlue	CX87CEFA	H142C1EB
	SteelBlue	CX4682B4	H1477D70
	AliceBlue	CXF0F8FF	H148F8FF
	DodgerBlue	CX1E90FF	H1498FFF
	SlateGray	CX708090	H14A8020
	LightSlateGray	CX778899	H14A8824
	LightSteelBlue	CXB0C4DE	H14DC769
	CornflowerBlue	CX6495ED	H152A9CA
	RoyalBlue	CX4169E1	H15991B9

1 The HLS color naming scheme in SAS follows the Tektronix Color Standard, which places blue at 0 degrees on the hue coordinate. For more information, see "HLS Color Codes" on page 372.

50

Color Utility Macro Reference

Dictionary

%COLORMAC Macro Statement

creates the color utility macros in the current SAS session.

Note: You must run the %COLORMAC macro in your current SAS session before you can use the other color utility macros. After you run the %COLORMAC macro, the color utility macros remain available for the duration of the SAS session.

Syntax

%COLORMAC;

%HELPCLR Macro Statement

writes Help information about the color utility macros to the SAS log.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Syntax

%HELPCLR<(ALL | *macro-name*)>;

Without Arguments

The %HELPCLR macro writes a list of the color utility macros for which Help information is available to the SAS log.

Optional Arguments

ALL
writes a short description and an example for each of the color utility macros to the SAS log.

macro-name
writes a short description and an example for the specified color utility macro to the SAS log. Do not include the percent sign (%) in *macro-name*.

Example The following example writes the Help information about the %CMY macro to the SAS log.

```
%COLORMAC;
%HELPCLR(CMY);
```

%CMY Macro Function

creates an RGB color name from numeric CMY component values.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Tip: Run the %HELPCLR(CYM) macro to write the %CYM macro Help information to the SAS log.

Syntax

%CMY(*cyan*, *magenta*, *yellow*);

Required Arguments

cyan*, *magenta*, *yellow
specifies the percentage of cyan, magenta, and yellow to include the RGB color value.

Range 0–100 for each value

Example The following code writes the value CX00FF00, which is green, to the SAS log.

```
%COLORMAC;
data _null_;
put "%CMY(100,0,100)";
run;
```

%CMYK Macro Function

creates a CMYK color name from numeric CMYK component values.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Tip: Run the %HELPCLR(CMYK) macro to write the %CMYK macro Help information to the SAS log.

Syntax

%CMYK(*cyan*, *magenta*, *yellow*, *black*);

Required Arguments

cyan*, *magenta*, *yellow*, *black
specifies the percentage of cyan, magenta, yellow, and black.

Range 0–100 for each value

See “CMYK Color Codes” on page 371 for more information about the color value produced by using this macro.

Example The following code writes the CMYK value 0075294F, which is purple, to the SAS log. In the PUT statement, notice that the %CMYK macro is not placed in quotation marks.

```
%COLORMAC;
data _null_;
put %CMYK(0,46,16,31);
```

```
run;
```

%CNS Macro Function

creates a SAS/GRAPH HLS color name from a color-naming scheme (CNS) color name.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Tip: Run the %HELPCLR(CNS) macro to write the %CNS macro Help information to the SAS log.

Syntax

%CNS(*CNS-color-name*);

Required Argument

CNS-color-name
specifies a CNS color name.

Note The %CNS macro accepts only CNS color names where a space is used to separate the words in the color name.

See "HLS Color Codes" on page 372 for more information about HLS color values.

"Color Naming System (CNS) Values" on page 375 or the %CNS macro Help for more information about valid color-naming scheme color names.

Example The following code writes the HLS value H04B8040, which is grayish reddish purple, to the SAS log.

```
%COLORMAC;
data _null_;
put "%CNS(GRAYISH REDDISH PURPLE)";
run;
```

%HLS Macro Function

creates an HLS color name from numeric HLS component values.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Tip: Run the %HELPCLR(HLS) macro to write the %HLS macro Help information to the SAS log.

Syntax

%HLS(*hue*, *lightness*, *saturation*);

Required Arguments

hue*, *lightness*, *saturation
specifies the hue and the percentage of lightness and saturation to be included in the HLS value.

Ranges *hue*: 0–360

lightness and *saturation*: 0–100 for each value

See "HLS Color Codes" on page 372 for more information.

Example The following example writes the HLS value H00080FF, which is blue, to the SAS log.

```
%COLORMAC;
data _null_;
put "%HLS(0,50,100)";
run;
```

%HSV Macro Function

creates an HLS color name from numeric HSV component values.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Tip: Run the %HELPCLR(HSV) macro to write the %HSV macro Help information to the SAS log.

Syntax

%HSV(*hue*, *saturation*, *value*);

Required Arguments

hue*, *saturation*, *value
specifies the hue and the percentage of saturation and brightness (*value*) to be included in the HLS value.

Ranges *hue*: 0–360

saturation and *value*: 0–100 for each value

See "HSV (or HSB) Color Codes" on page 373 and "HLS Color Codes" on page 372 for more information.

Example The following example writes the HSV value V000FFBF, which is dark red, to the SAS log.

```
%COLORMAC;
data _null_;
put "%HSV(0,100,75)";
run;
```

%RGB Macro Function

creates an RBG color name from numeric RGB component values.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Tip: Run the %HELPCLR(RGB) macro to write the %RGB macro Help information to the SAS log.

Syntax

%RGB(*red*, *green*, *blue*);

Required Arguments

red*, *green*, *blue
specifies the percentage of red, green, and blue to be included in the RGB value.

Range 0–100 for each value

See "RGB Color Codes" on page 369 for more information.

Example The following example writes the RGB value CXFFFF00, which is yellow, to the SAS log.

```
%COLORMAC;
data _null_;
put "%RGB(100,100,0)";
run;
```

%HLS2RGB Macro Function

creates an RGB color name from an HLS color name.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Note: Round-trip conversions using the HLS2RGB and RGB2HLS macros might produce ultimate output values that differ from the initial input values. For example, converting CXABCDEF (a light blue) using %RGB2HLS produces H14ACDAD. Converting this value back to RGB using %HLS2RGB returns CXAACCEE. While not identical, the colors are very similar on the display, and when printed.

Tip: Run the %HELPCLR(HLS2RGB) macro to write the %HLS2RGB macro Help information to the SAS log.

Syntax

%HLS2RGB(*HLS-color-name*);

Required Argument

HLS-color-name
specifies the HLS color name that is to be converted to an RGB color name.

See "HLS Color Codes" on page 372 and "RGB Color Codes" on page 369 for more information.

Example The following example converts HLS name H04B8040 to RGB name CX9F5F8F, which is grayish reddish purple, and writes it to the SAS log.

```
%COLORMAC;
data _null_;
put "%HLS2RGB(H04B8040)";
run;
```

%RGB2HLS Macro Function

creates an HLS color name from an RGB color name.

Requirement: You must run the %COLORMAC on page 1697 macro in the current SAS session before you can use this macro.

Note: Round-trip conversions using the RGB2HLS and HLS2RGB macros might produce ultimate output values that differ from the initial input values. For example, converting CXABCDEF (a light blue) using %RGB2HLS produces H14ACDAD. Converting this value back to RGB using %HLS2RGB returns CXAACCEE. While not identical, the colors are very similar on the display, and when printed.

Tip: Run the %HELPCLR(RGB2HLS) macro to write the %RGB2HLS macro Help information to the SAS log.

Syntax

%RGB2HLS(*RGB-color-name*);

Required Argument

RGB-color-name

specifies the RGB color name that is to be converted to an HLS color name.

See "RGB Color Codes" on page 369 and "HLS Color Codes" on page 372 for more information.

Example The following example converts RGB name CX9F5F8F to HLS name H04C7F40, which is grayish reddish purple, and writes it to the SAS log.

```
%COLORMAC;
data _null_;
put "%RGB2HLS(CX9F5F8F)";
run;
```

Appendix 9

GREPLAY Procedure Template Reference

Overview

This SAS/GRAPH code can be used to re-create the templates stored in SASHELP.TEMPLT. You can modify the code to create custom templates. For detailed information about using, creating, and modifying templates, refer to Chapter 44, “GREPLAY Procedure,” on page 1415.

H2: One Box Left and One Box Right

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef H2 des= "1 BOX LEFT, 1 BOX RIGHT"
       1/llx=0   lly=0
         ulx=0   uly=100
         urx=50  ury=100
         lrx=50  lry=0
         color=black
       2/llx=50  lly=0
         ulx=50  uly=100
         urx=100 ury=100
         lrx=100 lry=00
         color=black;
quit;
```

H2S: One Box Left and One Box Right with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef H2S des="1 BOX LEFT, 1 BOX RIGHT (WITH SPACE)"
       1/llx=0   lly=0
         ulx=0   uly=100
         urx=48  ury=100
         lrx=48  lry=0
```

```
          color=black
        2/llx=52  lly=0
          ulx=52  uly=100
          urx=100 ury=100
          lrx=100 lry=00
          color=black;
quit;
```

H3: Three Boxes Across

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef H3 des=" 3 BOXES ACROSS"
       1/llx=0    lly=0
         ulx=0    uly=100
         urx=33.3 ury=100
         lrx=33.3 lry=0
         color=black
       2/llx=33.3 lly=0
         ulx=33.3 uly=100
         urx=66.6 ury=100
         lrx=66.6 lry=00
         color=black
       3/llx=66.6 lly=0
         ulx=66.6 uly=100
         urx=100  ury=100
         lrx=100  lry=00
         color=black;
 quit;
```

H3S: Three Boxes across with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef H3S des=" 3 BOXES ACROSS (WITH SPACE)"
       1/llx=0   lly=0
         ulx=0   uly=100
         urx=30  ury=100
         lrx=30  lry=0
         color=black
       2/llx=35  lly=0
         ulx=35  uly=100
```

```
          urx=65  ury=100
          lrx=65  lry=00
          color=black
        3/llx=70  lly=0
          ulx=70  uly=100
          urx=100 ury=100
          lrx=100 lry=00
          color=black;
quit;
```

H4: Four Boxes Across

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef H4 des= "4 BOXES ACROSS"
       1/llx=0   lly=0
         ulx=0   uly=100
         urx=25  ury=100
         lrx=25  lry=0
         color=black
       2/llx=25  lly=0
         ulx=25  uly=100
         urx=50  ury=100
         lrx=50  lry=00
         color=black
       3/llx=50  lly=0
         ulx=50  uly=100
         urx=75  ury=100
         lrx=75  lry=00
         color=black
       4/llx=75  lly=0
         ulx=75  uly=100
         urx=100 ury=100
         lrx=100 lry=00
         color=black;
quit;
```

H4S: Four Boxes across with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef H4S des= "4 BOXES ACROSS (WITH SPACE)"
```

```
        1/llx=0   lly=0
          ulx=0   uly=100
          urx=22  ury=100
          lrx=22  lry=0
          color=black
        2/llx=26  lly=0
          ulx=26  uly=100
          urx=48  ury=100
          lrx=48  lry=00
          color=black
        3/llx=52  lly=0
          ulx=52  uly=100
          urx=74  ury=100
          lrx=74  lry=00
          color=black
        4/llx=78  lly=0
          ulx=78  uly=100
          urx=100 ury=100
          lrx=100 lry=00
          color=black;
quit;
```

L1R2: One Box Left and Two Boxes Right

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef L1R2 des= "1 BOX LEFT, 2 BOXES RIGHT"
        1/llx=0   lly=0
          ulx=0   uly=100
          urx=50  ury=100
          lrx=50  lry=0
          color=black
        2/llx=50  lly=50
          ulx=50  uly=100
          urx=100 ury=100
          lrx=100 lry=50
          color=black
        3/llx=50  lly=0
          ulx=50  uly=50
          urx=100 ury=50
          lrx=100 lry=00
          color=black;
quit;
```

L1R2S: One Box Left and Two Boxes Right with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef L1R2S des= "1 BOX LEFT, 2 BOXES RIGHT (WITH SPACE)"
       1/llx=0   lly=0
         ulx=0   uly=100
         urx=48  ury=100
         lrx=48  lry=0
         color=black
       2/llx=52  lly=52
         ulx=52  uly=100
         urx=100 ury=100
         lrx=100 lry=52
         color=black
       3/llx=52  lly=0
         ulx=52  uly=48
         urx=100 ury=48
         lrx=100 lry=00
         color=black;
quit;
```

L2R1: Two Boxes Left and One Box Right

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef L2R1 des= "2 BOXES LEFT, 1 BOX RIGHT"
       1/llx=0   lly=50
         ulx=0   uly=100
         urx=50  ury=100
         lrx=50  lry=50
         color=black
       2/llx=0   lly=0
         ulx=0   uly=50
         urx=50  ury=50
```

```
          lrx=50  lry=0
          color=black
        3/llx=50  lly=0
          ulx=50  uly=100
          urx=100 ury=100
          lrx=100 lry=00
          color=black;
quit;
```

L2R1S: Two Boxes Left and One Box Right with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef L2R1S des= "2 BOXES LEFT, 1 BOX RIGHT (WITH SPACE)"
       1/llx=0   lly=52
         ulx=0   uly=100
         urx=48  ury=100
         lrx=48  lry=52
         color=black
       2/llx=0   lly=0
         ulx=0   uly=48
         urx=48  ury=48
         lrx=48  lry=0
         color=black
       3/llx=52  lly=0
         ulx=52  uly=100
         urx=100 ury=100
         lrx=100 lry=00
         color=black;
quit;
```

L2R2: Two Boxes Left and Two Boxes Right

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef L2R2 des="2 BOXES LEFT, 2 BOXES RIGHT"
```

```
        1/llx=0   lly=50
          ulx=0   uly=100
          urx=50  ury=100
          lrx=50  lry=50
          color=black
        2/llx=0   lly=0
          ulx=0   uly=50
          urx=50  ury=50
          lrx=50  lry=0
          color=black
        3/llx=50  lly=50
          ulx=50  uly=100
          urx=100 ury=100
          lrx=100 lry=50
          color=black
        4/llx=50  lly=0
          ulx=50  uly=50
          urx=100 ury=50
          lrx=100 lry=0
          color=black;
quit;
```

L2R2S: Two Boxes Left and Two Boxes Right with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef L2R2S des="2 BOXES LEFT, 2 BOXES RIGHT (WITH SPACE)"
        1/llx=0   lly=52
          ulx=0   uly=100
          urx=48  ury=100
          lrx=48  lry=52
          color=black
        2/llx=0   lly=0
          ulx=0   uly=48
          urx=48  ury=48
          lrx=48  lry=00
          color=black
        3/llx=52  lly=52
          ulx=52  uly=100
          urx=100 ury=100
          lrx=100 lry=52
          color=black
        4/llx=52  lly=0
          ulx=52  uly=48
          urx=100 ury=48
          lrx=100 lry=0
```

```
          color=black;
quit;
```

U1D2: One Box Up and Two Boxes Down

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef U1D2 des= "1 BOX UP, 2 BOXES DOWN"
       1/llx=0   lly=50
         ulx=0   uly=100
         urx=100 ury=100
         lrx=100 lry=50
         color=black
       2/llx=0   lly=0
         ulx=0   uly=50
         urx=50  ury=50
         lrx=50  lry=0
         color=black
       3/llx=50  lly=0
         ulx=50  uly=50
         urx=100 ury=50
         lrx=100 lry=00
         color=black;
quit;
```

U1D2S: One Box Up and Two Boxes Down with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef U1D2S des= "1 BOX UP, 2 BOXES DOWN (WITH SPACE)"
       1/llx=0   lly=52
         ulx=0   uly=100
         urx=100 ury=100
         lrx=100 lry=52
         color=black
       2/llx=0   lly=0
```

```
          ulx=0   uly=48
          urx=48  ury=48
          lrx=48  lry=0
          color=black
        3/llx=52  lly=0
          ulx=52  uly=48
          urx=100 ury=48
          lrx=100 lry=00
          color=black;
quit;
```

U2D1: Two Boxes Up and One Box Down

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef U2D1 des= "2 BOXES UP, 1 BOX DOWN"
       1/llx=0   lly=50
         ulx=0   uly=100
         urx=50  ury=100
         lrx=50  lry=50
         color=black
       2/llx=50  lly=50
         ulx=50  uly=100
         urx=100 ury=100
         lrx=100 lry=50
         color=black
       3/llx=00  lly=0
         ulx=00  uly=50
         urx=100 ury=50
         lrx=100 lry=00
         color=black;
quit;
```

U2D1S: Two Boxes Up and One Box Down with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef U2D1S des= "2 BOXES UP, 1 BOX DOWN (WITH SPACE)"
       1/llx=0   lly=52
         ulx=0   uly=100
         urx=48  ury=100
         lrx=48  lry=52
         color=black
       2/llx=52  lly=52
         ulx=52  uly=100
         urx=100 ury=100
         lrx=100 lry=52
         color=black
       3/llx=0   lly=0
         ulx=0   uly=48
         urx=100 ury=48
         lrx=100 lry=00
         color=black;
quit;
```

V2: One Box Up and One Box Down

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef V2 des= "1 BOX UP, 1 BOX DOWN"
       1/llx=0   lly=50
         ulx=0   uly=100
         urx=100 ury=100
         lrx=100 lry=50
         color=black
       2/llx=00  lly=00
         ulx=00  uly=50
         urx=100 ury=50
         lrx=100 lry=0
         color=black;
quit;
```

V2S: One Box Up and One Box Down with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef V2S des= "1 BOX UP, 1 BOX DOWN (WITH SPACE)"
       1/llx=0   lly=52
         ulx=0   uly=100
         urx=100 ury=100
         lrx=100 lry=52
         color=black
       2/llx=00  lly=00
         ulx=00  uly=48
         urx=100 ury=48
         lrx=100 lry=0
         color=black;
quit;
```

V3: Three Boxes Vertically

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef V3 des= "3 BOXES VERTICALLY"
           /* define panel 1 */
       1/llx=0   lly=66.6
         ulx=0   uly=100
         urx=100 ury=100
         lrx=100 lry=66.6
         color=black
       2/llx=0   lly=33.3
         ulx=0   uly=66.6
         urx=100 ury=66.6
         lrx=100 lry=33.3
         color=black
       3/llx=0   lly=0
         ulx=0   uly=33.3
         urx=100 ury=33.3
         lrx=100 lry=00
         color=black;
quit;
```

V3S: Three Boxes Vertically with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef V3S des="3 BOXES VERTICALLY (WITH SPACE)"
```

```
        1/llx=0   lly=70
          ulx=0   uly=100
          urx=100 ury=100
          lrx=100 lry=70
          color=black
        2/llx=00  lly=35
          ulx=00  uly=65
          urx=100 ury=65
          lrx=100 lry=35
          color=black
        3/llx=00  lly=00
          ulx=00  uly=30
          urx=100 ury=30
          lrx=100 lry=00
          color=black;
quit;
```

WHOLE: Entire Screen Template

Start the GREPLAY procedure in line-mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
   tdef WHOLE des="ENTIRE SCREEN TEMPLATE"
        1/llx=0   lly=0
          ulx=0   uly=100
          urx=100 ury=100
          lrx=100 lry=0
          color=white;
quit;
```

Appendix 10

Differences between Base SAS ODS Graphics and SAS/GRAPH

Differences between Base SAS ODS Graphics and SAS/GRAPH

SAS provides two distinct systems for generating graphics output. The following table lists the major differences between these two systems.

Table A33.1 *Differences between ODS Graphics and SAS/GRAPH*

Graphics System	Description
ODS Graphics	ODS Graphics is a template-based graphics system that is delivered with Base SAS. In ODS Graphics, output is produced from a compiled ODS template of type STATGRAPH. Predefined templates that are supplied by SAS are stored in SASHELP.TMPLMST. User-defined templates are stored in SASUSER.TMPLMST. The Base SAS procedures that produce template-based graphics are the SGPLOT, SGPANEL, SGSCATTER, SGDESIGN, and SGRENDER. Many SAS/STAT, SAS/ETS, and SAS/QC procedures also produce template-based graphics automatically by default. Template-based graphics are always produced as files and never as GRSEG catalog entries. For template-based graphics, the ODS GRAPHICS statement is used to control the graphical environment. Template-based graphics are referred to as ODS Graphics.

SAS/GRAPH	SAS/GRAPH is a device-based graphics system that is a separate product. It must be licensed and installed at your site. In SAS/GRAPH, output is produced by graphics devices, which determine the type of output that is generated. Examples of graphics devices are SVG, PNG, GIF, ACTIVEX, and SASPRTC. The graphics devices that are supplied by SAS are stored in the SASHELP.DEVICES catalog. Most procedures that produce device-based graphics also produce GRSEG catalog entries in addition to any image files, vector files, or displayed output that are produced. SAS/GRAPH procedures that produce device-based graphics and GRSEG catalog entries include the GCHART, GPLOT, GMAP, GBARLINE, GCONTOUR, and G3D procedures. The device-based procedures that do not produce GRSEG catalog entries are the GAREABAR, GKPI, and GTILE procedures. For device-based graphics, the GOPTIONS statement is used to control the graphical environment.

Although ODS Graphics and SAS/GRAPH are distinct systems, the following SAS statements apply to both:

- ODS destination statements such as ODS HTML, ODS PDF, and so on, specify the output document format for both ODS Graphics and SAS/GRAPH.
- Global statements TITLE, FOOTNOTE, and NOTE add text to output from both ODS Graphics and SAS/GRAPH.

Conversely, the following features of each are not compatible:

- The Annotate facility for ODS Graphics and SAS/GRAPH are not interchangeable.
- The SAS/GRAPH global statements AXIS, LEGEND, PATTERN, and SYMBOL do not apply to ODS Graphics.

Appendix 11

Understanding Hexadecimal Values

Understanding Hexadecimal Values

This section provides an overview of the hexadecimal numbering system. The hexadecimal system is a base 16 numbering system where each digit represents one of the values shown in the following table.

Hexadecimal Value	Decimal Equivalent
0–9	0–9
A	10
B	11
C	12
D	13
E	14
F	15

The maximum decimal value that a hexadecimal value can represent is $16^n - 1$, where *n* is the number of digits in the hexadecimal value. A two-digit hexadecimal value can represent a maximum decimal value of 255, and a four-digit hexadecimal value can represent a maximum decimal value of 65,535. To convert a hexadecimal

number to decimal, sum the product of each hexadecimal character and its base power, 16^n, where *n* is the digit's significance. For example, to convert hexadecimal value C8A4 to decimal manually, do the following:

$$\left(12 \times 16^3\right) + \left(8 \times 16^2\right) + \left(10 \times 16\right) + 4 = 51364$$

To convert from decimal to hexadecimal manually, iteratively divide the decimal value by 16. In each iteration, multiply the remainder by 16 to get the hexadecimal character for that iteration, and then use the quotient in the next iteration until the quotient is zero. This method generates the hexadecimal value from the least-significant digit to the most significant digit. The following table demonstrates how to convert decimal value 51,364 back to its hexadecimal value.

Divide by 16	Quotient	Remainder	16 x Remainder	Hexadecimal Value
51364 / 16	3210	0.25	4	4
3210 / 16	200	0.625	10	A
200 / 16	12	0.5	8	8
12 / 16	0	0.75	12	C

Since the hexadecimal characters are generated from the least-significant digit to the most significant, the result is C8A4.

Apart from the base and the value representation, the methods for manipulating hexadecimal values, such as addition, subtraction, and multiplication, are the same as those that are used in the base 10 system. Here are some examples.

Base 10	Base 16
9 + 1 = 10	9 + 1 = A
8 + 8 = 16	8 + 8 = 10
3 x 85 = 255	3 x 55 = FF

Many pocket calculators and calculator programs enable you to manipulate hexadecimal values and convert values between hexadecimal and decimal. You can also use a DATA step in SAS to manipulate and convert hexadecimal values. Here is a simple macro that uses the INPUT statement in a DATA step to convert a hexadecimal value to its equivalent decimal value.

```
%macro hex2dec(hex);
  data _null_;
    msg=cat("#", %upcase("&hex"), " = ", input("&hex", hex.));
    put msg;
  run;
%mend hex2dec;
```

The following example shows how to use this macro to convert the hexadecimal value C8A4 to its decimal equivalent.

```
%hex2dec(c8a4);
```

```
#C8A4 = 51364
```

Here is a simple macro that uses the PUT statement in a DATA step to convert a decimal value to its equivalent hexadecimal value.

```
%macro dec2hex(dec);
  data _null_;
    msg=cat("&dec = #", put(&dec, hex4.));
    put msg;
  run;
%mend dec2hex;
```

The following example shows how to use this macro to convert the decimal value 51,364 back to its hexadecimal equivalent.

```
%dec2hex(51364);
51364 = #C8A4
```

For information about macros, see *SAS Macro Language: Reference*. For information about the INPUT and PUT DATA step functions, see *SAS Functions and CALL Routines: Reference*.

When you use the hexadecimal system to define you own colors, the exact value for the color is not as significant as the relationship of the digits in the value to each other, and the relationship of the value of this color to other colors.

Appendix 12

Transporting and Converting Graphics Output

About Transporting and Converting Graphics Output

You can use the following methods to transport and convert graphics output within SAS:

- Use the CPORT and CIMPORT procedures in Base SAS software to transport catalogs that contain graphics output to other operating environments that are running the same version of SAS/GRAPH software.
- Use a LIBNAME statement and the CATALOG procedure to convert catalogs from Version 6 to Version 7 or later.

Transporting Catalogs across Operating Environments

Transporting Catalogs Using CPORT and CIMPORT

Use the CPORT and CIMPORT procedures to transport catalogs and catalog entries from one machine to another machine running in a different operating environment. In addition to graphics output stored in GRSEG catalog entries, SAS/GRAPH software produces several other files that you can transport from host environment to host environment. These other files include

- colors maps
- templates
- fonts
- device descriptions.

To transport catalog entries that contain graphics output (catalog entries of type GRSEG), follow these steps:

1. Use the CPORT procedure to create a transport file from the catalog entries in the current host environment. A transport file is a sequential file that contains the catalog in SAS transport format. To create a transport file, you must specify a catalog to be converted and a fileref for the transport file.

   To retain the original order of the GRSEG entries in the catalog, use SELECT= in the PROC CPORT statement to export individual graphs in the order in which they were created. Otherwise, when you use the GREPLAY procedure to list the graphics entries in the imported catalog, the procedure will list the entries in alphabetical order, rather than the order in which they were created.

   Note: Only the GREPLAY procedure can list catalog entries in the order in which they were created. All other procedures list entries in alphabetical order.

   To export a catalog that contains groups of entries created using the GREPLAY procedure, you must use SELECT= in the PROC CPORT statement to select the names of the groups, rather than the names of individual graphs, to be included in the transport file. If you export the entire catalog without using SELECT=, the groups are not maintained in the catalog created when you import the transport file in the new host environment.

   When you use the CPORT procedure, messages in the SAS log identify the catalog entries that have been placed in the transport file. If the catalog entry was created by replaying several graphs into a template, the log messages list the names of all of the entries that contributed to the templated graph.

2 Move the transport file to the target machine, if necessary. You must move the transport file in binary format. If you do not move the transport file in binary format, the CIMPORT procedure cannot read the file that you create.

3 Once you have moved the transport file to the target machine, import the transport file into a catalog in the new host environment using the CIMPORT procedure. The entries are imported in the order specified in SELECT= in the PROC CPORT statement used to create the transport file.

   The SELECT= option in the PROC CIMPORT statement does not affect the order of the imported entries.

Note: You must use the CIMPORT procedure from the current version of SAS. The CIMPORT procedure in a previous release cannot read a transport file created by the CPORT procedure in the current version.

For details about using the CPORT and CIMPORT procedures, see the *Base SAS Procedures Guide*.

Example of Transporting GRSEGs

This example shows how to port three entries from the catalog MYLIB.GRAPHS.

First, the CPORT procedure writes selected graphs from MYLIB.GRAPHS to the transport file TRANFILE. The SELECT option names the graphs to be ported.

```
libname mylib "SAS-data-library";
filename tranfile "external-file";

proc cport file=tranfile
     catalog=mylib.graphs
     select=(GPLOT.GRSEG GPLOT1.GRSEG GPLOT3.GRSEG);
run;
```

Once the transport file has been moved to the new host environment using communications software or tape, the CIMPORT procedure creates a new catalog called MYLIB.GRAPHS on the new machine.

```
libname mylib "SAS-data-library";
filename tranfile "external-file";

proc cimport catalog=mylib.graphs
     infile=tranfile
     select=(GPLOT.GRSEG GPLOT1.GRSEG  GPLOT3.GRSEG);
run;
```

Example of Transporting Color Maps and Templates

To transport color maps (catalog entries of type CMAP) and templates (catalog entries of type TEMPLATE) from one host environment to another, use the CPORT

and CIMPORT procedures. For example, you could export a color map from the NEWLIB.CMAPS catalog using the following statements:

```
filename tranfile "external-file";
libname newlib "SAS-data-library";

proc cport file=tranfile catalog=newlib.cmaps select=(mymap.cmap);
run;
```

After moving the transport file to the new host environment, you can import the color map using the following statements:

```
filename tranfile "external-file";
libname newlib "SAS-data-library";

proc cimport infile=tranfile catalog=newlib.cmaps;
run;
```

Example of Transporting Fonts

To transport fonts (catalog entries of type FONT) from one operating system to another, use the CPORT and CIMPORT procedures. For example, you could export a font from the GFONT0.FONTS catalog using the following statements:

```
filename tranfile "external-file";
libname gfont0 "SAS-data-library";

proc cport file=tranfile
     catalog=gfont0.fonts
     select=(figures.font);
run;
```

After moving the transport file to the new host environment, you can import the font using the following statements:

```
filename tranfile "external-file";
libname gfont0 "SAS-data-library";

proc cimport infile=tranfile catalog=gfont0.fonts;
run;
```

Example of Transporting Device Attributes and Device Entries

To transport device entries (catalog entries of type DEV) from one operating environment to another, use the CPORT and CIMPORT procedures. For example, you could export a device entry from the GDEVICE0.DEVICES catalog using the following statements:

```
filename tranfile "external-file";
libname gdevice0 "SAS-data-library";
```

```
proc cport file=tranfile
     catalog=gdevice0.devices
     select=(cgm.dev);
run;
```

After moving the transport file to the new host environment, you can import the device entry using the following statements:

```
filename tranfile "external-file";
libname gdevice0 "SAS-data-library";

proc cimport infile=tranfile catalog=gdevice0.devices;
run;
```

Converting Catalogs to a Different Version of SAS

To convert catalogs to a different version of SAS, use the LIBNAME statement and the CATALOG procedure.

Note: You will not be able to use your old catalogs without transporting them first.

Before using PROC CATALOG, you must assign librefs to both catalogs and specify the Version 6 Compatibility Engine (saseb) on the input catalog LIBNAME. Then use PROC CATALOG with a COPY statement to convert a catalog from Version 6 to Version 7 or later. For details about using the CATALOG procedure, see the *Base SAS Procedures Guide*.

For example, the following statements can be submitted from Version 8 to assign the Version 6 Compatibility Engine and convert a catalog from Version 6 to Version 8.

```
libname v6lib saseb "SAS-data-library";
libname v8lib "SAS-data-library";

proc catalog catalog=v6lib.v6cat;
   copy out=v8lib.v8cat;
run;
```