THE POWER TO KNOW.

SAS/GRAPH® 9.2 Reference

Second Edition

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2010. *SAS/GRAPH® 9.2 Reference, Second Edition*. Cary, NC: SAS Institute Inc.

SAS/GRAPH® 9.2 Reference, Second Edition

978-1–60764–449–1

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

1st electronic book, May 2010
1st printing, May 2010

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents

What's New

Overview

The changes and enhancements for SAS/GRAPH 9.2 are very extensive. Highlights include the following:

- The new SAS/GRAPH statistical graphics suite provides a new set of procedures, a new language, and a graph editor specifically designed for creating and editing statistical graphics.
- All SAS/GRAPH procedures now support ODS styles for all devices.
- SAS/GRAPH now automatically selects an appropriate device and style for all open destinations.
- SAS/GRAPH now provides TrueColor support, which allows over 16 million colors in a single image.
- The new Network Visualization Workshop enables you to visualize and investigate the patterns and relationships hidden in network data (node-link data).
- The new GKPI procedure generates several key performance indicators.
- The new GTILE procedure generates tile charts.
- The new GEOCODE procedure enables you to add geographic coordinates to data sets that contain location information such as mailing addresses or to perform geolocation with non-address location data.
- The new GINSIDE procedure determines which polygon in a map data set contains the geographic coordinates in your input data set.
- All procedures now support graphics output filenames up to 256 characters long.
- Many procedures have significant enhancements and new options. See "Procedures" on page xix for a complete list.
- The new Scalable Vector Graphics devices enable you to generate SVG output.
- Several new map data sets, as well as new feature data sets, have been added to the MAPS library. Several existing map data sets have been updated.

The SAS/GRAPH Statistical Graphics Suite

ODS Statistical Graphics (referred to as ODS Graphics for short) is major new functionality for creating statistical graphics that is available in a number of SAS

software products, including SAS/STAT, SAS/ETS, SAS/QC, and SAS/GRAPH. Many statistical procedures have been enabled to use this functionality, and these procedures now produce graphs as automatically as they produce tables. In addition, the new statistical graphics (SG) family of SAS/GRAPH procedures use this functionality to produce plots for exploratory data analysis and customized statistical displays.

ODS Graphics includes the new SAS/GRAPH statistical graphics suite. This suite provides the following new features:

SAS/GRAPH statistical graphics procedures
: provide a simple syntax for creating graphics commonly used in exploratory data analysis and for creating customized statistical displays. These new procedures include the SGPANEL, SGPLOT, and SGSCATTER procedures. In addition, the SGRENDER procedure provides a SAS procedure interface to the new Graph Template Language. For more information, including changes and enhancements for SAS 9.2 Phase 2, see *SAS/GRAPH: Statistical Graphics Procedures Guide*.

Graph Template Language (GTL)
: is the underlying language for the default templates that are provided by SAS for procedures that use ODS Statistical Graphics. You can use the GTL either to modify these templates or to create your own highly customized graphs. Templates written with the GTL are built with the TEMPLATE procedure. For more information about Graph Template Language, see the *SAS/GRAPH: Graph Template Language Reference* and the *SAS/GRAPH: Graph Template Language User's Guide*.

ODS Graphics Editor
: is an interactive editor that enables you to edit and enhance graphs that are produced by procedures that use ODS Statistical Graphics. You can use the ODS Graphics Editor to modify the existing elements of a graph such as titles and labels, or to add features such as text annotation for data points. For more information, including changes and enhancements for SAS 9.2 Phase 2, see *SAS/GRAPH: ODS Graphics Editor User's Guide*.

ODS Graphics Designer
: provides a point-and-click interface for creating ODS graphics. Using the ODS Graphics Designer does not require knowledge of ODS templates or the Graph Template Language. With the ODS Graphics Designer, you can easily create multi-cell graphs, classification panels, scatter plot matrices, and more. You can save your output as an image file or as an ODS Graphics Designer file (SGD file) that you can edit later.

  The ODS Graphics Designer is available beginning with SAS 9.2 Phase 2. For more information, including information about changes and enhancements for the third maintenance release of SAS 9.2, see *SAS/GRAPH: ODS Graphics Designer User's Guide*.

Note: For additional information on the ODS Statistical Graphics functionality, see *SAS Output Delivery System: User's Guide* and *SAS/STAT User's Guide*. △

The SAS/GRAPH Network Visualization Workshop

The Network Visualization (NV) Workshop application 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, Web sites, 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. To start NV Workshop, select **Start ▸ Programs ▸ SAS ▸ SAS GRAPH NV Workshop 2.1**.

For more information, including changes and enhancements for SAS 9.2 Phase 2, see *SAS/GRAPH: Network Visualization Workshop User's Guide*.

Support for Multiple Open ODS Destinations

If you have multiple ODS destinations open, SAS/GRAPH automatically selects the appropriate device for each destination. In addition, each graph uses the ODS style associated with each destination. You do not need to specify a device or style to get optimal results. For example, if you do not specify a device, then SAS/GRAPH automatically selects the PNG device for the HTML destination if it is open and the SASEMF device for the RTF destination.

Also, if you have multiple ODS destinations open and you are using a device other than the Java or ActiveX devices (ACTIVEX, JAVA, ACTXIMG, or JAVAIMG), a different GRSEG is created for each open destination. The GRSEGs for the first destination are stored in WORK.GSEG. The GRSEGs for any other open destinations are stored in catalogs named according to the destinations, for example, WORK.HTML.

Support for ODS Styles

All SAS/GRAPH procedures and devices now support ODS styles. By default, all colors, fonts, symbols, and graph sizes are derived from the current style. Procedure statement options and SAS/GRAPH GOPTIONS override individual elements of the style, so you can easily customize the appearance of any graph.

Additionally, the colors used by the styles have been updated to enhance the appearance of your graphics output.

The use of ODS styles by default is controlled by the GSTYLE system option. For information on the GSTYLE option, refer to *SAS Language Reference: Dictionary*.

Devices

- The new Scalable Vector Graphics devices enable you to create SVG graphs. The SVG devices (SVG, SVGZ, SVGView, and SVGT) are supported for the LISTING, HTML, and PRINTER destinations.
- The default device for the ODS HTML destination has changed from GIF to PNG, which provides TrueColor support. Using the PNG device might result in graphs that have spacing or size differences, such as slightly narrower bars in bar charts.
- Data tips are now supported by the JAVAIMG device.
- Several devices have been added for compatibility with previous releases of SAS/GRAPH. These devices are named Z*device*, where device is the name of the device in previous releases.
  - The following devices ignore the FONTRENDERING= system option and force host font rendering (see "Fonts and Font Rendering" on page xviii): ZGIF, ZGIF733, ZGIFANIM, ZJPEG, ZPNG, ZSASBMP, ZTIFFB, ZTIFFBII, ZTIFFBMM, ZTIFFG3, ZTIFFG4, and ZTIFFP.

- The following devices support printer-resident fonts only: ZPCL5, ZPDF, ZPDFC, ZPSCOLOR, ZPSEPSFC, ZPSL, and ZPSLEPSF. They will not work well with ODS styles (see "Support for ODS Styles" on page xvii) because they do not support TrueType fonts, which are used by the styles.
- Several Universal Printing shortcut devices have been added. The UPCL5, UPCL5E and UPCL5C devices have been added for printing support. The UPDF and UPDFC devices have been added for PDF support. The UPSL and UPSLC devices have been added for PostScript support. See also "Using Universal Printer Shortcut Devices" on page 75.

Colors

- SAS/GRAPH now provides TrueColor support, which allows over 16 million colors in a single image.
- The number of colors in the default color list has been increased to 38.

Fonts and Font Rendering

- The following fonts are now obsolete: DAVID, NHIRA, NKATA.
- Some of the characters in the Hebrew font are mapped differently to the Roman character set than they were previously.
- Fonts are now rendered using the FreeType engine. This new font rendering might result in fonts appearing larger than they did in previous versions of SAS/GRAPH. See also "Changing the Appearance of Output to Match That of Earlier SAS Releases" on page xix.
- Many new TrueType fonts have been added. These new fonts are listed in Table 0.1 on page xviii.

Table 0.1 TrueType Fonts Supplied by SAS

Albany AMT*	Thorndale Duospace WT SC	GungsuhChe
Cumberland AMT*	Thorndale Duospace WT TC	Dotum
Thorndale AMT*	Arial Symbol*	DotumChe
Symbol MT	Times New Roman Symbol*	Gulim
Monotype Sorts	MS PMincho	GulimChe
Monotype Sans WT J	MS Mincho	NSimSun
Monotype Sans WT K	MS PGothic	SimHei
Monotype Sans WT SC	MS UI Gothic	SimSun
Monotype Sans WT TC	Batang	PMingLiU
Thorndale Duospace WT J	BatangChe	MingLiU
Thorndale Duospace WT K	Gungsuh	HeiT

* Albany AMT, Cumberland AMT, Thorndale AMT, Arial Symbol, and Times New Roman Symbol are font families. Normal, bold, italic, and bold italic versions of these fonts are provided.

Changing the Appearance of Output to Match That of Earlier SAS Releases

SAS/GRAPH 9.2 introduces many new features that significantly change the default appearance of your SAS/GRAPH output. To produce output that looks as if it was 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 153.
- 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 native SAS/GRAPH devices (see "Device Categories And Modifying Default Output Attributes" on page 72). For example, this option works for GIF, TIFFP, JPEG, and ZPNG devices, but it is not applicable to PNG, SVG, or SASPRT* devices.
- Specify DEVICE=ZGIF on the GOPTIONS statement when you are sending output to the HTML destination.
- In other cases where your application specifies a device, specify a compatible Z device driver, if applicable. See "Devices" on page xvii for more information.

Procedures

Support for Long Filenames

The NAME= option for each procedure has been enhanced to allow you to specify filenames up to 256 characters long for graphics output files (PNG files, GIF files, and so on). See the documentation for the specific SAS/GRAPH procedures for more information.

GAREABAR Procedure

The GAREABAR procedure has the following new options and enhancements:

- The GAREABAR procedure now supports the BY and LEGEND statements.
- The CONTINUOUS option enables you to display a range of numeric values along the width axis.
- The DESCRIPTION= option specifies the description of the catalog entry for the plot.
- The LEGEND= option assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.
- The NOLEGEND option suppresses the legend automatically generated by the SUBGROUP= option.

GBARLINE Procedure

The GBARLINE procedure has the following new options and enhancements:

- The PLOT statement supports the creation of multiple plot lines on a single bar chart.

- The SUBGROUP= option divides the bar into segments according to the values of the SUBGROUP variable values.
- The HTML= option on the PLOT statement supports data tips and drill-down links on the markers of the line plot.
- The HTML_LEGEND= option supports data tips and drill-down legend links.
- The IMAGEMAP= option enables you to generate an image map with drill-down functionality in an HTML file.
- The LEGEND= option enables you to generate both BAR and PLOT legends.
- The LEVELS=ALL option has been enhanced to display any number of midpoints.
- The ASCENDING and DESCENDING options now join plot points from left-to-right by default when the bars are reordered.
- The PLOT statement now supports several options for references lines on the plot (right) response axis.
  - The AUTOREF option draws a reference line at each major tick mark.
  - The REF= option draws reference lines at the specified positions.
  - The CREF=, LREF=, and WREF= options enable you to specify the color, line style, and width of user-defined reference lines.
  - The CAUTOREF=, LAUTOREF=, and WAUTOREF= options enable you to specify the color, line style, and width of AUTOREF lines.
- The WREF= and WAUTOREF= options on the BAR statement enable you to specify the width of reference lines on the bar (left) response axis.
- The PLOT statement now supports the following options:

CAXIS=	specifies a color for the tick marks and the axis area frame
CTEXT=	specifies a color for all text on the plot response axis and legend
NOAXIS	suppresses the right PLOT response axis

GCHART Procedure

The GCHART procedure has the following new options and enhancements:

- The COUTLINE= option has been enhanced to include outlines on cylinder-shaped bars.
- The GAXIS= option is now supported by the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices.
- The MAXIS= option is now supported by the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices.
- The NOPLANE option enables you to remove walls from three–dimensional bar charts.
- The PCTSUM option in the HBAR statement displays a column of percentages for the sum variable values.
- The new PCTSUMLABEL= option enables you to specify the text for the column label for the PCTSUM statistic in the table of statistics.
- The PLABEL= option enables you to specify the font, height, and color of pie slice labels.
- The NOZERO option on the BAR statement is now supported by the JAVA and JAVAIMG devices.
- The RADIUS= option on the PIE statement enables you to specify the radius of the pie chart.

- The RAXIS= option is now supported by the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices.
- The SHAPE= option on BLOCK statement is now supported by the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices.
- The WREF= and WAUTOREF= options enable you to specify the width of reference lines.
- The pie and bar name variable now support up to 256 characters.

GCONTOUR Procedure

The GCONTOUR procedure has the following changes and enhancements.

- When used with the Java and ActiveX devices, the LJOIN option displays filled contour areas with separated by contour lines.
- When used with the Java and ActiveX devices, the SMOOTH option produces smooth gradient areas between levels.
- The WAUTOHREF= and WAUTOVREF= options specify the line width for reference lines generated with the AUTOHREF and AUTOVREF options, respectively.
- The WHREF= and WVREF= options specify the line width for reference lines generated with the HREF= and VREF= options, respectively.

GEOCODE Procedure

The new GEOCODE procedure enables you to add geographic coordinates (latitude and longitude) to data sets that contain location information such as mailing addresses. You can also perform geolocation, which is adding geographic coordinates to non-address locations such as sale territories.

For SAS 9.2 Phase 2 and later, the new RANGE geocoding method enables you to perform geolocation for IP addresses.

For the third maintenance release of SAS 9.2, the new STREET geocoding method enables you to perform geolocation for street addresses.

GINSIDE Procedure

The new GINSIDE procedure determines which polygon in a map data set contains the X and Y coordinates in your input data set. For example, if your input data set contains coordinates within Canada, you can use the GINSIDE procedure to identify the province for each data point.

GKPI Procedure

The new GKPI procedure generates key performance indicators, including sliders, bullet graphs, speedometers, dials, and traffic lights. This GKPI procedure is supported by the JAVAIMG device only.

GMAP Procedure

The GMAP procedure has the following new features:

- The AREA statement enables you to control the appearance of regions in block maps and prism maps.

- The CDEFAULT= option specifies the color for empty map areas.
- The DENSITY= option enables you to reduce the number of map points that are drawn.
- The RELZERO= option specifies that the heights of bars and regions are relative to zero, rather than the minimum value.
- The STATISTIC= option specifies a statistic to use for the response variable.
- The STRETCH option stretches the extents of a map to fill the output device.
- The UNIFORM option specifies that each map that is created when you use the BY statement uses the same colors and legend.
- The WOUTLINE= option on the BLOCK and CHORO statements is now supported by the JAVA and JAVAIMG devices.

GPLOT Procedure

The GPLOT procedure has the following new options and enhancements:

- The BFILL= option enables you to generate gradient, solid-filled bubble plots.
- The FRONTREF= option specifies that reference lines are drawn in front of filled areas.
- The OVERLAY option is no longer required to display a legend when the PLOT (or PLOT2)statement specifies only one plot.
- The WAUTOHREF= and WAUTOVREF= options specify the line width for reference lines generated with the AUTOHREF and AUTOVREF options, respectively.
- The WHREF= and WVREF= options specify the line width for reference lines generated with the HREF= and VREF= options, respectively.
- Enhanced features in box plots enable you to click on the interior of the boxes for simple drill-down functionality. Previously, you could click only on visible box elements. Now, you can click anywhere inside the box to drill down to more detailed data.

GPROJECT Procedure

The NODATELINE option enables contiguous projections when projecting maps that cross the line between 180 degrees and –180 degrees longitude.

The following options for the GPROJECT procedure have been renamed:

Old Name	New Name
DEGREE	DEGREES
PARALEL1	PARALLEL1
PARALEL2	PARALLEL2

GRADAR Procedure

The GRADAR procedure has the following new options and enhancements:

- The CALENDAR option produces a chart showing twelve equal-sized segments, one for each month of the year.

- The NLEVELS= option specifies the number of colors to use in calendar charts.
- The NOLEGEND option turns off the automatically generated legend.
- The SPOKESCALE= option specifies whether every spoke is drawn to the same scale or each spoke is drawn to a different scale.
- The WINDROSE option produces a windrose chart, which is a type of histogram.
- The FREQ= option now supports only non-zero integers. Zero and negative values are dropped. Decimal values are truncated to integers.
- The WEIGHT= option is no longer supported.
- The GRADAR procedure now draws missing overlay values to the center. Previously, missing values were drawn to zero.

GREMOVE Procedure

The GREMOVE procedure has the following new options:

- The FUZZ= option specifies an error tolerance for the point matching algorithm.
- The NODECYCLE option enables some types of polygons to be closed properly.

GTILE Procedure

The new GTILE procedure enables you to create and display tile charts using the Java or ActiveX device drivers. Tile charts are designed for visualizing a large quantity of hierarchical-type data and are sometimes referred to as rectangular tree maps. Tile charts display rectangles of varying sizes and colors based on the magnitude of the variables specified and provides drill-down links to more detailed data.

MAPIMPORT Procedure

The ID statement for the MAPIMPORT procedure enables you to group related polygons.

Global Statements

- The REPEAT= option on the LEGEND statement enables you to specify the number of times a plot symbol is displayed in a single legend item in the legend.
- The VALUE=EMPTY option on the PATTERN statement is now supported by three-dimensional bar charts.
- The STAGGER option offsets the axis values on a horizontal axis.
- The TICK= suboption on the VALUE= option of the LEGEND statement is now supported by the Java Map Applet.
- The ROWMAJOR and COLMAJOR options on the LEGEND statement enable you to control whether legend entries are listed by row or by column.

Graphics Options

- The ACCESSIBLE graphics option generates descriptive text and the summary statistics that are represented by the graph. This option is valid for the Java and ActiveX devices only.
- The ALTDESC option enables you to specify whether the text specified in the DESCRIPTION= option is used as the data tip text.
- The TRANSPARENCY option is supported by the ACTIVEX and ACTXIMG devices when the output is used in a PowerPoint presentation.

Transparent Overlays

Transparent overlays from GIF files are now supported in SAS/GRAPH output. You can use transparent GIFs with the IMAGE function in the Annotate facility and with the IBACK and IFRAME graphics options.

ActiveX Control

The following are enhancements for the ActiveX Control:

- The ActiveX control now displays calendar and windrose charts generated by the GRADAR procedure.
- The control also displays tile charts created by the new GTILE procedure.
- Support for UNICODE fonts has been added.
- A new field in the user interface enables you to provide interactive graphs in Microsoft Powerpoint slideshows.
- The user interface now enables you to specify the properties of scroll bars in your graph.
- Data tips are supported for scatter plots generated with the GCONTOUR procedure.
- Enhanced support of the Annotate Facility listed under "The Annotate Facility" on page xxv.

Java Map Applet

The Java Map Applet user interface enables you to change block sizes.

Support has been added for the MENUREMOVE parameter, which enables you to remove menu items from the applet user interface.

Java Tilechart Applet

The new Java Tile Chart applet creates and displays tile charts. Tile charts are designed for visualizing a large quantity of hierarchical-type data and are sometimes referred to as rectangular tree maps. They display rectangles of varying sizes and colors based on the magnitude of the variables specified and provide drill-down links to

more detailed data. You can generate the applet with the GTILE procedure and the JAVA device.

The Annotate Facility

The following new features are available for the Annotate facility:

- The ANGLE, CBORDER, CBOX, LINE, and ROTATE variables are now supported by the ACTIVEX and ACTXIMG devices.
- The ARROW function and %ARROW macro enable you to draw arrows.
- A new value for the HSYS= option, 'D', specifies points as the unit of measurement for font sizes.
- The IMAGE function is now supported by the JAVA and JAVAIMG devices.
- The WIDTH variable for the PIE function specifies the thickness of the outline around the pie slice.

New Map Data Sets

New map data sets are provided for Antarctica (ANTARCTI, ANTARCT2), Montenegro (MONTENEG, MONTENE2), Romania (ROMANIA, ROMANIA2), Rwanda (RWANDA, RWANDA2), and Serbia (SERBIA, SERBIA2).

The continent map data sets now have corresponding feature data sets (ANTARCT2, AFRICA2, EUROPE2, OCEANIA2, NAMERIC2, SAMERIC2).

Note: Antarctica uses the new continent code 97. △

Updated Map Data Sets

Some of the map data sets in the MAPS library have been updated. Table 0.2 on page xxv contains a list of the changes.

Table 0.2 Changes to the Map Data Sets

Data Set(s)	Changes
Continent data sets (ASIA, AFRICA, EUROPE, NAMERICA, OCEANIA, SAMERICA)	updated to include new geographic features. Each data set includes a new DENSITY variable.
	Brunei, Indonesia, and the Philippines have moved from OCEANIA to ASIA. The continent code for these countries has changed from 96 to 95.
	OCEANIA replaces SPACIFIC as continent 96. Tasmania has been added to the OCEANIA data set.
CHINA, CHINA2	updated with new province names and ID numbers. The new OLDID and OLDIDNAME variables in the CHINA2 data set contain the old ID numbers and province names.
	Because the ID numbers and province names have changed, you might need to change your response data in any existing SAS programs that use these data sets.

Data Set(s)	Changes
GERMANY, GERMANY2	updated with new districts and states. The following new variables have been added: □ AREA □ DISTNAME □ DISTRICT □ ID2 The IDNAME variable contains the values that were previously in IDNAME2. The IDNAME2 variable has been removed.
INDIA, INDIA2	updated with new states and new ID numbers. The new OLDID variable in the INDIA2 data set contains the old ID numbers. Additionally, the IDNAME2 variable in the INDIA2 data set contains alternate spellings for the state names. The INDIA data set contains a new DENSITY variable. Because the ID numbers have changed, you might need to change your response data in any existing SAS programs that use these data sets.
ITALY, ITALY2	updated with new provinces and ID numbers. The new OLDID variable in the ITALY2 data set contains the old ID numbers. The ITALY data set contains new DENSITY, NUTS, and REGNAME2 variables. Because the ID numbers have changed, you might need to change your response data in any existing SAS programs that use these data sets.
JAOSAKA, JAOSAKA2	updated with new ID values. The new TYPE variable in JAOSAKA2 contains feature types.
JATOKYO, JATOKYO2	updated with new ID values. The new TYPE variable in JATOKYO2 contains feature types.
LUXEMBOU, LUXEMBO2	updated with more detail and new variables. The LUXEMBOU data set has a new DENSITY variable. The LUXEMBO2 data set has the following new variables: □ DISTNAME □ DISTRICT □ IDNAME2 □ NUTS4

Data Set(s)	Changes
NAMES (feature table for the WORLD data set)	contains three new variables: ID2 for territories, specifies the ID values for the countries that the territory is associated with. For example, Greenland has an ID2 value of **315** because it is a territory of Denmark. If a territory is claimed by more than one country, its ID2 value might consist of several three–digit ID values to identify each country. _REGION_ specifies a geographic region for each country or territory. For example, Panama belongs to the **Central America** region. TERRITORY for territories, describes the association between the territory and the country or countries that are identified by ID2. For example, Togo is described as **Overseas territory of France**.
PHILIPPI, PHILIPP2	updated with more detail and new variables. The PHILIPPI data set has a new DENSITY variable. The PHILIPP2 data set has the following new variables: □ ISLANDG □ ISLAND_GROUP □ OLDID □ PROVINCE □ PSGC_PROV □ PSGC_REG □ REGION □ REGNAME □ REGNAME2 The ID numbers for these data sets have changed. You might need to change your response data in any existing SAS programs that use these data sets.
POLAND, POLAND2	updated with new values and variables. The POLAND data set has a new DENSITY variable. The POLAND2 data set has new PROVNAME and PROVNAME2 variables.
SPACIFIC	renamed to OCEANIA.
SPAIN, SPAIN2	updated with values and new variables. The SPAIN data set contains a new DENSITY variable. The new OLDID variable in the SPAIN2 data set contains the old ID numbers. The new REGION and REGNAME variables identify regions. The new IDNAME2 and REGNAME2 variables contain alternate spellings for the province and region names. Because the ID numbers have changed, you might need to change your response data in any existing SAS programs that use these data sets.

Data Set(s)	Changes
SWEDEN, SWEDEN2	updated with new provinces and ID numbers. The SWEDEN data set contains a new DENSITY variable. The new OLDID variable in the SWEDEN2 data set contains the old ID numbers. The new REGNAME variable contains region names. Because the ID numbers have changed, you might need to change your response data in any existing SAS programs that use these data sets.
SWITZERL, SWITZER2	updated with new province names and ID numbers, and new variables. The new OLDID and OLDNAME variables in the SWITZER2 data set contains the old names and ID numbers. The SWITZERL data set contains new DENSITY and LAKE variables. Because the province names and ID numbers have changed, you might need to change your response data in any existing SAS programs that use these data sets.
THAILAND, THAILAN2	updated with more detail and new variables. The THAILAND data set has a new DENSITY variable. The THAILAN2 data set has the following new variables: □ IDNAME2 □ OLDID □ REGION □ REGNAME The provinces have new ID numbers. The new OLDID variable in the THAILAN2 data set contains the old ID numbers. Because the ID numbers have changed, you might need to change your response data in any existing SAS programs that use these data sets.
UKRAINE, UKRAINE2	updated with more detail and new variables. The UKRAINE data set has a new DENSITY variable. The UKRAINE2 data set has the following new variables: □ IDNAME2 □ OLDID □ OLDIDNAME The provinces have new names and ID numbers. The new OLDIDNAME and OLDID variables in the UKRAINE2 data set contain the old province names and ID numbers. Because the province names and ID numbers have changed, you might need to change your response data in any existing SAS programs that use these data sets.

Data Set(s)	Changes
US, USCENTER, USCITY	Puerto Rico added as state 72. The new STATECODE variable in the US and USCITY data sets contains two-letter state abbreviations. The USCITY data set has new cities, and some city names have been standardized. The PLACE variable now includes the state FIPS code as the first two digits. *Note:* The projected X and Y values might be different due to the need to re-project the data sets with the addition of more cities in USCITY. △
VIETNAM, VIETNAM2	updated with more detail and new variables. The VIETNAM data set has a new DENSITY variable. The VIETNAM2 data set has the following new variables: □ PROVINCE □ REGION □ REGNAME The ID numbers for these data sets have changed. You might need to change your response data in any existing SAS programs that use these data sets.

Data Set(s)	Changes
WORLD	simplified to use fewer observations. In addition, the following changes have been made: □ The values are now projected using the CYLINDRI algorithm. □ Continent 96 has been renamed from South Pacific to Oceania. □ Antarctica has been added as continent 97. □ Brunei, Indonesia, and the Philippines have been reassigned from continent 96 to continent 95. □ French Southern Territories and Heard & McDonald Islands have been reassigned from continent 96 to continent 97. □ St. Helena has been reassigned from continent 91 to continent 94. □ The former country of Yugoslavia has been split into Serbia and Montenegro. □ Newfoundland has been added to Canada (ID 260). □ Tasmania has been added to Australia (ID 160). □ More data points are included for Cuba (ID 300). □ The Galapagos Islands have been added to Ecuador (ID 325). □ Hong Kong is now included as part of China.
YUGOSLA, YUGOSLA2	replaced by the new SERBIA, SERBIA2, MONTENEG, MONTENE2 data sets.

Map Data Set Descriptions

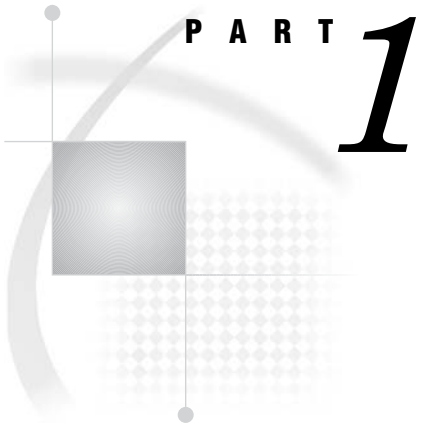
Descriptive labels have been added to the map data sets in the MAPS library.

New Data Set for Military ZIP Codes

The new ZIPMIL data set in the SASHELP library contains ZIP codes for U.S. military post offices.

Changes in SAS/GRAPH Documentation

- Information about the DS2CSF macro has been removed. The functionality of the DS2CSF macro is available through the new GKPI procedure.
- Information about the META2HTM macro has been removed. To generate the Metaview applet, use the JAVAMETA device.



PART 1

SAS/GRAPH Concepts

CHAPTER

1

Introduction to SAS/GRAPH Software

Overview

SAS/GRAPH is the data visualization and presentation (graphics) component of the SAS System. As such, SAS/GRAPH:

- 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 allowing 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, review, or send to a hard copy graphics output device such as a laser printer, plotter, or slide camera.
- provides utility procedures and statements to manage the output.

This chapter describes the graphs that are produced by SAS/GRAPH and explains some of the parts and features of SAS/GRAPH programs.

Components of SAS/GRAPH Software

There are several components to SAS/GRAPH software.

Device-based SAS/GRAPH procedures
: enable you to create a variety of graphs, including bar charts, pie charts, scatter plots, surface plots, contour plots, a variety of maps, and much more. The device-based SAS/GRAPH procedures include the GAREABAR, GCHART, GPLOT, GMAP, 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 as well as create output in a variety of formats such as PNG files and interactive ActiveX controls or Java applets. This document, *SAS/GRAPH: Reference*, describes the device-based SAS/GRAPH procedures and how to use devices. See also "Device-Based Graphics and Template-Based Graphics" on page 6.

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, GPLOT, GMAP, GBARLINE, GCONTOUR, and G3D, as well as others. For more information, see Chapter 29, "Using Annotate Data Sets," on page 641.

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, Web sites, 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. For additional information, see *SAS/GRAPH: Network Visualization Workshop User's Guide*.

SAS/GRAPH statistical graphics suite
: is part of ODS Statistical Graphics (referred to as ODS Graphics for short). ODS Graphics is functionality for creating statistical graphics that is available in a number of SAS software products, including SAS/STAT, SAS/ETS, SAS/QC, and SAS/GRAPH. The SAS/GRAPH statistical graphics suite provides the following features:

  SAS/GRAPH statistical graphics procedures
  : provide a simple syntax for creating graphics commonly used in exploratory data analysis and for creating customized statistical displays. These procedures include the SGPANEL, SGPLOT, and SGSCATTER procedures. In addition, the SGRENDER procedure provides a SAS procedure interface to create graphs using the Graph Template Language. These procedures are template-based procedures; they do not use devices like the device-based SAS/GRAPH procedures. For more information, see "Device-Based Graphics and Template-Based Graphics" on page 6 and *SAS/GRAPH: Statistical Graphics Procedures Guide*.

  Graph Template Language (GTL)
  : is the underlying language for the default templates that are provided by SAS for procedures that use ODS Statistical Graphics. You can use the GTL either to modify these templates or to create your own customized graphs. Templates written with the GTL are built with the TEMPLATE procedure. For more information about Graph Template Language, see *SAS/GRAPH: Graph Template Language User's Guide* and *SAS/GRAPH: Graph Template Language Reference*.

  ODS Graphics Editor
  : is an interactive editor that enables you to edit and enhance graphs that are produced by procedures that use ODS Graphics. You can use the ODS Graphics Editor to modify the existing elements of a graph such as titles and labels, or to add features such as text annotation for data points. The ODS Graphics Editor runs in Windows and UNIX operating environments only. For more information, see *SAS/GRAPH: ODS Graphics Editor User's Guide*.

  ODS Graphics Designer
  : provides a point-and-click interface for creating ODS Graphics. Using the ODS Graphics Designer does not require knowledge of ODS templates or the Graph Template Language. With the ODS Graphics Designer, you can easily create multi-cell graphs, classification panels, scatter plot matrices, and more. You can save your output as an image file or as an ODS Graphics Designer file (SGD file) that you can edit later. The ODS Graphics Designer

runs in Windows and UNIX operating environments only. For more information, see *SAS/GRAPH: ODS Graphics Designer User's Guide*.

For additional information on the ODS Statistical Graphics functionality, see *SAS Output Delivery System: User's Guide* and *SAS/STAT User's Guide*.

Device-Based Graphics and Template-Based Graphics

SAS/GRAPH produces graphics using two very distinct systems. SAS/GRAPH can produce output using a device-based system or using a template-based system. The traditional system for producing graphics output that most users are familiar with is the device-based system.

device-based graphics
: are SAS/GRAPH output that is generated by a default or user-specified device (DEVICE= option). Device drivers supplied by SAS are stored in the SAS/GRAPH catalog. Examples of device drivers are GIF, PNG, ACTIVEX, SVG, and SASPRTC. Most procedures that produce device-based graphics also produce GRSEG catalog entries in addition to any image files that are produced. Common 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, you can use the GOPTIONS statement to control the graphical environment.

template-based graphics
: are SAS/GRAPH output that is produced from a compiled ODS template of type STATGRAPH. Templates supplied by SAS are stored in SAS/GRAPH. Device drivers and most global statements (such as SYMBOL, PATTERN, AXIS, and LEGEND) have no effect on template-based graphics. The SAS/GRAPH procedures that produce template-based graphics are the SGPLOT, SGPANEL, SGSCATTER, and SGRENDER procedures. Many SAS/STAT, SAS/ETS, and SAS/QC procedures also produce template-based graphics when you specify the ODS GRAPHICS ON statement. (Template-based graphics are frequently referred to as *ODS graphics*.) Template-based graphics are always produced as image files and never as GRSEG catalog entries. For template-based graphics, you must use the ODS GRAPHICS statement to control the graphical environment.

The *SAS/GRAPH: Reference* contains information about device-based graphics only. For information about template-based graphics, see *SAS/GRAPH: Statistical Graphics Procedures Guide* and *SAS/GRAPH: Graph Template Language Reference*.

Graph Types

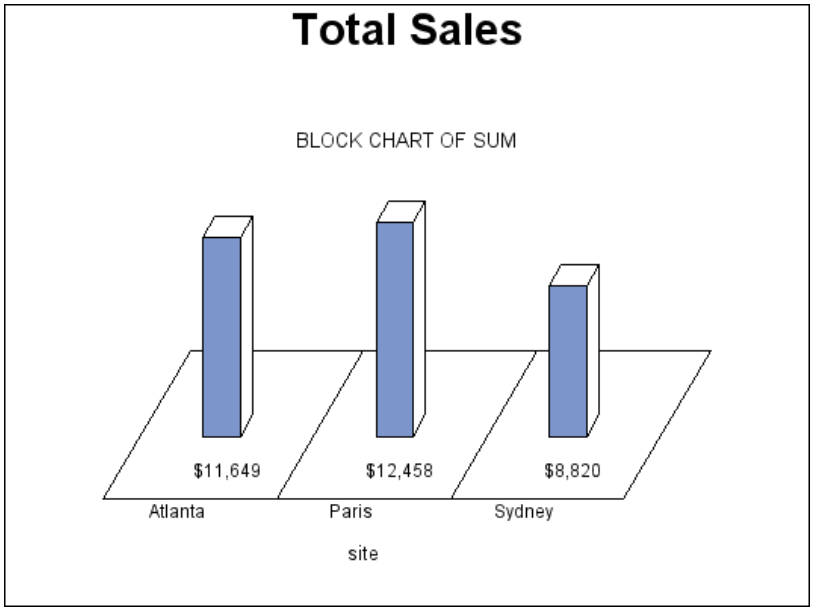
SAS/GRAPH produces many kinds of charts, plots, and maps in both two- and three-dimensional versions. In addition to helping you understand the variety of graphs that are available to you, these descriptions also help you choose the correct type of graph for your data and point you to the appropriate chapter.

Charts

SAS/GRAPH uses the GCHART procedure to produce charts that graphically represent the value of a statistic for one or more variables in a SAS data set. See Chapter 36, “The GCHART Procedure,” on page 989 for a complete description.

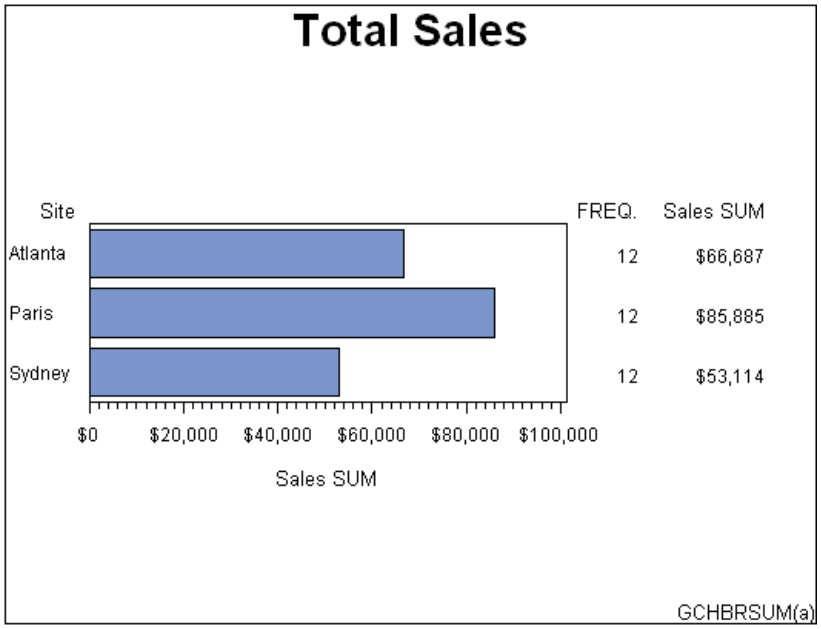
Block charts

Block charts use three-dimensional blocks to graphically represent values of statistics. Block charts are useful for emphasizing relative magnitudes and differences among data values.



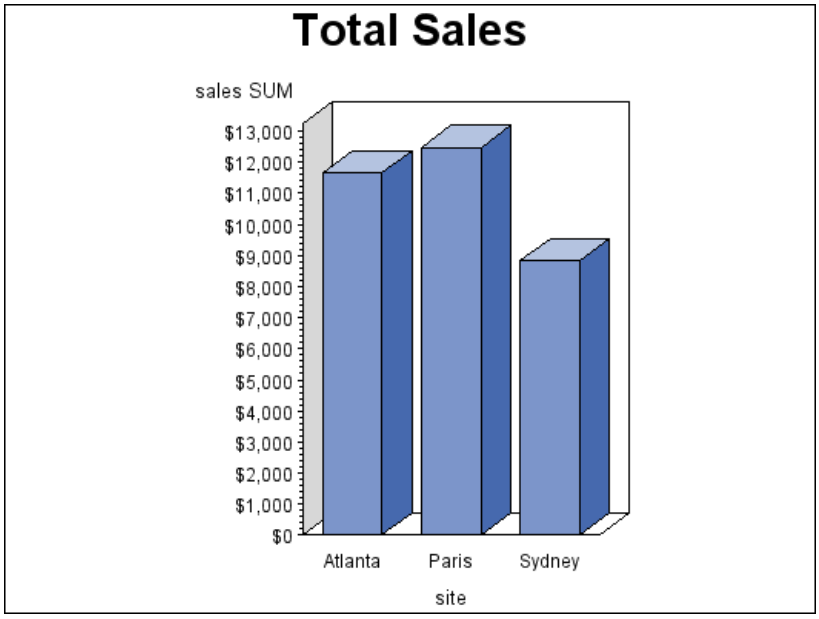
Horizontal bar charts

Horizontal bar charts use horizontal bars to represent statistics based on the values of one or more variables. Horizontal bar charts can generate a table of chart statistics and are useful for displaying exact magnitudes and emphasizing differences.



Vertical bar charts

Vertical bar charts use vertical bars to represent statistics based on the values of one or more variables. Vertical bar charts, which generate only one statistic, are useful for displaying exact magnitudes and emphasizing differences.



Pie charts, Detailed pie charts, 3D pie charts, and Donut charts

Pie charts, detailed pie charts, 3-D pie charts, and Donut charts use the angle of pie slices to graphically represent the value of a statistic for a data range. Pie charts are useful for examining how the values of a variable contribute to the whole and for comparing the values of several variables.

Figure 1.1 Detailed Pie Chart

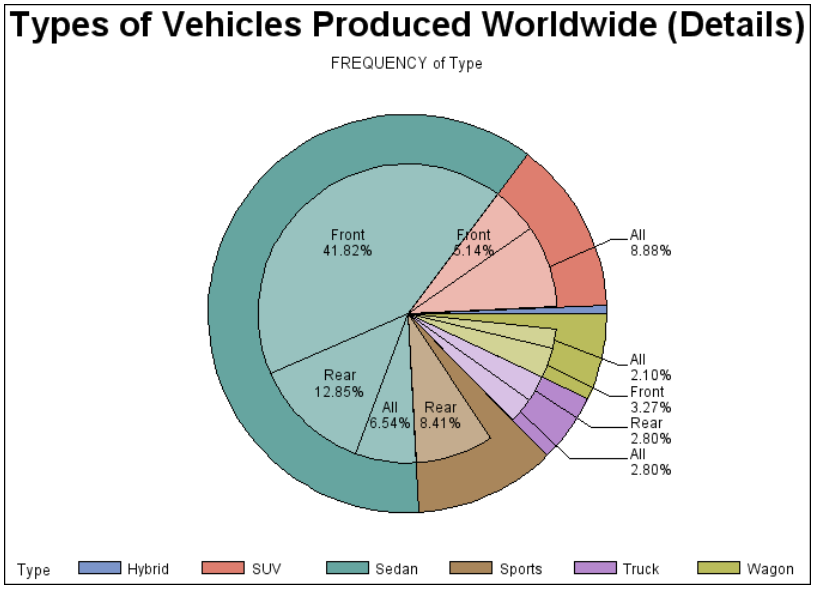
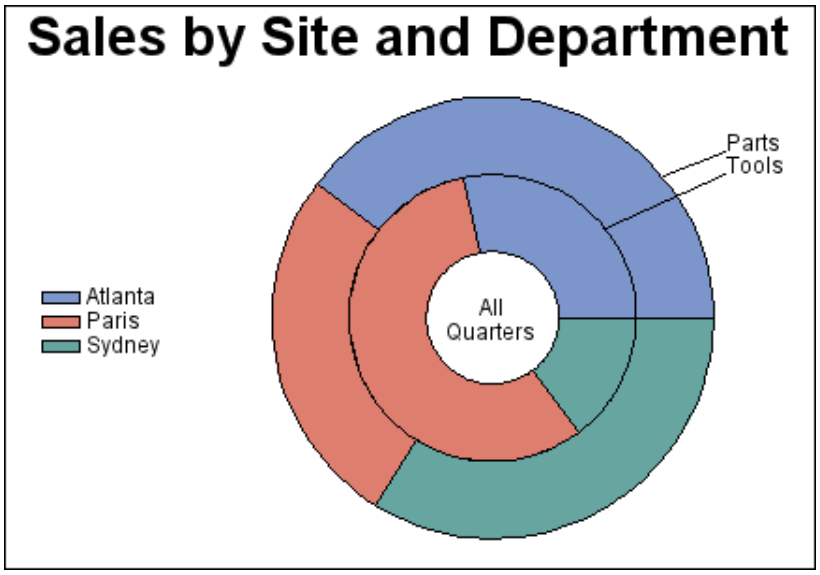
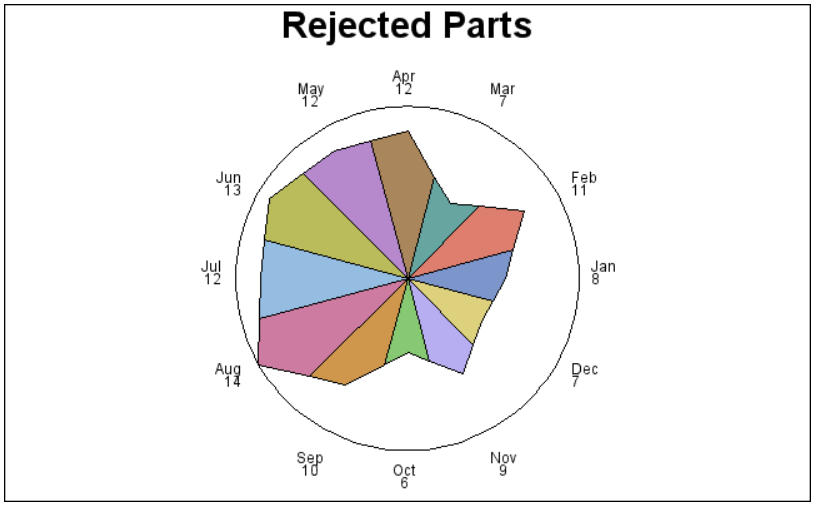


Figure 1.2 Donut Chart



Star charts

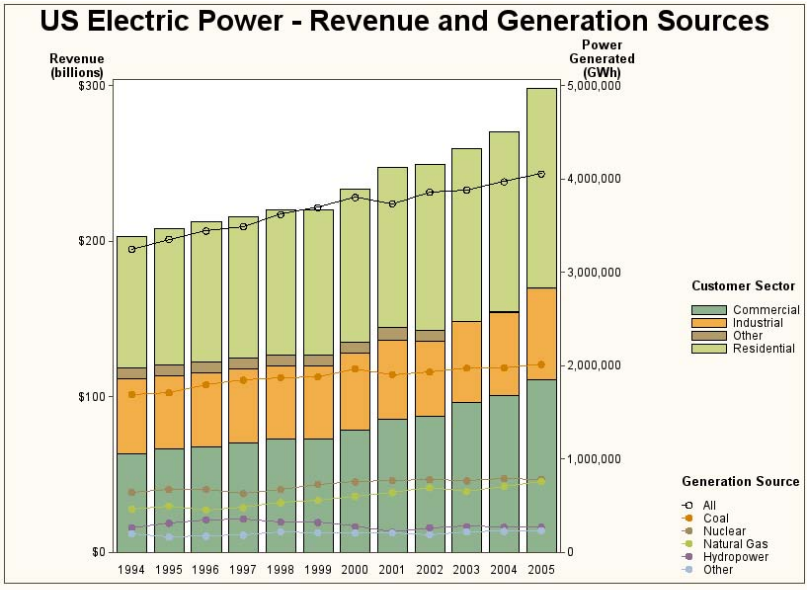
Star charts use the length of spines to graphically represent the value of a statistic for a data range. Star charts are useful for analyzing where data are out of balance.



Bar-line Charts

The GBARLINE procedure produces vertical bar charts with 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.

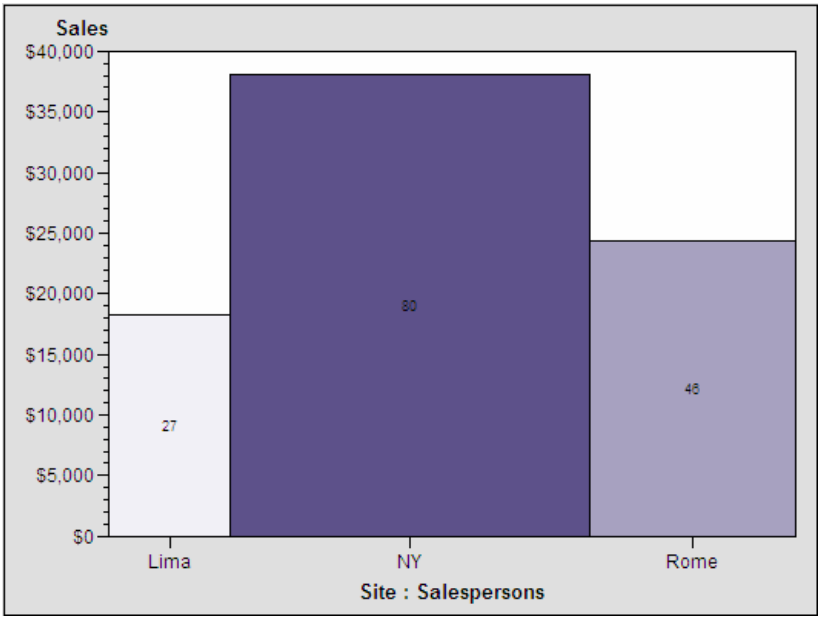
See Chapter 35, “The GBARLINE Procedure,” on page 947 for a complete description.



Link to Bar Data: USEIA Energy Customer Sectors
Link to Line Data: USEIA Energy Generation Sources

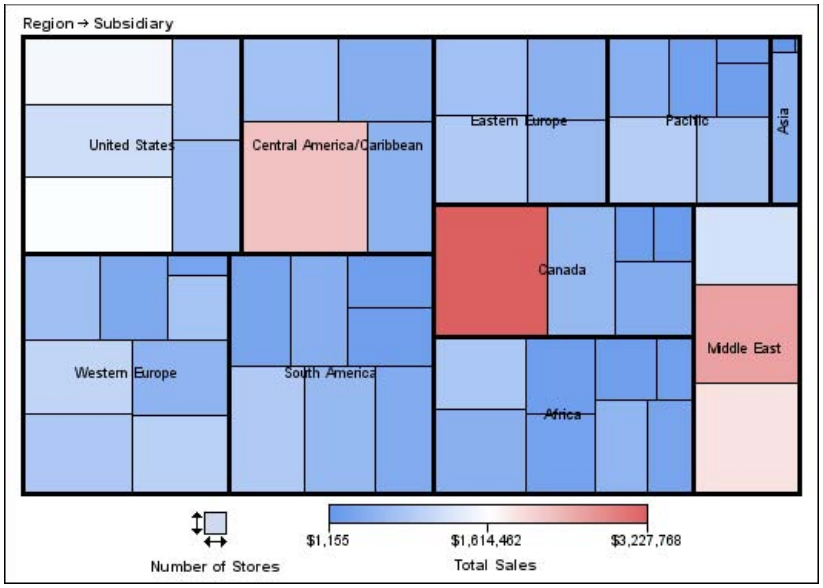
Area Bar Charts

The GAREABAR procedure produces area bar charts that show the magnitudes of *two* variables for each category of data. For example, the following area bar chart shows the sales total for each of three geographical sites. The width of each bar indicates the number of sales persons at each site. In a bar chart such as the chart shown in "Vertical bar charts" on page 8, the width is the same for each bar. In an area bar chart, the width and height of each bar is determined by the value of variables. See Chapter 34, "The GAREABAR Procedure," on page 931 for a complete description.



Tile Charts

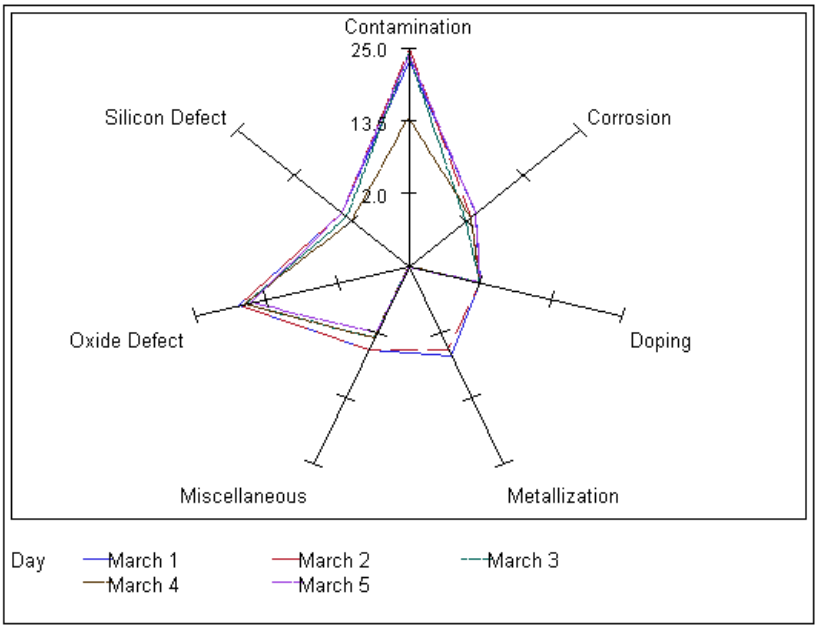
The GTILE procedure produces charts that tile charts, which consist of a rectangle or square divided into tiles. The sizes of the individual tiles represent the value of the size variable. You can also specify a color variable, so that the colors of the individual tiles represent the magnitude of the color variable. Tile charts are useful for determining the relative magnitude of categories of data or the contribution of a category toward the whole.



Radar Charts

The GRADAR procedure produces radar charts that show the relative frequency of data measures. On a radar chart, 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, thus giving them the look of a radar screen. Radar charts are frequently called star charts and are often used in quality control or market research problems.

See Chapter 47, “The GRADAR Procedure,” on page 1419 for a complete description.



Two-Dimensional Plots

SAS/GRAPH uses the GPLOT procedure to produce two-dimensional graphs that plot one or more dependent variables against an independent variable within a set of coordinate axes. GPLOT can display the data points as individual symbols (as in a scatter plot), or use interpolation methods specified by the SYMBOL statement to join the points, request spline interpolation or regression analysis, produce various high-low plots, or generate several other types of plots.

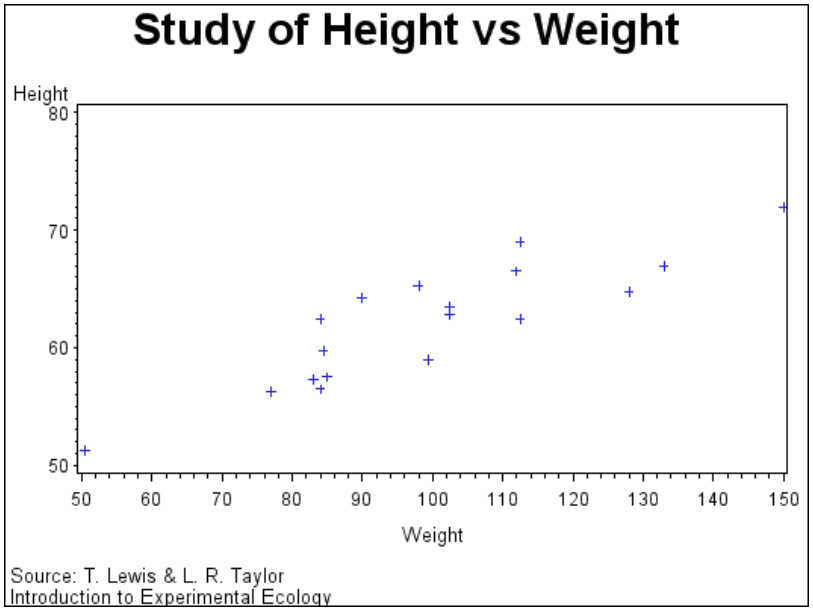
GPLOT can also display data as bubble plots in which circles of different sizes represent the values of a third variable.

Plots are useful for demonstrating the relationship between two or more variables and frequently compare trends or data values or depict movements of data values over time.

See Chapter 45, “The GPLOT Procedure,” on page 1325 for a complete description.

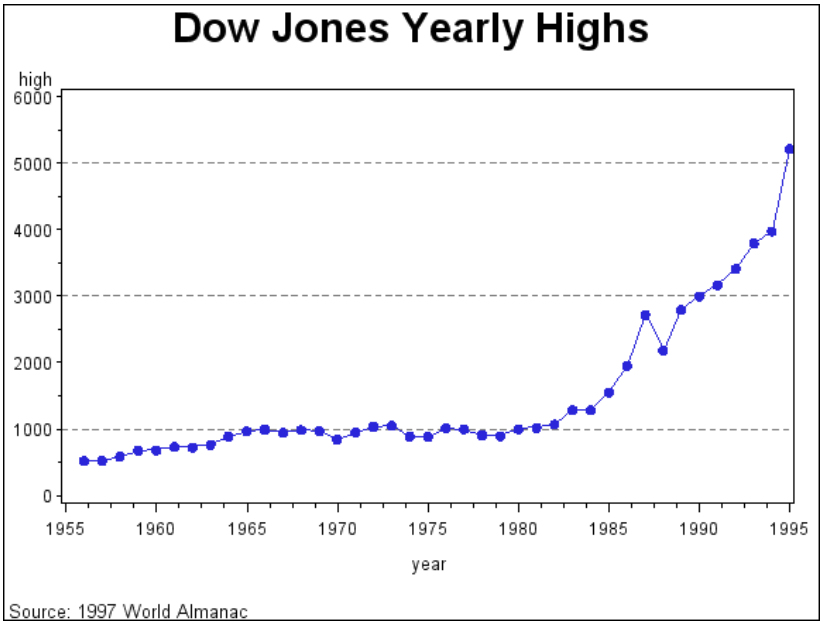
Two-dimensional scatter plots

Two-dimensional scatter plots show the relationship of one variable to another, often revealing concentrations or trends in the data. Typically, each variable value on the horizontal axis can have any number of corresponding values on the vertical axis.



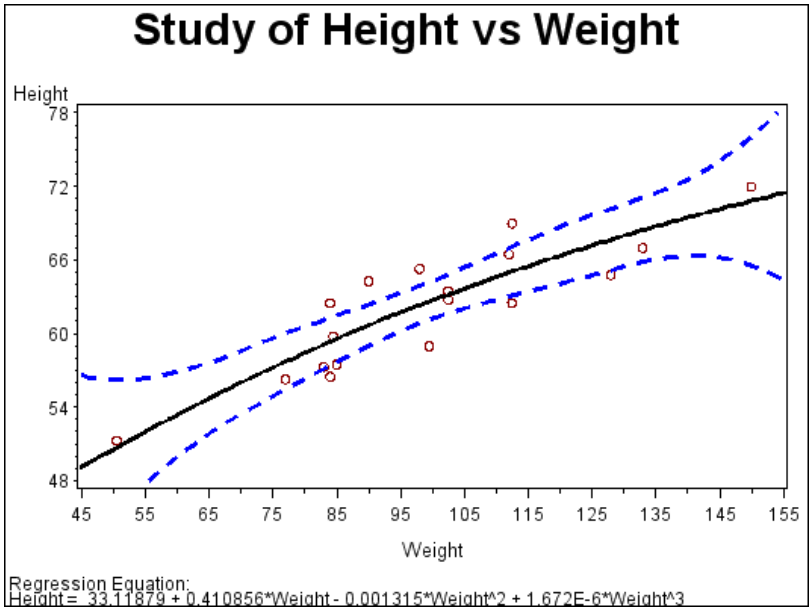
Simple line plots

Simple line plots show 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. The line connecting data points can be smoothed using a variety of interpolation methods, including the Lagrange and the cubic spline interpolation methods.



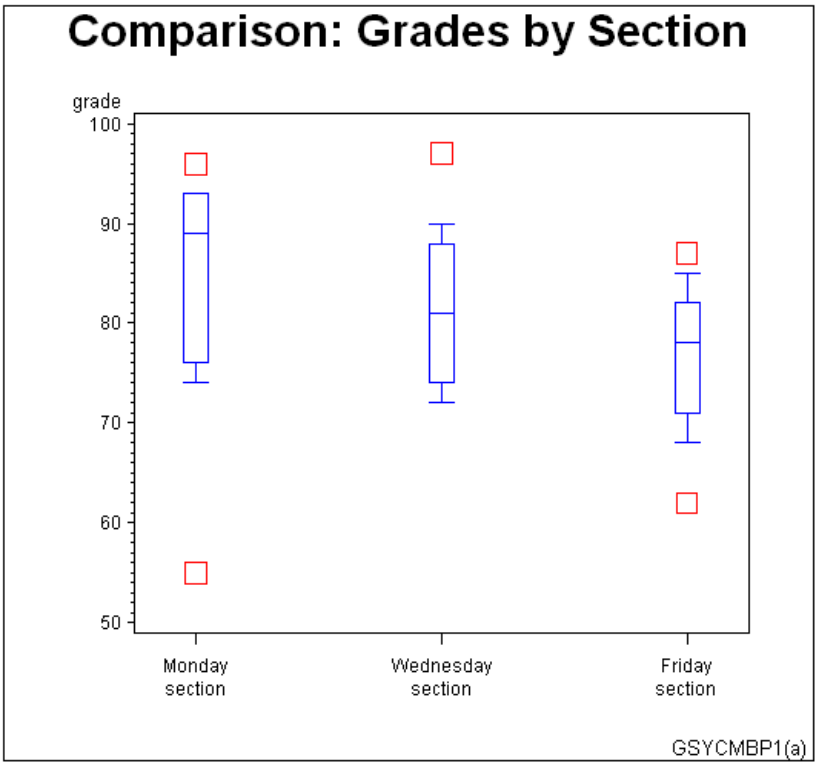
Regression plots

Regression plots specify that the plot is a regression analysis. You can specify one of three types of regression equation – linear, quadratic, or cubic, and you can choose to display confidence limits for mean predicted values or individual predicted values.



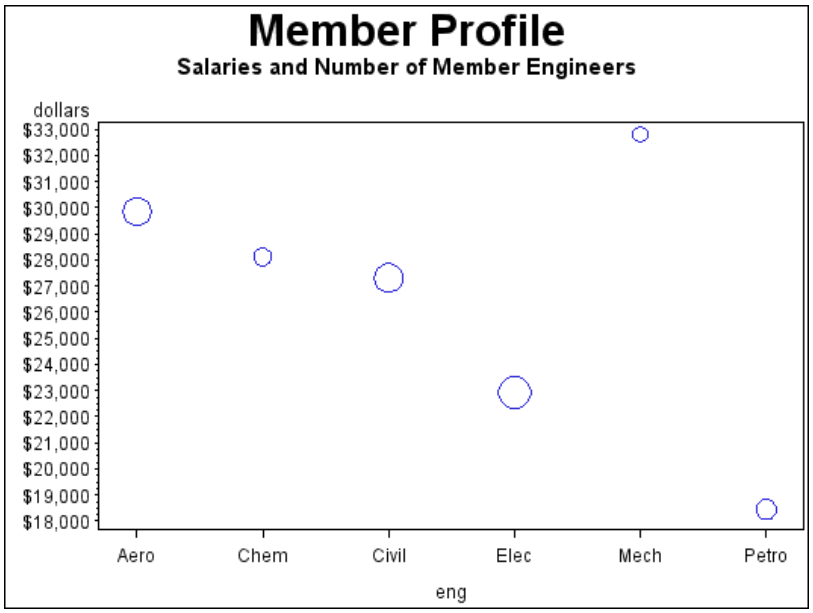
High-low plots

High-low plots show how several values of one variable relate to one value of another variable. Typically, each variable value on the horizontal axis has several corresponding values on the vertical axis. High-low plots include box, needle, and stock market plots.



Bubble plots

Bubble plots show the relative magnitude of one variable in relation to two other variables. The values of two variables determine the position of the bubble on the plot, and the value of a third variable determines the size of the bubble.



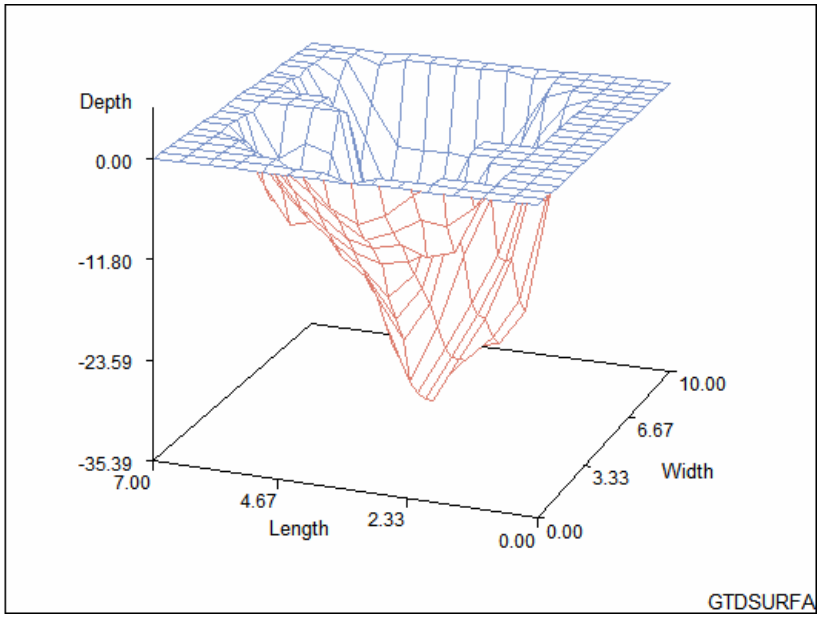
Three-Dimensional Plots

SAS/GRAPH uses the G3D procedure to produce three-dimensional surface and scatter plots that examine the relationship among three variables. Variable values are plotted on a set of three coordinate axes.

See Chapter 53, “The G3D Procedure,” on page 1541 for a complete description.

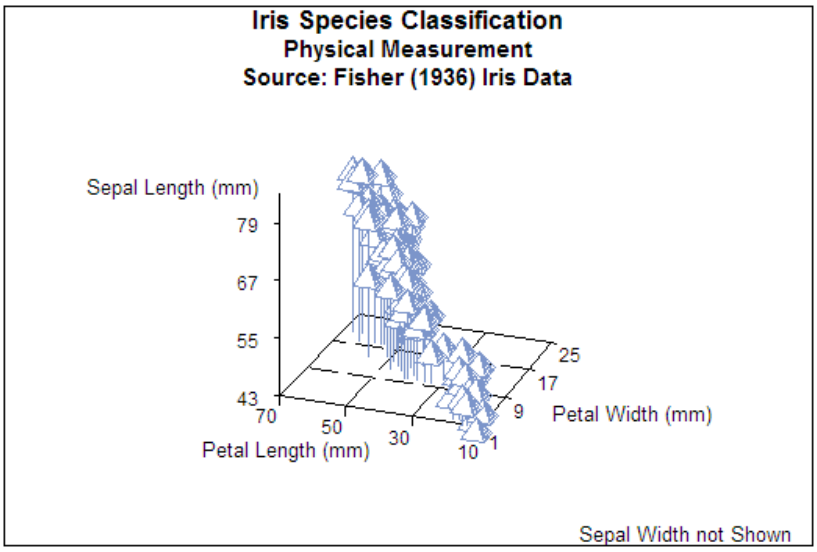
Surface plots

Surface plots are three-dimensional plots that display the relationship of three variables as a continuous surface. Surface plots examine the three-dimensional shape of data.



Scatter plots

Scatter plots enable you to examine three-dimensional data points instead of surfaces and to classify your data using size, color, shape, or a combination of these features.

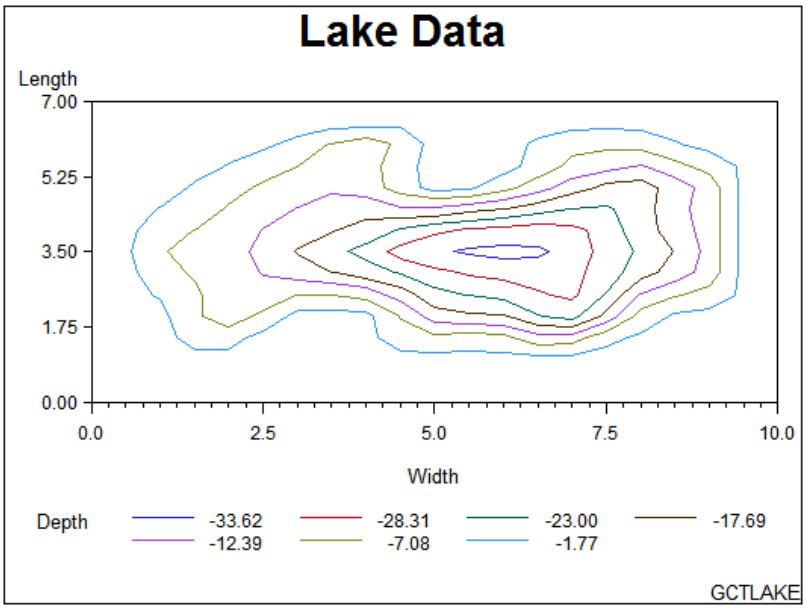


Contour plots

SAS/GRAPH uses the GCONTOUR procedure to examine three-dimensional data in two dimensions. Lines or areas in a contour plot represent levels of magnitude (z) corresponding to a position on a plane (x,y).

See Chapter 37, "The GCONTOUR Procedure," on page 1095 for a complete description.

Contour plots are two-dimensional plots that show three-dimensional relationships. These plots use contour lines or patterns to represent levels of magnitude of a contour variable plotted on the horizontal and vertical axes.



When you need to interpolate or smooth data values that are used by the G3D and GCONTOUR procedures, use the G3GRID procedure. The G3GRID procedure does not produce graphics output but processes existing data sets to create data sets that the G3D or GCONTOUR procedure can use to produce three-dimensional surface or contour plots. See Chapter 54, “The G3GRID Procedure,” on page 1571 for a complete description.

Maps

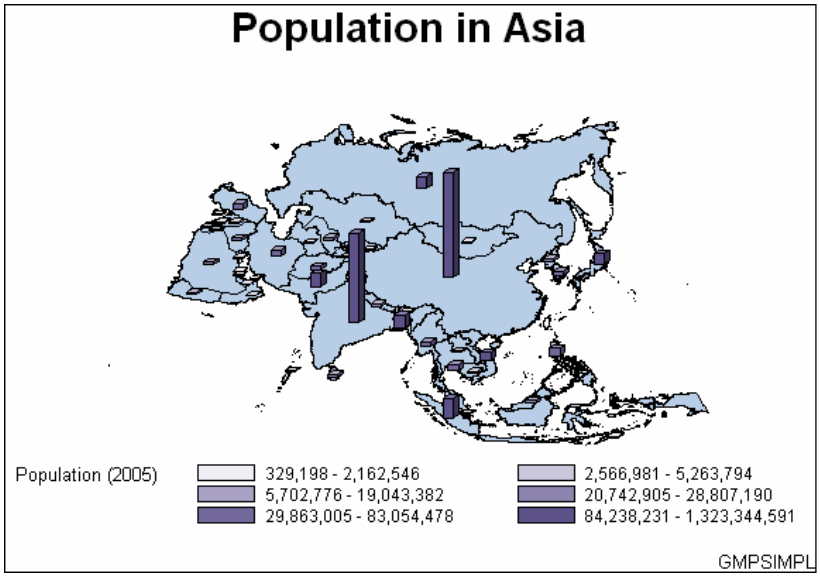
SAS/GRAPH uses the GMAP procedure to produce two- and three-dimensional maps that can show an area or represent values of response variables for subareas.

SAS/GRAPH includes data sets to produce geographic maps. In addition, you can create your own map data sets.

See Chapter 43, “The GMAP Procedure,” on page 1239 for a complete description.

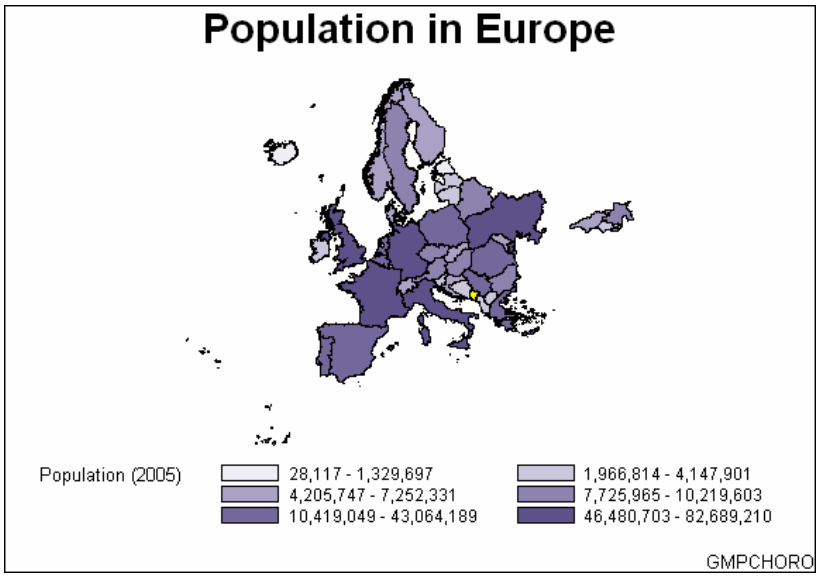
Block maps

Block maps are three-dimensional maps that represent data values as blocks of varying height rising from the middle of the map areas.



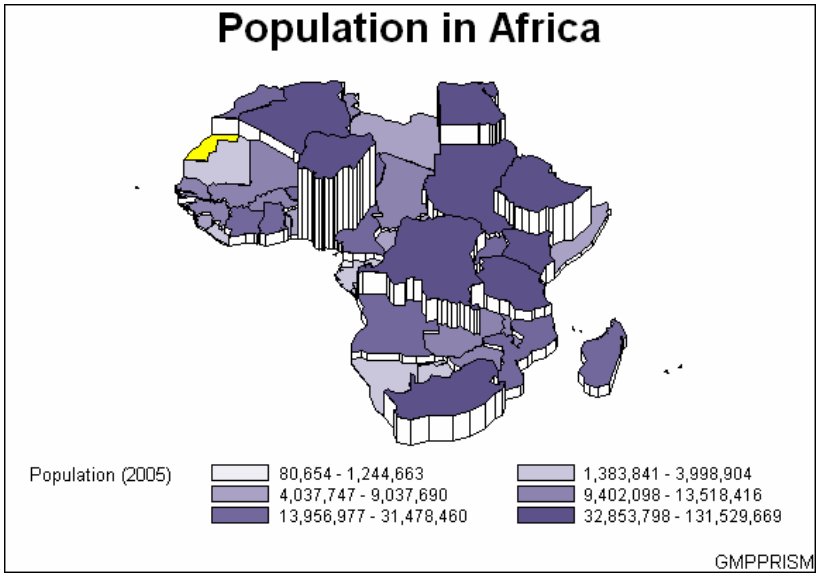
Choropleth maps

Choropleth maps are two-dimensional maps that display data values by filling map areas with combinations of patterns and color that represent the data values.



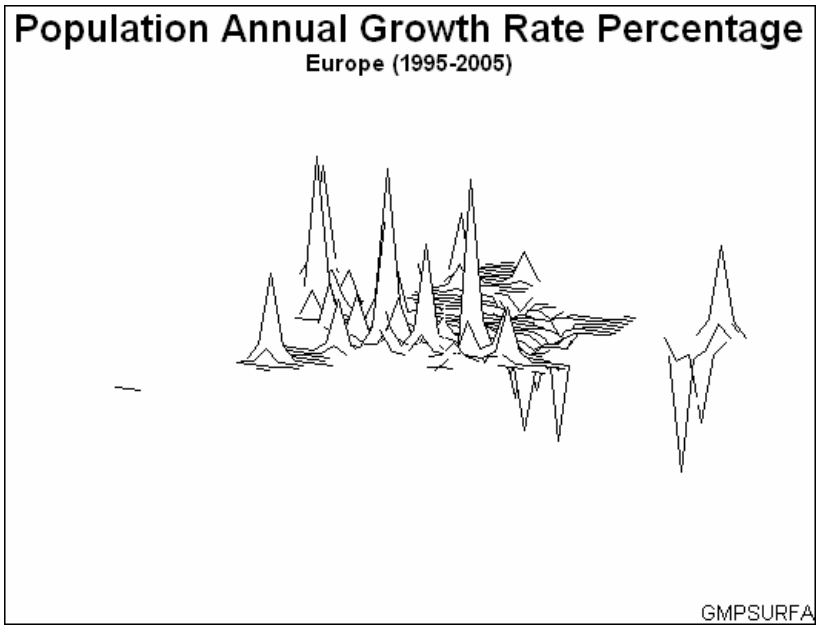
Prism maps

Prism maps are three-dimensional maps that display data by raising the map areas and filling them with combinations of patterns and colors.



Surface maps

Surface maps are three-dimensional maps that represent data values as spikes of varying heights.



SAS/GRAPH also provides several utility procedures for handling map data.

The GPROJECT procedure lets you choose how geographic maps are projected. This is particularly important for large areas because producing a map of any large area on the Earth involves distorting some areas in the process of projecting the spherical surface of the Earth onto a flat plane. You can use the procedure to select the projection method that least distorts your map.

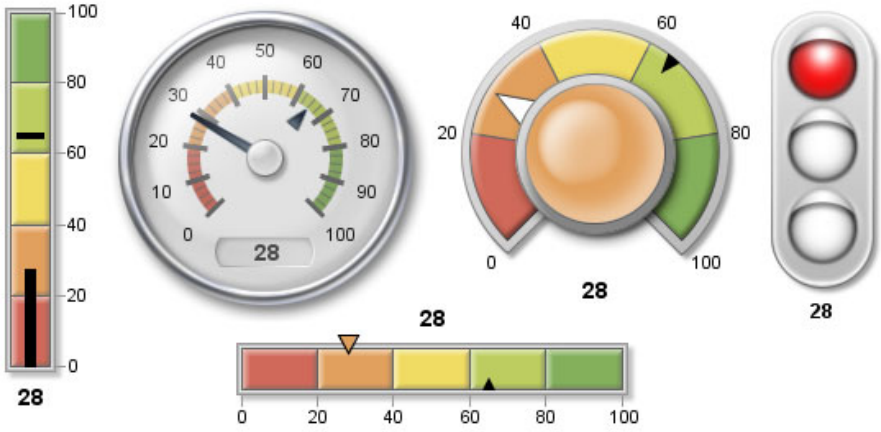
Map areas are constructed of joined data points. Each data point represents an observation in a SAS data set. For large maps, the amount of data can be prohibitively expensive (in terms of computing resources or time to process); the GREDUCE

procedure enables you to reduce the number of points in the data set. The GREMOVE procedure enables you to remove boundary lines within a map.

KPI Charts

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)
- radial dial
- speedometer
- traffic light (vertical or horizontal).



Creating Text Slide and Presentation Graphics

You can use SAS/GRAPH to create slide presentations of your graphs. With SAS/GRAPH you can

- create text slides with the GSLIDE procedure
- combine several graphs into one output with the GREPLAY procedure
- automatically or manually replay your graphs and text slides with the GREPLAY procedure.

Text Slides

Use the GSLIDE procedure to create text slides in which you can specify a variety of colors, fonts, sizes, angles, overlays, and other modifications as well as drawing lines and boxes on the output.

See Chapter 51, "The GSLIDE Procedure," on page 1517 for a complete description.

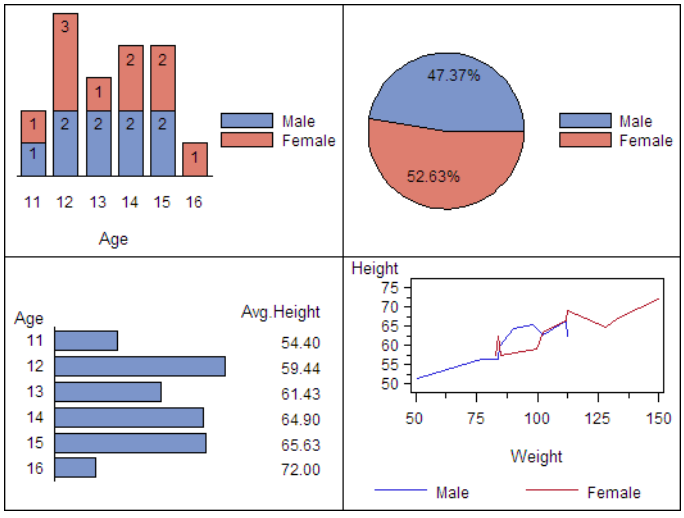
Text slides display text as graphics output. Text slides can be used as title slides for presentations, or to produce certificates, signs, or other display text.

Combining Output into One Slide

Use the GREPLAY procedure to combine several graphs into a single output. You can create special effects by overlaying or rotating the graphs at any angle.

Templated graphs display two or more graphs or text slides as one output by replaying stored graphs into a template or framework. Like graphs and text slides, templated graphs can be ordered in groups and stored in catalogs for replay as part of a presentation.

Figure 1.3 Templated graphs



In addition, you can use the GREPLAY procedure to create an automated or user-controlled presentation of graphics output. The GREPLAY procedure enables you to name, arrange, and customize the presentation of graphs that are stored in a catalog.

See Chapter 50, “The GREPLAY Procedure,” on page 1473 for a complete description.

Enhancing Graphics Output (graphs and text slides)

SAS/GRAPH Statements

You can also use *global statements* and *graphics options* in SAS/GRAPH programs. With global statements, you can add titles and footnotes and control the appearance of axes, symbols, patterns, and legends. With graphics options, you can control the appearance of graphics elements by specifying default colors, fill patterns, fonts, text height, and so on.

The Annotate Facility

The Annotate facility enables you to program graphics by using certain variables in SAS data sets. It is often used to add text or special elements to the graphics output of other procedures, although it can also be used to construct custom graphics output. Text and graphics can be placed at coordinates derived from input data, as well as coordinates expressed as explicit locations on the display.

Figure 1.4 Annotated graphs

Creating Custom Graphics

The Annotate facility can also be used to generate custom graphics without using any of the SAS/GRAPH graphing procedures.

The DATA Step Graphics Interface

The DATA Step Graphics Interface provides functions and calls that produce graphics output from the DATA step, rather than from a procedure. The functions and calls are similar in form to those specified by the ISO Graphic Kernel Standard (GKS); however, the interface is not an implementation of the GKS. The form is similar enough that many GKS-compliant programs can be converted easily to run as SAS/GRAPH programs.

Graph-N-Go

To generate presentation graphs without writing any SAS/GRAPH code, you can use Graph-N-Go (not available on mainframes). You can start Graph-N-Go in several ways:

- from the menus in any SAS window, select **Solutions ▸ Reporting ▸ Graph-N-Go**
- submit either of the following from the SAS command line:

```
gng

graphngo
```

- use an Explorer window to directly open a GFORM entry. Double-click (or right-click and choose Open) on a GFORM entry to start a Graph-N-Go session using that entry.

Information on using the application is in Graph–N-Go help, which you can access from the application's main window in either of two ways:

- select **Help ▸ Using This Window**
- press F1 (this might not work in some operating environments).

You can also get help for the application by submitting the following command from the SAS command line:

```
help gng
```

About this Document

This document provides reference information for all facilities, procedures, statements, and options that can be used with SAS/GRAPH. This chapter describes what you need to know to use SAS/GRAPH, and what conventions are used in text and example code. To gain full benefit from using this document, you should familiarize yourself with the information presented in this chapter, and refer to it as needed.

Audience

This document is written for users who are experienced in using the SAS System. You should understand the concepts of programming in the SAS language, and you should have an idea of the tasks you want to perform with SAS/GRAPH.

Prerequisites

The following table summarizes the SAS System concepts that you need to understand in order to use SAS/GRAPH:

To learn how to	Refer to
invoke the SAS System 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 the SAS Text Editor to enter and edit text	*SAS Language Reference: Concepts* or *SAS Language Reference: Dictionary*

To learn how to	Refer to
allocate SAS libraries and assign librefs create external files and assign filerefs	documentation for using the SAS System under the operating system for the hardware at your site
manipulate SAS data sets using SAS procedures	*Base SAS Procedures Guide*

Conventions Used in This Document

This section explains the conventions this document uses for text, SAS language syntax, and file and library references. The document uses the following terms in discussing syntax:

keyword — is a literal that is a primary part of the SAS language. (A literal must be spelled exactly as shown, although it can be entered in uppercase or lowercase letters.) Keywords in this document are procedure names, statement names, macro names, routine names, and function names.

argument — is an element that follows a keyword. It is either literal, or it is user-supplied. It has a built-in value (for example, NODISPLAY), or it has a value assigned to it (for example, COLOR=*text-color*).

Arguments that you must use are *required arguments*. Other arguments are *optional arguments*, or simply *options*.

value — is an element that follows an equal sign. It assigns a value to an argument. It might be a literal, or it might be a user-supplied value.

parameter — is a value assigned to an argument that itself takes a value, for example, the COLOR= parameter of the LABEL= option in a LEGEND statement, as shown in the following statement:

```
legend label=(color=blue);
```

Syntax Conventions

Type styles have special meanings when used in the presentation of SAS/GRAPH syntax in this document. The following list explains the style conventions for the syntax sections:

UPPERCASE — identifies SAS keywords such as the names of statements and procedures (for example, PROC GCHART). Uppercase characters also identify arguments and values that are literals (for example, NOLEGEND and LABEL=NONE).

italic — identifies arguments or values that you supply. Items in italic can represent user-supplied values that are either

- nonliteral values assigned to an argument (for example, *axis-color* in COLOR=*axis-color*)
- nonliteral arguments (for example, VBAR *chart-variable*. . . ;).

In addition, an item in italics can be the generic name for a list of arguments or parameters from which the user can choose (for example, *appearance-options*).

The following symbols are used to indicate other syntax conventions:

< > (angle brackets)	identify optional arguments. Any argument not enclosed in angle brackets is required.
| (vertical bar)	indicates that you can choose one value from a group. Values separated by bars are mutually exclusive.
. . . (ellipsis)	indicates that the argument following the ellipsis can be repeated any number of times (*plot-request* <. . . *plot-request-n*>, for example). If the ellipsis and the following argument are enclosed in angle brackets, they are optional. In SAS/GRAPH, an ellipsis also indicates a range from which a value is selected (LINE=1 . . . 46, for example).

The following examples illustrate the syntax conventions described in this section. These examples contain selected syntax elements, not complete syntax.

PROC GANNO ANNOTATE=*Annotate-data-set*
<DATASYS>;

- PROC GANNO is in uppercase because it is a SAS keyword, the name of a statement. The remaining elements are arguments for the statement.
- ANNOTATE= is not enclosed in angle brackets because it is a required argument. It is in uppercase to indicate that it is a literal and must be spelled as shown.
- *Annotate-data-set* is in italic because it is a value that you must supply; in this case, the value must be a data set name.
- DATASYS is enclosed in angle brackets because it is an optional argument. It is in uppercase to indicate that it is a literal and must be spelled as shown.
- The ending semicolon (;) is required because it is outside the angle brackets for the option.

SYMBOL <1 . . . 99>
<COLOR=*symbol-color*>
<MODE=EXCLUDE|INCLUDE>
<*appearance-options*>;

- SYMBOL is in uppercase because it is a SAS keyword, the name of a statement. The numbers 1 . . . 99 are in angle brackets because they are optional. The ellipsis indicates that you choose one from the range of numbers 1 through 99. The remaining elements are arguments for the statement.
- COLOR= is enclosed in angle brackets because it is an optional argument.
- *Symbol-color* is in italics because it represents a value that you specify.
- MODE= is enclosed in angle brackets because it is an optional argument.
- EXCLUDE and INCLUDE are in uppercase because they are literal values and must be spelled exactly as shown. They are separated by a vertical bar (an OR bar) because you use one or the other but not both.
- *Appearance-options* is in italics because it is a generic name for a list of options that can be used in the SYMBOL statement.

HBAR *chart-variable*< . . . *chart-variable-n*>
</ <PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP>
<*statistic-options*>>;

- *Chart-variable* is italic because it is an argument that you supply. It is required because it is not in angle brackets.
- *Chart-variable-n* is enclosed in angle brackets because additional user-supplied arguments are optional. The ellipsis before the argument indicates that it can be repeated as many times as desired.
- PATTERNID= is a literal option. The values BY, GROUP, MIDPOINT, and SUBGROUP are literal values that are mutually exclusive. You can use only one, and it must be spelled as shown.
- *Statistic-options* is in italics because it is the generic name of a list of options that affect the chart statistics.

When you are using an option, a statement, or a procedure whose syntax shows arguments or values in italics, you must supply the argument or value. When the argument or value is a font, color, or variable name, SAS/GRAPH expects valid font names, color names, and variable names. Consider the following four syntax samples:

FONT=*font*

COLOR=*color*

COLOR=*text-color*

PIE *chart-variable* < . . . *chart-variable-n*>;

- *Font* must be a valid SAS font name. (See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 for details.)
- *Color* and *text-color* must be valid SAS/GRAPH colors. (See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 for details.)
- *Chart-variable* must be a valid SAS variable name. (See *SAS Language Reference: Dictionary* for details.)

Conventions for Examples and Output

Most of the chapters in this document include examples that illustrate some of the features of a procedure or its statements. Each example contains

- a description of the highlights of the example
- the program statements that produce the output
- the actual output from the example
- an explanation of the features of the example.

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 Listing 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 you see. Most of the images of output in this document were generated with a GOPTIONS

statement that specified a size approximately equal 5.5 inches by 4.2 inches, although some images might be larger, if necessary, to accommodate the content of the graph.

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

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

Most examples specify these options:

RESET=ALL	sets all graphics options to default values and cancels all global statements.
BORDER	draws a border around the graphics output area.

Information You Should Know

This section outlines information you should know before you attempt to run the examples in this document.

Support Personnel

Most sites have personnel available to help users learn to run SAS System. Record the name of your on-site SAS support personnel. Also, record the names of anyone else you regularly turn to for help with running 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 the OnlineDoc and paste it into the Program Editor in your SAS session.

These same programs are included in the sample library 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. If the SAS Sample Library has been installed at your site, ask your on-site SAS support personnel where it is located.

To access the sample programs through SAS Help and Documentation or through your file system, you must understand the naming convention used for the samples. 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." For example, the code for the first example in the GMAP procedure chapter, Example 1 on page 1301, is stored in sample member GMPSIMPL. The sample-library member name is sometimes displayed as a footnote in the output's lower-right corner.

- In the Help system, 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 the first example in the GMAP procedure chapter, select **Learning to Use SAS ► SAS/GRAPH ► Samples**, scroll to **`GMAP Procedure`**, and select **`GMPSIMPL-Producing a Simple Block Map`**.

- In your file system, the files that contain the sample code have filenames that match the sample member names. For example, in a directory-based system, the code for sample member GMPSIMPL is located in a file named GMPSIMPL.SAS.

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

Table 1.1 Product Codes for SAS/GRAPH Procedures

Procedure	Code
dsgi	DS
ganno	AN
gareabar	AB
gbarline	BL
gchart	CH
gcontour	CT
geocode	GE
gfont	FO
ginside	IN
gkpi	KP
gmap	MP
goptions	OP
gplot	PL
gproject	PJ
gradar	RR
greduce	RD
gremove	RM
greplay	RE
gslide	SL
gtile	TL
g3d	TD
g3grid	TG

Table 1.2 Product Codes for SAS/GRAPH Statements

Statement	Code
axis	AX
by	BY
footnote	FO

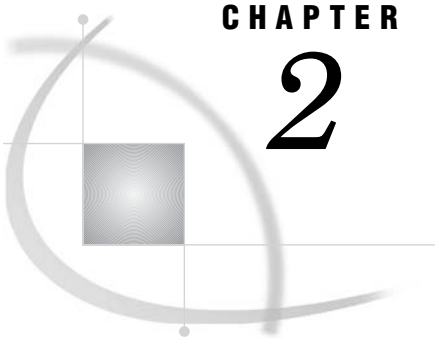
Statement	Code
goptions	ON
legend	LG
note	NO
pattern	PN
symbol	SY
title	TI

Map Data Sets

To run the examples that draw maps, you need to know where the map data sets are stored on your system. Depending on your installation, the map data set might automatically be assigned a libref. Ask your on-site SAS support personnel or system administrator where the map data sets are stored for your site.

Annotate Macros Data Set

To run the examples using Annotate macros, you need to know where the Annotate macro data set 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.



CHAPTER

2

Elements of a SAS/GRAPH Program

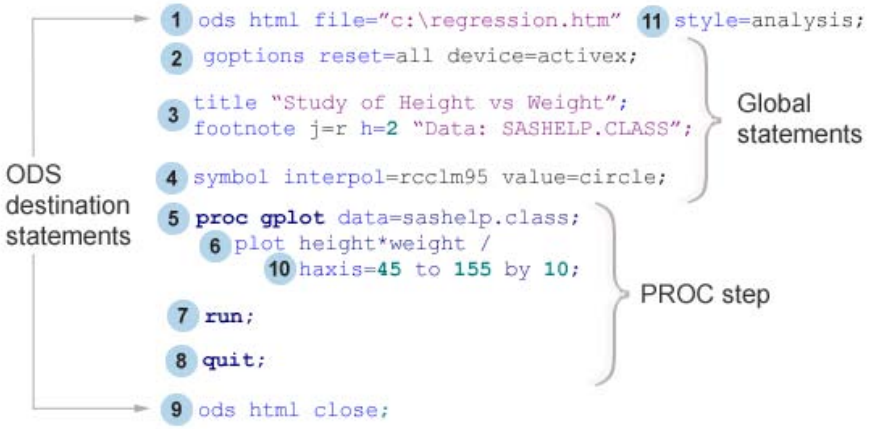
Overview

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, to define its basic parts, and to show 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 and DSGI functions 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.

Display 2.1 Typical SAS/GRAPH Program



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 and (ending) RUN statements of a procedure. These statements can identify and analyze the data in SAS data sets, generate the graphics output, control the appearance of the output, 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, but 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, GCHART, GMAP, 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. For example, the SUBGROUP= option in the BLOCK statement of the GCHART procedure 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 Display 2.1 on page 32, 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. Global statements set values and attributes for all the output created from that point in the program when the statement is specified. The specifications in a global statement are not confined to the output generated by any one procedure but apply to all the output generated then point on 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 a brief description of each. See Chapter 14, "SAS/GRAPH Statements," on page 197 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 withSAS/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.

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.

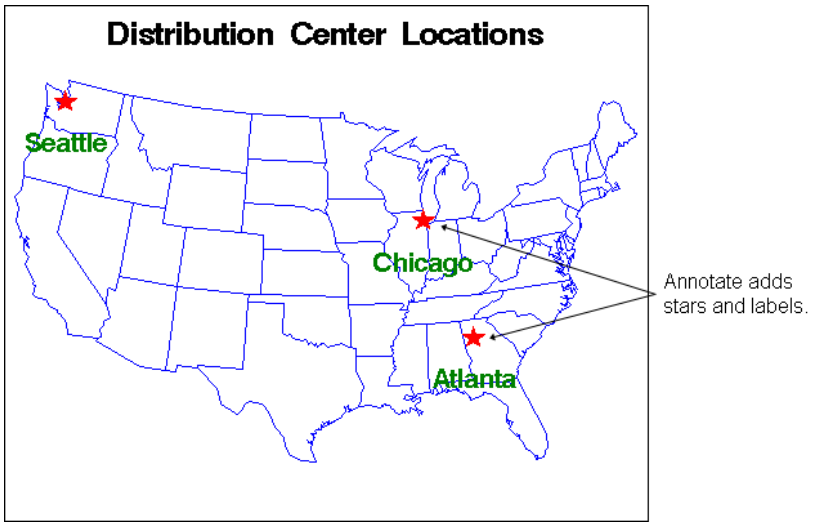
TITLE, NOTE, and FOOTNOTE 3
: add text to maps, plots, charts, and text slides. They control the content, appearance, and placement of text on your graph. The FOOTNOTE statement is used to display lines of text at the bottom of the page. The TITLE statement is used to specify up to ten title lines to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

  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 29, “Using Annotate Data Sets,” on page 641 for information on 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 Chapter 30, “Annotate Dictionary,” on page 667 for a complete description of all Annotate functions and variables. Below is an example of how the Annotate facility can be used to add text labels and symbols to a graph that was created using the GMAP procedure.

Display 2.2 Using Annotate with GMAP Procedure Output



DSGI Functions and Routines in a DATA Step

The DATA Step Graphics Interface (DSGI) enables you to create graphics output within the DATA step or from within an SCL application. Through DSGI, you can call the graphics routines used by SAS/GRAPH to generate a custom graph, or to rescale and replay existing graphs into viewports. DSGI GASK routines can be used to query current system or graphics area settings. For more information on DSGI functions and routines, see Chapter 31, “The DATA Step Graphics Interface,” on page 769.

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. While 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 listing destination. Some other destinations include HTML, RTF, and PDF. For more information about ODS destinations, see “Understanding ODS Destinations” in *SAS Output Delivery System: User’s Guide*.

As shown in Display 2.1 on page 32, the ODS destination statement is used at the beginning and end of the program to open and close the destination, respectively. If you do chose 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. To conserve system resources, you should also use the ODS destination statement to close the LISTING destination if you do not need LISTING output.

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. For more information about the STYLE= option, see Chapter 10, "Controlling The Appearance of Your Graphs," on page 133.

Base SAS Language Elements

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.

FILENAME
: associates a SAS fileref with an external text file or output device. See "FILENAME Statement" on page 36 for a more detailed description of this statement.

RUN statement [7]
: executes the statements in the PROC step.

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.

LIBNAME
: associates a libref with a SAS library. See "LIBNAME Statement" on page 36 for a more detailed description of this statement.

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=). See Chapter 3, "Getting Started With SAS/GRAPH," on page 39 for information on using ODS with SAS/GRAPH procedures.

OPTIONS statement
: changes the value of one or more SAS system options.

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 these statements, see "Statements" in *SAS Language Reference: Dictionary*.

FILENAME Statement

The FILENAME statement associates a SAS fileref with an external text file or output device. With SAS/GRAPH software, you can use a FILENAME statement to to 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 111 for information on 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 you want to reference. For details on 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 software 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 97 for information on 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 on 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 on using and managing SAS/GRAPH programs to create graphics output, see Chapter 3, "Getting Started With SAS/GRAPH," on page 39.

- For more information on bringing SAS/GRAPH output to the Web, see Chapter 16, "Introducing SAS/GRAPH Output for the Web," on page 439.
- For information on using and managing SAS/GRAPH output, see Chapter 7, "SAS/GRAPH Output," on page 87.

CHAPTER

3

Getting Started With SAS/GRAPH

Introduction

Like other SAS procedures, the output from SAS/GRAPH procedures is controlled by ODS (Output Delivery System). ODS controls where your output is sent, which could be to a file, to the GRAPH window, directly to a printer, and so on. By default, ODS also applies a style to your output. Styles set the overall appearance of your output; that is, the colors and fonts that are used.

SAS/GRAPH uses device drivers to generate graphics output. SAS/GRAPH device drivers control the format of your graphics. For example, they determine whether SAS/GRAPH produces a PNG file, an SVG file, or an ActiveX control.

Note: This document deals only with device-based graphics. See “Device-Based Graphics and Template-Based Graphics” on page 6. △

Each ODS destination is associated with a default style and a default graphics device to optimize your output for that destination. However, using ODS statements and SAS/GRAPH statements and options, you can customize all of the aspects of your output, including where your output is sent, its appearance, and the format of your graphics.

ODS destination	The ODS destination controls where your output is sent, such as to a file or directly, to a printer, and so on. The ODS destination is specified by the ODS destination statement.
ODS style	The ODS style controls the appearance of your output, including colors and fonts. The ODS STYLE= attribute in the ODS destination is specified by the ODS style statement.
SAS/GRAPH device	The SAS/GRAPH device controls the format of your graphics output such as PNG, GIF, SVG , and so on. The SAS/GRAPH DEVICE= option is specified in the GOPTIONS SAS/GRAPH DEVICE= statement

Note: The LISTING destination is unique. For the LISTING destination, the device controls 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 you want.

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

Introduction to ODS Destinations and Styles

ODS destinations determine where your SAS/GRAPH output is sent. For example, the LISTING destination sends output to the GRAPH window (by default), and the HTML destination sends output to an HTML file. 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 LISTING (the GRAPH window, by default), HTML, RTF, and PDF, but other destinations are also available.

To generate output from SAS, a valid ODS destination must be open. By default, the LISTING destination is open, but you can open other destinations as needed by specifying an ODS destination statement. Depending on the options available for the destination, you can specify options such as the filename or the path to an output directory. With the exception of the LISTING destination, you must also close the destination before output is generated.

```
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 HTML destination, you would specify

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

For more information on ODS destinations, see "Managing ODS Destinations" on page 191 and "ODS Destination Statement Options" on page 192.

The LISTING Destination

The LISTING destination is open by default. If you are sending output to other destinations and are not interested in the output that is sent to the LISTING destination, you should close it to conserve resources. The usual practice is to close LISTING at the beginning of your program and to reopen it at the end. This practice ensures that you always have one open destination. See "Closing Destinations To Save System Resources" on page 51 for more information.

The LISTING destination is somewhat different from other ODS destinations. For the LISTING destination, if you do not specify a device, then your output is sent to the GRAPH window. However, if you specify a device, then where your output is sent is determined by the device. 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. See "Controlling the Graphics Output Format With the DEVICE= Option" on page 48 and Chapter 6, "Using Graphics Devices," on page 67 for more information about devices.

The LISTING destination is the only destination that does not have to be closed before output can be generated.

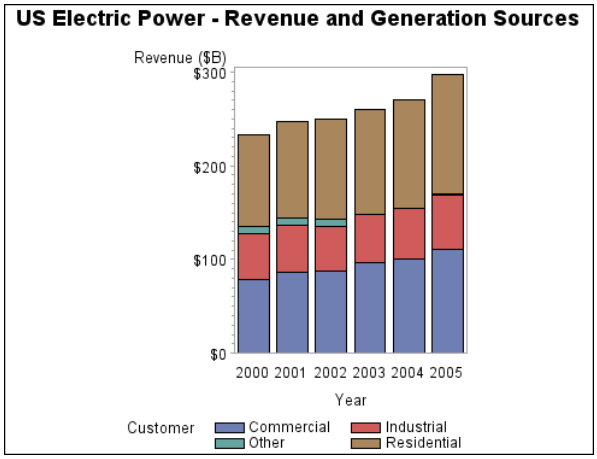
Introduction to Styles

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 3.1 on page 42. Many additional styles (see "Viewing the List of Styles Provided by SAS" on page 141) are available in the STYLES item store in SASHELP.TMPLMST.

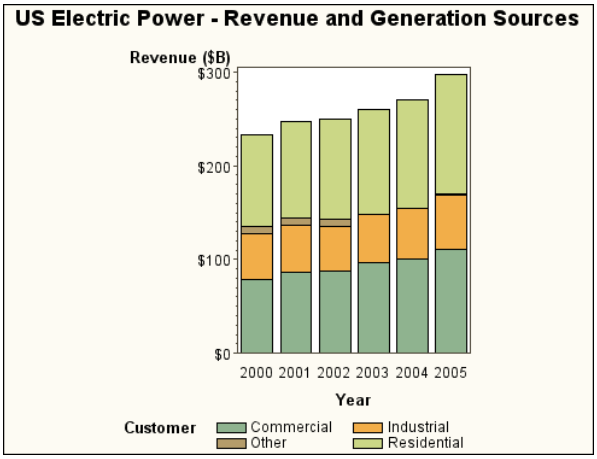
Each destination has a default style associated with it. For example, the default style for the PDF destination is Printer, and the default style for the HTML destination is Default. See "ODS Destinations and Default Styles" on page 135 and "Recommended Styles" on page 136 for more information.

Table 3.1 Examples of Styles Available in SASHELP.TMPLMST

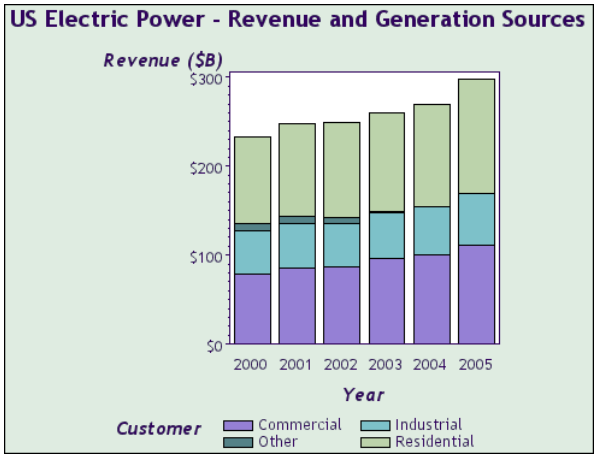
Display 3.1 Style=Statistical



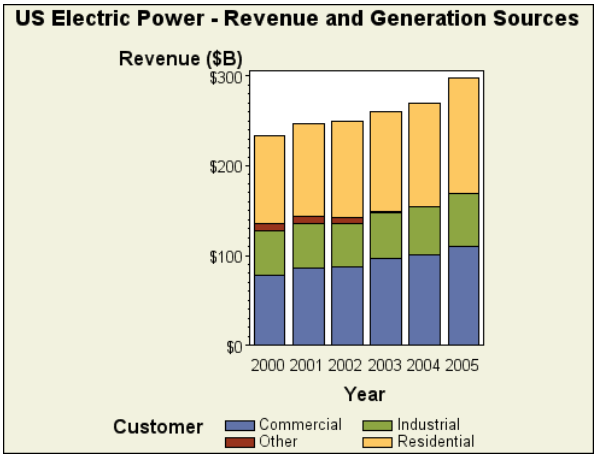
Display 3.2 Style=Analysis



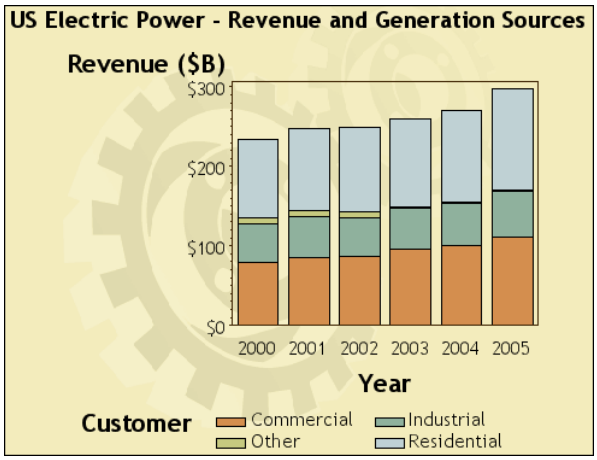
Display 3.3 Style=Ocean



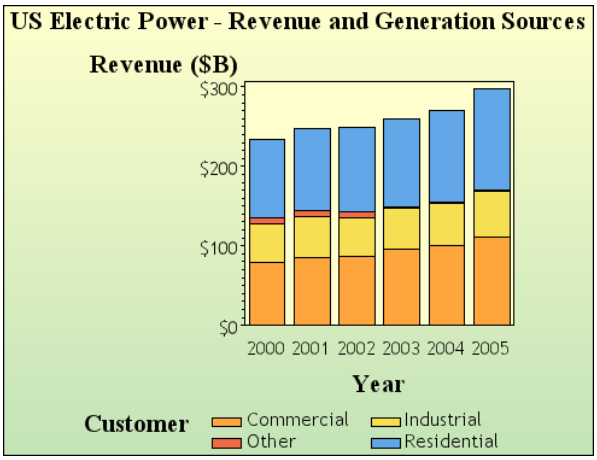
Display 3.4 Style=Harvest



Display 3.5 Style=Gears



Display 3.6 Style=Banker



Specifying a Style

To change the style that is applied to your output, specify the STYLE= option on your ODS destination statement. For example, if you want to change the overall look of your

graph for the HTML destination to the Analysis style, you would specify **style=analysis** in the ODS HTML destination statement as follows:

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

See "About Style Templates" on page 135 and "Specifying a Style" on page 139 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 154 and the GSTYLE system option in *SAS Language Reference: Dictionary* for more information. △

Generating Output With SAS/GRAPH Procedures

ODS provides many destinations to which you can send output. Some of the most often used destinations are LISTING, HTML (a Web page), RTF (an Microsoft Word document), and PDF. As described in "Introduction to Styles" on page 41, each destination is associated with a default style. The following topics each show the default output for each of the destinations listed above.

Each destination is also associated with a default device driver for generating graphics output. Device drivers determine the form that your graphics output takes. For example, the PNG device driver generates PNG image files, and the JAVA device driver generates Java applets that can be run from within HTML pages.

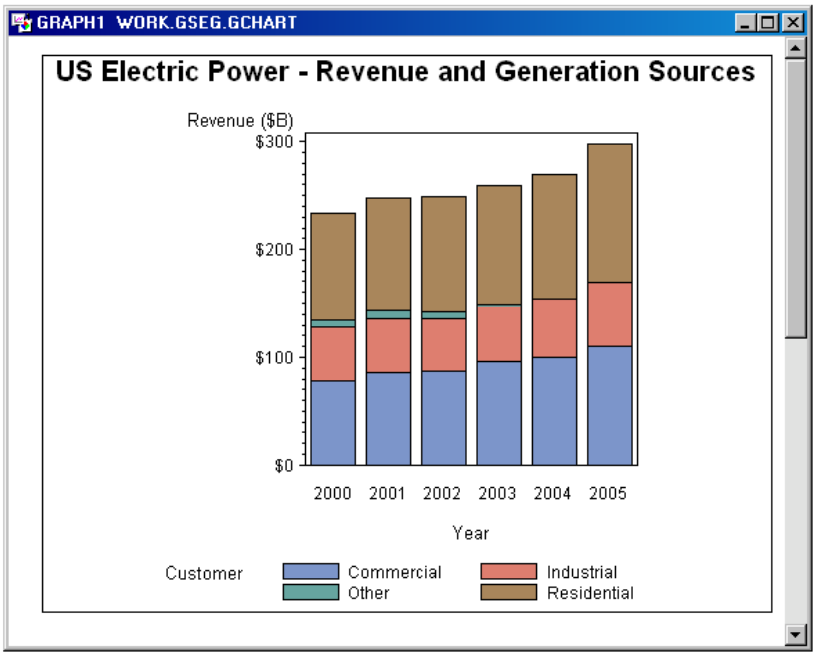
Each destination is associated with a default style and a default device, so you do not need to specify either one to get professional-quality output. You can even send output to several destinations at the same time without specifying either a device or a style.

Sending Output to the GRAPH Window (LISTING Destination)

When working in an interactive environment such as Windows, the LISTING destination is the GRAPH window. By default, the LISTING destination is open, so to send output to it, you simply submit your SAS/GRAPH program. The following example is a simple GCHART program that produces the output shown in Display 3.7 on page 44.

```
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;
```

Display 3.7 LISTING Destination Output Using the Listing Style (Shown in the GRAPH Window)



The default style applied to output sent to the LISTING destination is the Listing style. 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.* For the LISTING destination, the default device driver varies by operating environment. In a Display Manager Session (DMS), 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. If you do not specify a name with the GSFNAME= graphics option, the default name for the procedure or the name specified with the NAME= option is used as the filename.

To create a GIF file with the graph shown in Display 3.7 on page 44, 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.

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

* 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 was an image file such as a PNG, GIF, or JPEG file.

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

By default, the Listing style is applied to the graph as shown in Display 3.7 on page 44. In the FILENAME statement, the current directory is the default SAS output directory.

For more information on sending graphics output to a file, see “Controlling Where Your Output is Stored” on page 97.

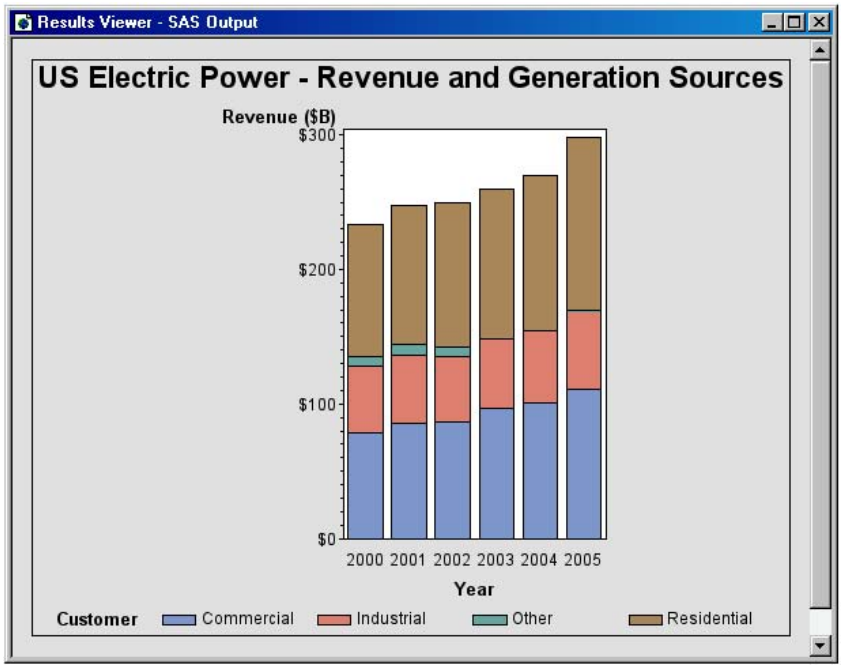
Sending Output to a Web Page

Tosend output to a Web page, send your output to the HTML destination by specifying the ODS HTML statement. This statement opens the HTML destination so that it can receive output. You must also close the HTML destination before output can be generated.

To create a Web page with the graph shown in Display 3.7 on page 44, add the ODS HTML statements around the procedure code.

```
ods listing close;
ods html;
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;
ods listing;
```

Display 3.8 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 Default (STYLES.DEFAULT).

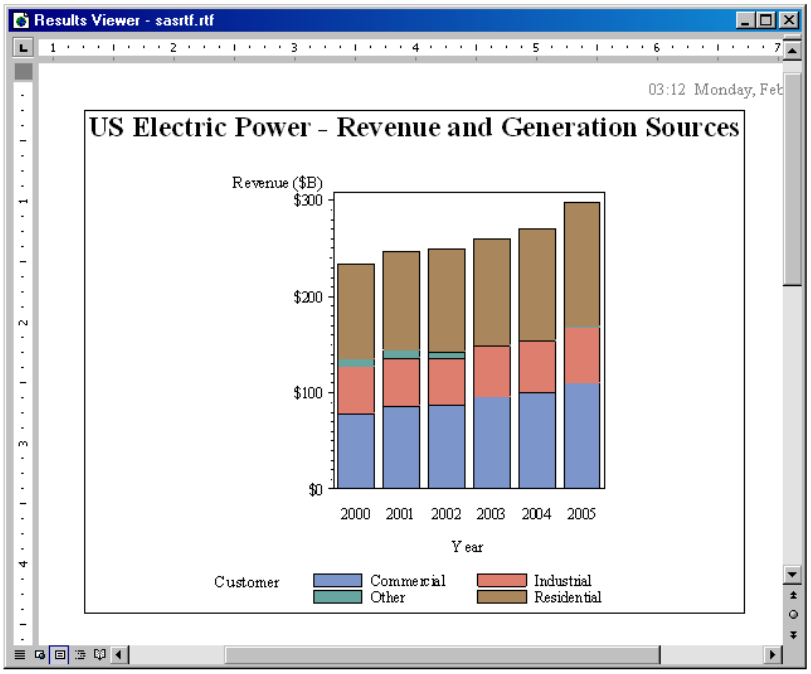
Sending Output to an RTF File (Microsoft Word Document)

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 Display 3.7 on page 44, add the ODS RTF statements around the procedure code.

```
ods listing 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;
ods listing;
```

Display 3.9 RTF Output Using the Rtf Style



By default, SAS/GRAPH creates an RTF file with the graph embedded in it and displays this RTF file in the Results 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.

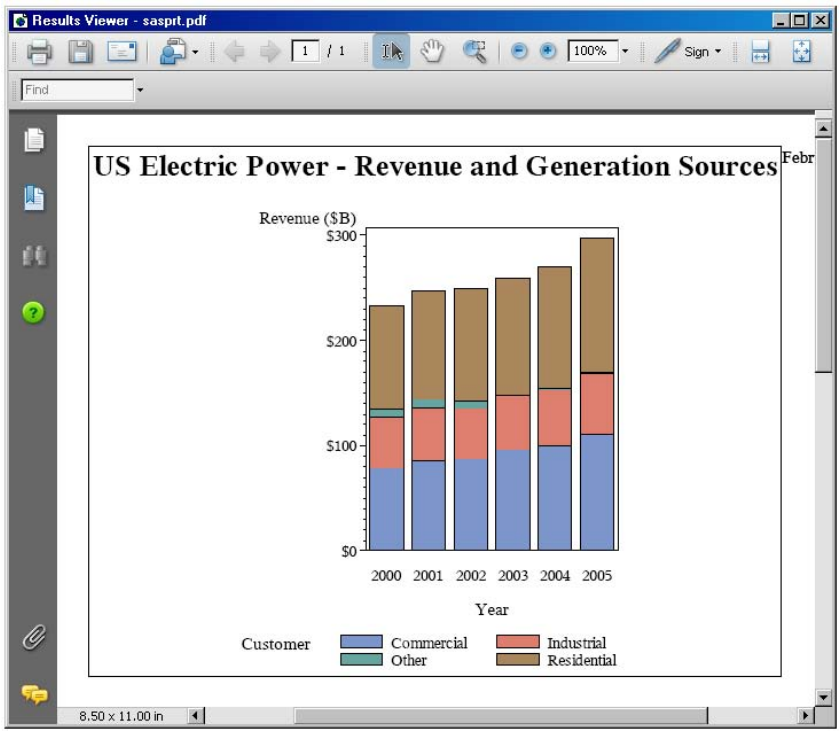
Sending Output to a PDF File

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 Display 3.7 on page 44, add the ODS PDF statements around the procedure code.

```
ods listing 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;
ods listing;
```

Display 3.10 PDF Output Using the Printer Style



By default, SAS/GRAPH creates a PDF file and displays this PDF file in the Results 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 Printer.

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 ACTIVEX device driver generates ActiveX controls that can be run within HTML pages or RTF documents. The SASPRTC device generates images 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 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 inserts the PNG output that was generated.)

Each destination supports several devices. For example, the HTML destination supports the SVG, PNG, GIF, JAVA, and ACTIVEX devices, in addition to several

others. “Viewing The List Of All Available Devices” on page 70 describes how to display the entire list of devices that are available. Table 3.2 on page 50 lists the default and supported devices for the LISTING, HTML, RTF, and PDF 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 with either the OPTIONS statement or the GOPTIONS statement. For example, to use the GOPTIONS statement to change the device, submit this code:

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

Devices that you might specify include PNG, GIF, JPEG, SVG, ACTIVEX, ACTXIMG, JAVA, JAVAIMG, 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 220. “Summary of Default Destinations, Styles, and Devices” on page 49 describes the supported devices for the LISTING, HTML, RTF, and PDF destinations. See also Chapter 6, “Using Graphics Devices,” on page 67.

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 3.2 on page 50 summarizes this information for the LISTING, HTML, 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 3.2 Default Devices and Styles for Commonly Used ODS Destinations

ODS Destination	Default Device	Default Style	Default Output	Recommended Devices
LISTING	WIN (Windows) XCOLOR (UNIX) IBMPCGX (z/OS)	Listing	Graphics output is displayed in the GRAPH window[1]	All devices[2] except JAVA and ACTIVEX
HTML	PNG	Default (Styles.Default)	HTML and PNG file	PNG GIF JPEG JAVA JAVAIMG ACTIVEX ACTXIMG SVG JAVAMETA GIFANIM
RTF	SASEMF	Rtf	RTF file (with embedded metafile)	SASEMF PNG JPEG JAVAIMG ACTIVEX ACTXIMG
PDF	SASPRTC	Printer	PDF file	SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)
PRINTER	SASPRTC	Printer	Controlled by the PRINTERPATH= system option (and by the SYSPRINT= system option on Windows)[3]	SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)

1 The default devices for the LISTING destination do not write image files to disk.
2 JAVAMETA is supported for the LISTING destination, but its output requires processing with the Metaview applet.
3 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.

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

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 LISTING destination is open by default, and you can open as many additional destinations as needed. For example, you can open the HTML 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 for 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 application specifies the following:

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

SAS/GRAPH uses the PNG device and the Default style to generate output for the HTML destination, and it 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, to use the ACTIVEX device and the ANALYSIS style for all output sent to both the HTML and RTF destinations, you would specify the GOPTIONS statement and the STYLE= option as follows:

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

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 topic. For more information on generating output with the SAS/GRAPH procedures, see the following topics:

- Chapter 7, "SAS/GRAPH Output," on page 87
- Chapter 16, "Introducing SAS/GRAPH Output for the Web," on page 439

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

- Chapter 13, "Managing Your Graphics With ODS," on page 191
- Chapter 10, "Controlling The Appearance of Your Graphs," on page 133

For more information on using the graphics devices, see Chapter 6, "Using Graphics Devices," on page 67.

CHAPTER

4

SAS/GRAPH Processing

Running SAS/GRAPH Programs

Here are the environments and modes in which you can run a SAS/GRAPH program:

- The *SAS windowing environment* provides a text editor for submitting programs, windows for the SAS log and SAS output, and many other facilities. For more information on the SAS windowing environment see "Introduction to the SAS Windowing Environment" in *SAS Language Reference: Concepts*.
- *Interactive line mode* enables you to submit programs one line at a time in response to prompts from the SAS/GRAPH system. In interactive line mode, the SAS/GRAPH program can display graphics output on your monitor as well as store the output in a file.
- *Noninteractive mode* enables you to issue a SAS command that executes a SAS/GRAPH program that is stored in an external file. This mode is valid only in your current terminal session. In this mode, the SAS/GRAPH program can display graphics output on your monitor as well as store the output in a file.
- *Batch mode* enables you to execute a SAS program (stored in a file) in a separate terminal session. In batch mode, the graphics output is not displayed on your monitor. In this case, your program must send the graphics output to a printer or plotter, permanent catalog, or an external file.

Note: Certain fonts called device-resident fonts are specific to the device being used and therefore are not portable between devices when running in batch mode. See "Overview" on page 1175 for more information on using fonts in batch mode. △

Regardless of how you run your programs, SAS/GRAPH software applies ODS styles by default to your graphics output. For more information on ODS styles see Chapter 10, "Controlling The Appearance of Your Graphs," on page 133.

See Chapter 7, “SAS/GRAPH Output,” on page 87 for more information about SAS/GRAPH output.

SAS Data Sets

Many SAS/GRAPH procedures use and create SAS data sets. SAS data sets are files stored in SAS libraries and can be either temporary or permanent.

When you create a SAS data set, it is stored automatically in the WORK library. Unless you specify a different library, the WORK library serves as a temporary holding place for all the data sets you access and create for the duration of a SAS session. By default, the WORK library and all the data sets stored in it will be removed after the SAS session ends.

You can also create permanent SAS libraries that can be saved in a specified location on your computer. Permanent libraries are not deleted when the SAS session terminates and are available for processing in subsequent SAS sessions.

For more information on SAS data sets and other data processing details, see *SAS Language Reference: Concepts*.

For a complete discussion of SAS data set options and SAS system options, see *SAS Language Reference: Dictionary*.

Specifying an Input Data Set

You can specify an input data set by using one of the following methods:

- a library reference
- a file specification

When using either of these methods, you usually specify the DATA= option in the procedure statement, as shown in this example:

```
proc gplot data=stocks;
```

If you omit the DATA= option, then the procedure uses the SAS data set that was most recently used or created in the current SAS session.

If you do not specify a SAS data set and no data set has been created in the current SAS session, an error occurs and the procedure stops.

Most of the procedures that read data sets or create output data sets accept data set options. SAS data set options appear in parentheses after the DATA= option specification, as shown in this example:

```
proc gplot data=stocks(where=(year=1997));
```

Using a Library Reference

A SAS library is a storage location for SAS data sets in your operating environment. Data sets stored in a SAS Library are created and referenced using either a one- or two-level name. SAS data sets stored in the temporary WORK library are usually specified using a one-level name. Procedures assume that SAS data sets that are specified with a one-level name are to be read from or written to the WORK library. Since temporary SAS data sets are typically stored by default in the WORK data library, you can specify them using a one-level name and SAS knows where to find them. For example, this statement specifies the data set stocks that resides in the WORK library:

```
proc gplot data=stocks;
```

To specify a permanent data set you typically use a two-level name. A permanent library reference is specified in the form *libref.SAS-data-set-name* in which libref identifies a storage location on your host system. A LIBNAME statement associates a libref with the storage location. See also "LIBNAME Statement" in *SAS Language Reference: Dictionary*. For example, these statements specify a permanent data set:

```
libname  reflib 'my-SAS-library';
     proc gplot data=reflib.stocks;
run;
```

You can use a one-level name for permanent SAS data sets if you specify a USER data library. In this case, the procedure assumes that data sets with one-level names are in the User library instead of in the WORK data library. You can assign a User library with a LIBNAME statement or the USER= SAS system option. For example, these statements use a single-level name to specify a permanent data set that is stored in the library identified as the User library:

```
options user='my-SAS-library';
proc gplot data=stocks;
```

For more information on SAS Libraries see "SAS Libraries" in *SAS Language Reference: Concepts*.

Using a File Specification

To use a file specification for specifying a data set, enclose the file specification in single quotation marks. The specification can be a filename, or a path and filename. The specification must follow the file naming conventions of your operating environment.

For example, the following code creates a file named *mydata* in the default storage location, which is the location where the SAS session was started:

```
data 'mydata';
```

The quotes are required for a file specification; if omitted, SAS treats the specification as a library reference. In the above example, if the quotes are omitted, SAS creates the data set in the temporary WORK catalog and identifies it by the name WORK.MYDATA.

To create the file in a location other than the default location, the quoted file specification must include the full path to the desired location.

You cannot use quoted file specifications for the following items:

- SAS catalog names
- MDDB and FDB references
- the _LAST_= system option

Input Data Set Requirements

SAS/GRAPH procedures often have certain requirements for the input data sets they use. Some procedures might require the input data set to be sorted in a certain way while others might require the data set to contain certain variables or types of information. If necessary, you can use DATA steps and Base SAS procedures in your program to manipulate the data appropriately. For more information on the requirements of any given procedure, see the "Concepts" section which is included at the beginning of each procedure overview.

Automatic Data Set Locking

All SAS/GRAPH procedures that produce graphics output automatically lock the input data sets during processing. By locking a data set, SAS/GRAPH software prevents another user from updating the data at the same time you are using it to produce a graph. If data in a data set changes while you are using it to draw a graph, unpredictable results can occur in the graph or your program can end with errors.

Using Engines with SAS/GRAPH Software

In SAS, procedures use *engines* to access data. Characteristics of these engines vary; generally, they enable SAS procedures to access a data library in a particular way. Engines can specify the expected format for the SAS data file, the type of read or write activity that can occur in SAS data files, and so on. In most cases, you use the default engine for the current SAS version and do not need to specify an engine.

For more information about SAS engines, see "Library Engines" in *SAS Language Reference: Concepts*.

RUN-Group Processing

You can use RUN-group processing with the GAREABAR, GBARLINE, GCHART, GKPI, GMAP, 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 or RUN CANCEL statement, or start a new procedure. If you do not submit a QUIT or RUN CANCEL 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, 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=sales;
```

```
   title1 "Sales Summary";
   plot sales*model_a;
run;

      /* second run group */
   plot sales*model_b;
run;
quit;
```

RUN-group Processing with BY statements

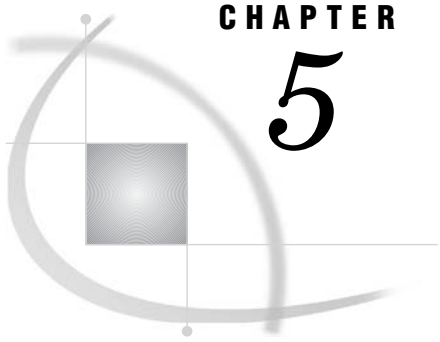
BY statements persist in exactly the same way as global and local statements. Therefore, if you submit a BY statement within a RUN-group, the BY-group processing produces a separate graph for each value of the BY variable for the RUN-group in which you submit it and for all subsequent RUN-groups until 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 "BY Statement" on page 216.

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 these exceptions:

- With the GMAP procedure, the WHERE variable must be in the input data set.
- 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.



CHAPTER

5

The Graphics Output Environment

Overview

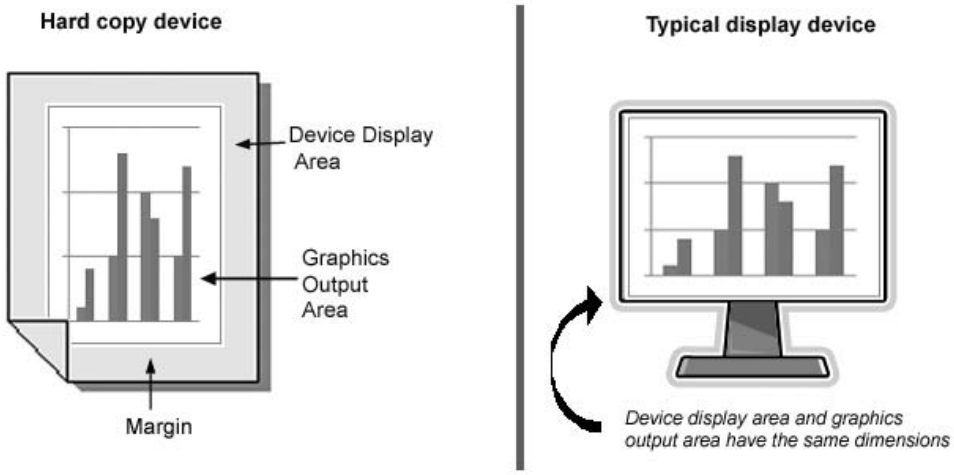
The result of most SAS/GRAPH procedures is the graphic display of data in the form of graphics output. *Graphics output* consists of commands that tell a graphics device how to draw graphic elements. A *graphics element* is a visual element of graphics output—for example, a plot line, a bar, a footnote,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

To generate graphics output, your 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 and make your programs work for different output devices.

The Graphics Output and Device Display Areas

When SAS/GRAPH produces graphics output, it draws the graphic 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, how these changes in dimension affect the output, and the types of units you can specify for your output. For a description of the graphics options and device parameters referred to in this section, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

Controlling Dimensions

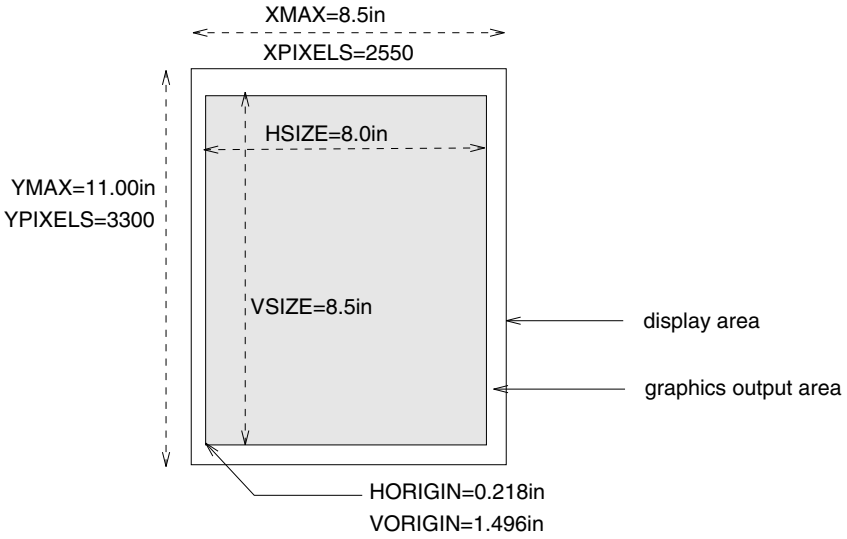
The outer dimensions of the device's display area are controlled by the values of the XMAX and YMAX device parameters. XMAX sets the maximum horizontal dimension; YMAX sets the maximum vertical dimension.

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 5.1 on page 61 illustrates such a device.

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

Figure 5.1 Default Dimensions of the PSCOLOR Device



For further discussion of how the default values for HSIZE and HORIGIN are determined using the value of the LEFTMARGIN option, see "HSIZE" on page 384 and "HORIGIN" on page 382.

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 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 5.1 on page 61 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 on controlling the resolution of your image see "Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for Device-Based Graphics" on page 96.

Units

Cells

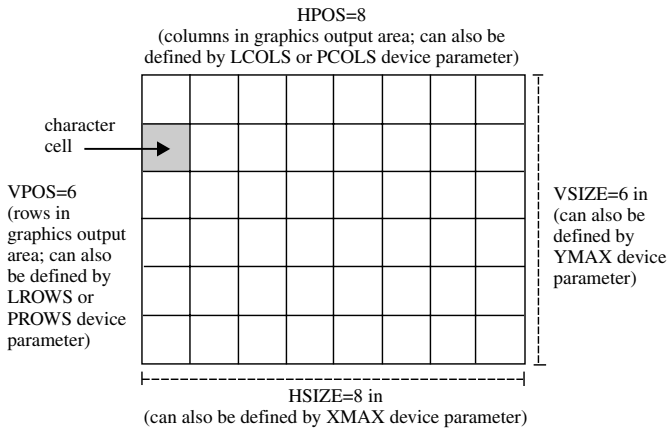
Within the graphics output area, SAS/GRAPH defines an invisible grid of rows and columns. This grid consists of *character cells* as shown in Figure 5.2 on page 62.

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

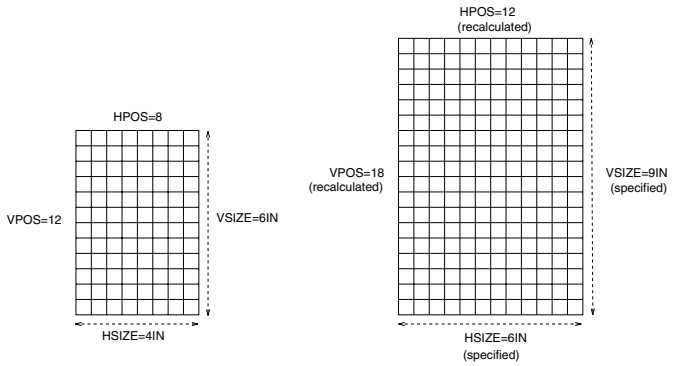
Figure 5.2 on page 62 illustrates how device parameter settings for the size of the output area relate to the parameter settings for the number of character cells in the output area.

Figure 5.2 Rows, Columns, and Cells in the Graphics Output Area



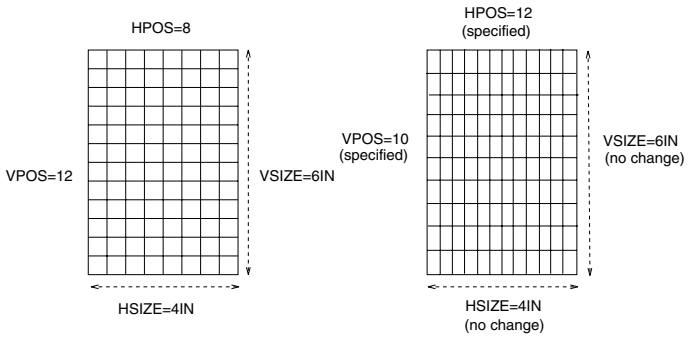
Changing only the outer dimensions of the graphics output area (HSIZE= and VSIZE=) retains the cell size but causes SAS/GRAPH to automatically recalculate the number of rows and columns, as illustrated in Figure 5.3 on page 63.

Figure 5.3 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 5.4 on page 63 shows how increasing the number of rows and columns reduces the size of the individual cells.

Figure 5.4 Changing HPOS= and VPOS= Changes Cell Size



If you use units of CELLS to control the size of the text in your graph while also changing the number of rows and columns, then 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 5.5 on page 64.

Figure 5.5 Changing HSIZE=, VSIZE=, HPOS=, and VPOS= Changes Dimensions and the Number and Size of Cells

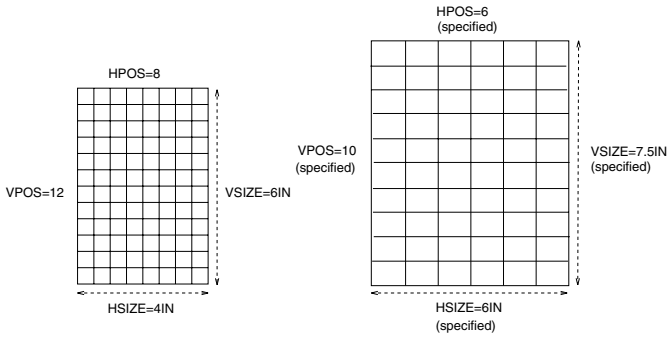


Table 5.1 on page 64 summarizes the interaction of the HSIZE=, VSIZE=, HPOS=, and VPOS= graphics options.

Table 5.1 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 graphic 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 graphic 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 graphic elements in proportion to the size of the graphics output area.

You can specify the unit for individual graphic elements, or you can use the GUNIT= graphics option to set the units for most graphic 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 "GUNIT" on page 378 for a complete description of the GUNIT= graphics option.

Getting the Colors You Want

Since ODS styles are designed to provide optimal results for a variety of devices, you use the STYLE= option in the ODS statement to chose a style best suited for your device. For example, you might want to chose 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 on ODS styles and destinations see "Specifying Devices And Styles With Multiple Open Destinations" on page 51. You can compare colors and patterns for different devices and choose the device that has the fewest colors. A slide camera, for example, offers over 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 Graphic Elements are Placed in the Graphics Output Area

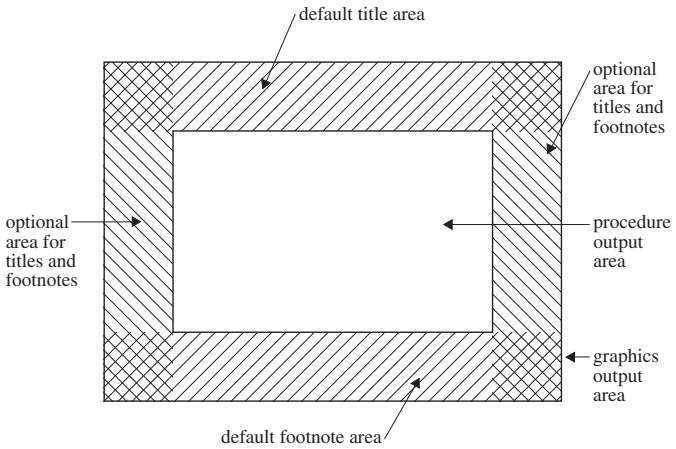
By default, SAS/GRAPH software positions certain graphics elements in predefined locations in the graphics output area. Figure 5.6 on page 66 shows the graphics output area and the areas within it that are used by the following graphic elements:

- 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 "LEGEND Statement" on page 225.

Note: Titles and footnotes can be positioned elsewhere on the graph as well, with different effects on space allocation. See "TITLE, FOOTNOTE, and NOTE Statements" on page 279 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 5.6 Default Locations for Graphic 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 14, "SAS/GRAPH Statements," on page 197 for a description of the appropriate statement. In addition, the section "GOPTIONS Statement" on page 220 lists the graphics options that control the size of various graphic elements. △

How Errors in Sizing Are Handled

Sometimes SAS/GRAPH cannot fit one or more graphic elements on the graph. This can happen if an element is too big for the available space (for example, the title is too long), or if you have too many elements to fit in a given space (for example, a bar chart has 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.

CHAPTER

6

Using Graphics Devices

Overview

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.

Note: SAS/GRAPH produces graphics using two very distinct systems. SAS/GRAPH can produce output using a device-based system or using a template-based system. Template-based graphics (ODS graphics) do not use SAS/GRAPH devices. See "Device-Based Graphics and Template-Based Graphics" on page 6. △

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 Entries" on page 85.

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 they produce.

Table 6.1 SAS/GRAPH Devices and the Output They Generate

Device	External Files
ACTIVEX	This device is used with the ODS HTML and ODS RTF destinations. It generates an HTML or RTF file that contains XML code that is consumed by the ActiveX control. When the HTML or RTF file is viewed in a browser, the SAS/GRAPH output is displayed as an interactive ActiveX control.
ACTXIMG	A PNG file that contains a static image of the graph that is generated with the ACTIVEX device.
BMP	A BMP file that contains the graph

Device	External Files
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.
GIF	A GIF file that contains the graph.
JAVA	This device is used with the ODS HTML destination. It generates a JavaScript that ODS includes in the HTML file. When the HTML file is viewed in a browser, the SAS/GRAPH output is displayed as an interactive Java applet.
IBMPCGX	Display device. This device is available on z/OS hosts only.
JAVAIMG	A PNG file that contains a static graph that is generated with the JAVA device.
JPEG	A JPG file that contains the graph.
PCL5	A PCL file that contains the graph.
PDF	A PDF file that contains one or more graphs and tables.
PNG	A PNG file that contains the graph.
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. This device is the default device for the ODS RTF destination.
SVG	An SVG file that contains the graph.
TIFFP	A TIFF file that contains the graph in color.
WIN	Display device. This device is available on Windows hosts only.
XCOLOR	Display device. This device is available on UNIX hosts only.

Note: Chapter 16, "Introducing SAS/GRAPH Output for the Web," on page 439 describes any requirements or limitations associated with using the ActiveX, Java, and SVG devices. △

Default Devices For ODS Destinations

Each ODS destination has a default device. Table 6.2 on page 70 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 72.

Table 6.2 Default Devices and Styles for Commonly Used ODS Destinations

ODS Destination	Default Device	Default Style	Default Output	Recommended Devices
LISTING	WIN (Windows) XCOLOR (UNIX) IBMPCGX (z/OS)	Listing	Graphics output is displayed in the GRAPH window[1]	All devices[2] except JAVA and ACTIVEX
HTML	PNG	Default (Styles.Default)	HTML and PNG file	PNG GIF JPEG JAVA JAVAIMG ACTIVEX ACTXIMG SVG JAVAMETA GIFANIM
RTF	SASEMF	Rtf	RTF file (with embedded metafile)	SASEMF PNG JPEG JAVAIMG ACTIVEX ACTXIMG
PDF	SASPRTC	Printer	PDF file	SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)
PRINTER	SASPRTC	Printer	Controlled by the PRINTERPATH= system option (and by the SYSPRINT= system option on Windows)[3]	SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)

1 The default devices for the LISTING destination do not write image files to disk.
2 JAVAMETA is supported for the LISTING destination, but its output requires processing with the Metaview applet.
3 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.

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 1136 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 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.

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 "Default Devices For ODS Destinations" on page 69 for each open destination.

Note: If you are working with multiple open destinations, see "Specifying Devices And Styles With Multiple Open Destinations" on page 51. △

If you need to specify a different device, you should specify one of the recommended devices in the table in "Default Devices For ODS Destinations" on page 69. 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 are using a color PostScript printer and you select a device for a black and white PostScript printer, your graph will not print in color.

When you are sending output to the HTML destination, there are several devices that you can specify. See "Selecting a Type of Web Presentation" on page 447 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. △

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 220.
- Specify the name of a device entry with the DEVICE= option in an OPTIONS statement. For details, see "DEVICE= System Option" in *SAS Language Reference: Dictionary*.
- 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 on **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. The DEVICE prompt window opens 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 on 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 110.

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

Device Categories And Modifying Default Output Attributes

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.

Note: Chapter 10, "Controlling The Appearance of Your Graphs," on page 133 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. △

Native SAS/GRAPH devices
: produce output in the native language of the device. For example, TIFFP, PS300, SASEMF, JPEG, CGMC, and GIF are native SAS/GRAPH devices. With native 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 the default size and resolution of your output and the default colors and fonts that are used when styles are turned

off. Native SAS/GRAPH devices do not set and or use the SYSPRINT= or PRINTERPATH= system options.

Java and ActiveX devices
produce output using different technologies than the native SAS/GRAPH devices. These devices are the JAVA, JAVAIMG, ACTIVEX, and ACTXIMG devices. These devices do not use information specified in the device entry.

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 on universal printing, see "Printing With SAS" in *SAS Language Reference: Concepts*.

Universal Printer shortcut devices can generate output in the following formats: PDF, PostScript, PCL, PNG, GIF, and SVG. For example, PNG and SVG are Universal Printer shortcut devices. Any device whose name begins with the letter U, such as UGIF or UPSL, is also a Universal Printer shortcut device. The description of a Universal Printer shortcut device generally says "Universal Printer" when you view the contents of the SASHELP.DEVICES catalog (see "Viewing The List Of All Available Devices" on page 70). The list of all Universal Printer shortcut devices is shown in Table 6.4 on page 76.

Universal Printer shortcut devices are designed to emulate a native SAS/GRAPH device, which means that these devices behave as much as possible like native 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 75 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 to 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 6.3 Device Categories, GOPTIONS, and DEV Entries

Device Category		Examples	Honor GOPTIONS specifications?	Honor Device (DEV) entry specifications?
Native SAS/GRAPH devices		GIF TIFFP JPEG CGM BMP[1] SASBMP[2] EMF, WMF[1] SASEMF, SASWMF[2] JAVAMETA ZPNG IBMPCGX	Yes[4]	Yes
Java and ActiveX devices		JAVA ACTIVEX JAVAIMG ACTXIMG	Yes, except as noted in the documentation for specific graphics options. Also, resolution is controlled by the operating environment.	no
Shortcut devices		PNG UGIF SVG	Yes[3], except for resolution[5]	Yes, except for size, resolution, and fonts
Interface devices	Printer[6]	SASPRTC SASPRTG SASPRTM	Yes, except for resolution[7]	Yes, except for size, resolution, and fonts
	Display	WIN XCOLOR	Yes, except for resolution[9]	Yes, except for size[8], resolution[9], and fonts
	Metafile	BMP[1] EMF, WMF[1]	Yes	Yes, except for resolution[9] and fonts

1 On Windows, BMP, EMF, and WMF are interface metafile devices. In all other operating environments, BMP, EMF, and WMF are native SAS/GRAPH devices.

2 In operating environments other than Windows, SASEMF, SASWMF, and SASBMP are copies of EMF, WMF, and BMP, respectively.

3 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.

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

5 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.

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

7 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.

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

9 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.

See also "Viewing and Modifying Device Entries" on page 85.

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 "Using Universal Printer Shortcut Devices" on page 75.
- 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. However, if you create your own device by copying one of the Universal Printer shortcut device entries, then you must make sure that you define a Universal Printer with the same name as your new device entry. For information on creating a new SAS/GRAPH device, see "Creating a Custom Device" on page 86. For information on defining a new Universal Printer, see "Define a New Printer" in the Printing With SAS section of *SAS Language Reference: Concepts*.

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 6.4 on page 76.

Table 6.4 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 device is set to SASPRTC because SASPRTC is the default device for the PRINTER destination.
The default output filename is controlled by the procedure, for example, sasgraph.png.	The Universal Printer is set to PNG by the PRINTER= or PRINTERPATH= option. The default output filename is controlled by ODS, for example, sasprt.png.

Table 6.5 on page 76 lists all of the Universal Printer shortcut devices that are provided by SAS.

Table 6.5 Universal Printer Shortcut Devices

Name	Description
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
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)
SVG	SVG Universal Printer
SVGT	SVG Transparency Universal Printer
SVGVIEW	SVG Printer w/ Control Buttons
SVGZ	SVG Compressed Universal Printer
UEPS	PostScript EPS (Gray Scale)
UEPSC	PostScript EPS (Color)
UGIF	GIF Universal Printer
UPCL5	PCL5c Universal Printer
UPCL5C	PCL5c Universal Printer

Name	Description
UPCL5E	PCL5e Universal Printer
UPDF	PDF Version 1.3 – color
UPNG	PNG Universal Printer
UPNGT	PNG Universal Printer with Transparency
UPSL	PostScript Level 1 (Gray Scale)
UPSLC	PostScript Level 1 (Color)

Using Scalable Vector Graphics Devices

Scalable Vector Graphics (SVG) is an XML language for describing two-dimensional vector graphics. SAS creates SVG documents based on the World Wide Web Consortium (W3C) recommendation for SVG documents. SAS SVG files are created using the UNICODE standard encoding.

Note: Animation is not supported in SAS 9.2. △

SAS can create SVG documents by using either the SVG Universal Printers or SAS/GRAPH SVG devices. There are four SVG devices: SVG, SVGT, SVGVIEW, and SVGZ. These devices are Universal Printer shortcut devices and are, therefore, intended mainly for use with the LISTING and HTML destinations. With the PRINTER destination, it is recommended that you use the SVG Universal Printers directly (see Table 6.4 on page 76).

The information provided here is limited to creating SVG documents using SAS/GRAPH devices in the LISTING and HTML destinations. For information about creating SVG documents in the PRINTER destination using the SVG Universal Printers, see "Creating Scalable Vector Graphics Using Universal Printing" in *SAS Language Reference: Concepts*.

For detailed information about the SVG standard, see the W3C documentation at http://www.w3.org/TR/SVG.

What Is an SVG Document?

An SVG document produced by SAS/GRAPH is an XML file that contains an **<svg>** element.

SVG document fragment
: any number of SVG graphic or container elements enclosed an **<svg>** element. Typical SVG graphics elements include circle, line, text, image, and many others. These elements draw the graphics that comprise the SVG document.

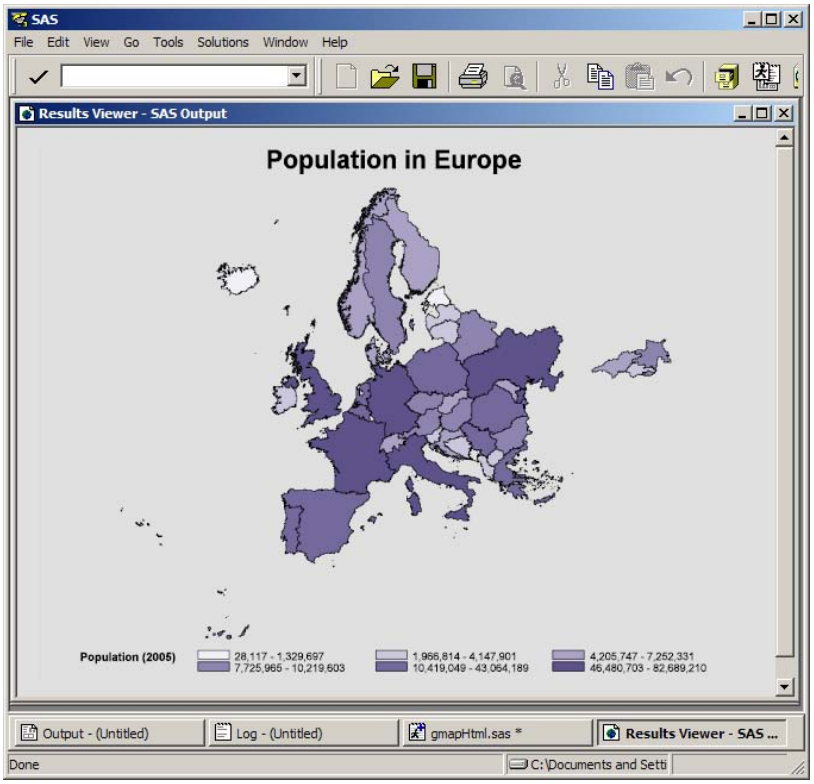
SVG document
: an SVG document fragment that can stand by itself. The SVG devices produce stand-alone SVG documents. When you send output to the HTML destination, the SVG document is embedded in an HTML document using the **<embed>** tag.

For example, the following code produces an SVG file named **europepop.svg** and an HTML file named **europe.htm**:

```
goptions reset=all device=svg;
ods listing close;
ods html file="europe.htm";
```

```
title "Population in Europe";
proc gmap map=maps.europe(where=(id ne 405 and id ne 845))
                  data=sashelp.demographics(where=(cont=93)) all;
    id id;
    choro pop / name="europePop";
run;
quit;
ods html close;
ods listing;
```

You can view the SVG coding by opening the SVG document, **europepop.svg**, in a text editor. When you view the SVG document in an SVG-enabled browser (see "Browsers That Support SVG Documents" on page 83), the browser renders the image.



Why Create SVG Documents?

Because SVG graphics are vector graphics, they can be resized without losing quality. A single SVG document can be scaled to any size or transformed to any resolution without compromising the clarity of the document. Bitmap images such as PNG and GIF lose quality any time 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 document. SVG documents display clearly at any size in any viewer or browser that supports SVG. The user can zoom in to view details in a complicated SVG graphic.

An SVG document might also be smaller in file size than the same graphic created by a bitmap (image) device such as GIF or PNG.

SVG documents are ideal for producing documents to display on a computer monitor, PDA, or cell phone; or documents to be printed.

The SVG Devices and the Output That They Create

There are four SVG devices:

SVG
: produces SVG 1.1 documents. When used in the HTML destination, if your procedure produces multiple graphs, the SVG device produces separate SVG documents for each graph. When used in the LISTING destination, the SVG device produces one SVG file, and the pages are in a continuous layout.

SVGT
: produces SVG 1.1 documents that are transparent (no background). These documents 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 Language Reference: Concepts* for more information.

SVGZ
: produces compressed SVG 1.1 documents, which are useful when file size is an issue. However, some browsers do not support compressed SVG documents, and you cannot view these files in a text editor. (See also "Browsers That Support SVG Documents" on page 83.)

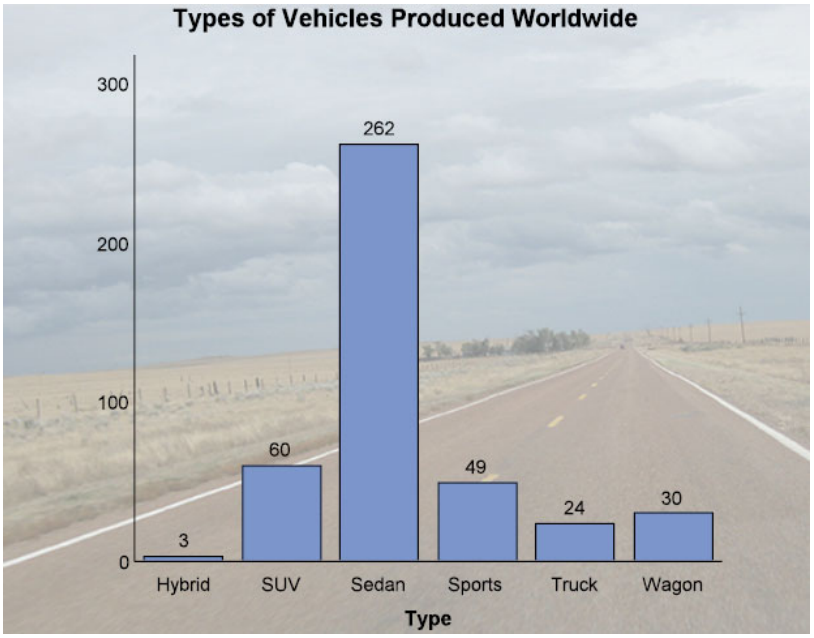
SVGVIEW
: produces SVG1.1 documents with navigational controls when the SVG file contains multiple pages. This device is primarily for use in the LISTING destination with procedures that produce multiple graphs. The navigational controls enable you to page through the graphs. See "Example: Generating A Single SVG Document With Multiple Pages and Page Controls" on page 81. When used in the HTML destination, the SVGVIEW device produces separate SVG documents for each graph, just like the SVG device.

Example: Placing Images Behind SVG Documents

You can use the IBACK= graphics option in the GOPTIONS statement to specify the graphics file that you want to be placed behind the SVG graphic. SAS/GRAPH creates a PNG file from the image file that you specify. This PNG file is used as the background image and is referenced in the SVG with an **<image>** tag. The **<image>** tag specifies a relative (not absolute) pathname to the PNG file. If the SVG file is moved, the PNG file must also be moved to the same location. If many images are referenced in an SVG file, it is recommended that you create a new directory and store your SVG file and any images it references in the directory. Then, the entire directory can be moved as a package.

```
/* Reset existing options, specify the SVG device, and   */
/* set the size of the SVG document.  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=4.8in vsize=3.2in
```

```
                imagestyle=fit iback="external-image-file";

/* Close the LISTING destination to conserve resources. */
/* Open the HTML destination and specify */
/* the name of the HTML output file.              */

ods listing close;
ods html file="carType.htm";

/* Specify the title for the graphic file and */
/* define response axis characteristics.     */
title h=2 "Types of Vehicles Produced Worldwide";
axis1 label=none major=none minor=none;

/* Generate the bar chart.  The NAME= option */
/* specifies the name of the SVG file.                */
proc gchart data=sashelp.cars;
   vbar type / raxis=axis1 outside=freq
               noframe name="carType";
run;
quit;

/* Close the HTML destination and    */
/* reopen the LISTING destination. */
ods html close;
ods listing;
```

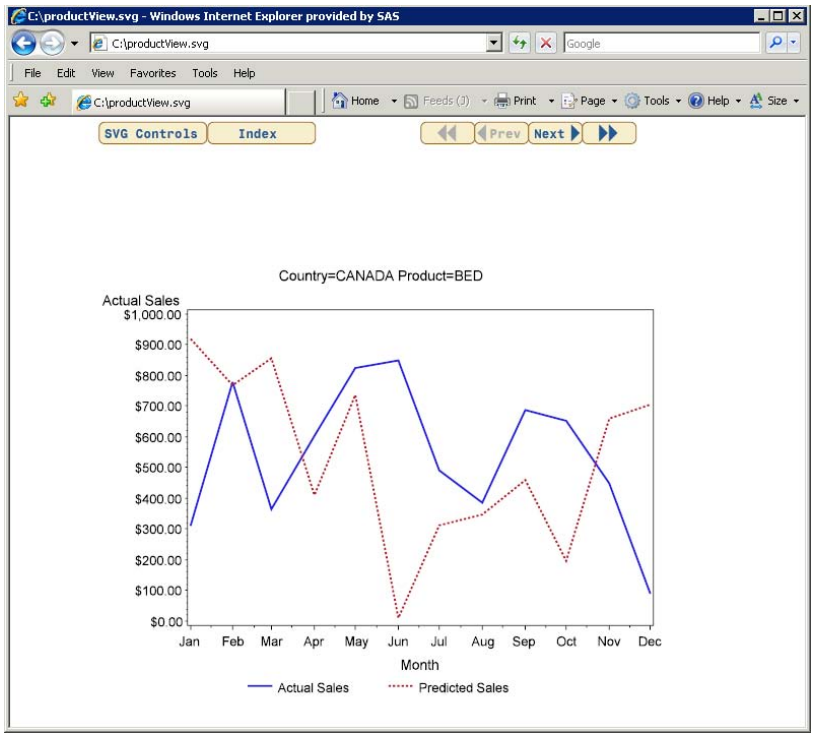


For additional information, see “Displaying an Image in a Graph Background” on page 182.

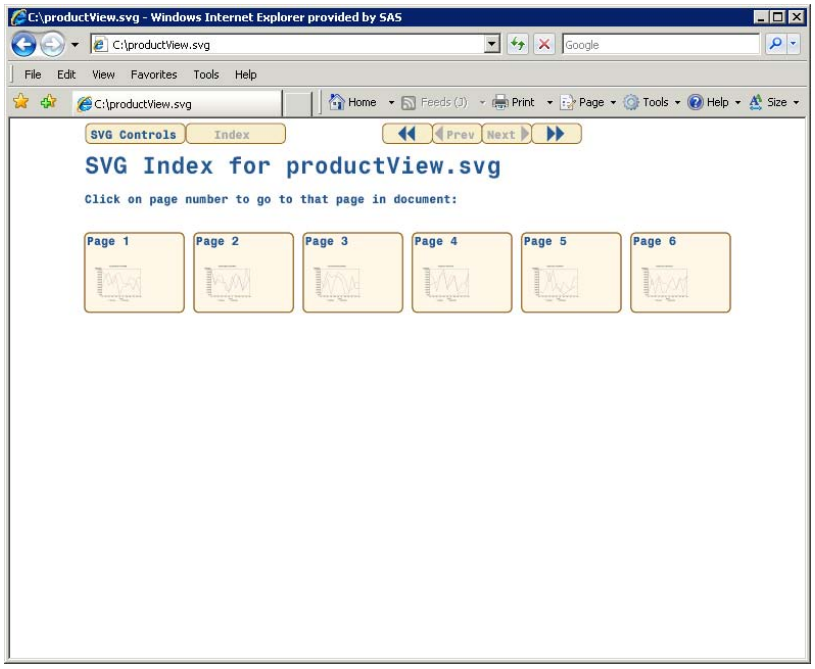
Example: Generating A Single SVG Document With Multiple Pages and Page Controls

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 document with multiple pages. Each graph produced by the procedure is on a different page. The SVG document, by default, has control buttons that enable you to navigate forward and backward through the graphs as well as display an index page that shows a thumbnail image of each page in the document.

For example, the following display shows the initial graph that is produced by the program in Example Code 6.1 on page 82. The program produces six graphs. You can page through them clicking on using the **`Prev`** and **`Next`** buttons.



The **`Index`** button displays a page of thumbnail images. There is one thumbnail for each page in the SVG document.



The program that generates this SVG document is as follows:

Example Code 6.1 Program Code: Using SVGVIEW Device With BY-Group Processing

```
/* 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;

/* Define a fileref for the SVG document. */
/* Use the GSFNAME= option to send the    */
/* output of the LISTING destination to   */
/* that fileref.                          */

filename mysvg "productView.svg";
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;
```

When used in the HTML destination, the SVGVIEW device produces separate SVG documents for each graph, just like the SVG device.

For additional information, see "Multi-Page SVG Documents in a Single File" and "Multi-Page SVG Documents in a Single File" in *SAS Language Reference: Concepts*.

Implementing Drill-Down Functionality With the SVG Devices

You can implement drill-down links in SVG documents 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 with the HTML= and HTML_LEGEND= Options" on page 601 for information on implementing drill-down links, including defining link variables.

Implementing drill-down links in SVG documents 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;
```

Web Server Content Type for SVG Documents

If the mime content type setting for your Web server does not have the correct setting for SVG documents, your Web browser might render SVG documents as text files or SVG documents might be unreadable.

To ensure that SVG documents are rendered correctly, you can configure your Web server to use this mime content type:

```
image/svg+xml
```

Browsers That Support SVG Documents

In order to view SVG documents, you need a viewer or browser that supports Scalable Vector Graphics. Some browsers, such as Mozilla Firefox, have built-in support for SVG documents. Other browsers, such as Microsoft Internet Explorer, require an SVG plug-in to view SVG documents. One such plug-in is available from Adobe Systems, Inc.

The following table lists some browsers and viewers that support SVG documents. See "Browser Support for Viewing SVG Documents" in *SAS Language Reference: Concepts* for additional information.

Table 6.6 SVG Browser Support

Browser or Viewer	Company
Adobe SVG Viewer 3[1]	Adobe Systems, Inc.
Batik SVG Toolkit	Apache Software Foundation
eSVG Viewer and IDE	eSVG Viewer for PC, PDA, Mobile
GPAC Project	GPAC
Mozilla Firefox[2]	Mozilla Foundation
Opera	Opera Software
TinyLine	TinyLine

1 Adobe SVG Viewer 3 works in Internet Explorer 7. Check **www.adobe.com** for information on support by Adobe Systems, Inc. for the SVG viewer.

2 Mozilla Firefox does not support compressed SVG documents or font embedding. To avoid font mapping problems, specify the NOFONTEMBEDDING system option. Zooming and panning features are not currently implemented. Also, if you select **View ▶ Page Style ▶ No Style**, all graphs appear as a black rectangle.

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. To change the resolution for these devices, you must use either the Print Setup dialog box or the PRTDEF procedure to change the resolution for the Universal Printer. Universal Printers have a fixed set of supported resolutions.

To use the Print Setup dialog box, select **File ▶ Print Setup**, and select the printer for which you want to change the resolution. Select **Properties** and click on the **Advanced** tab. Select the resolution that you want to use from the list.

For information on using the PRTDEF procedure, see “The PRTDEF Procedure” in *Base SAS Procedures Guide*.

Controlling Graph Size With the SVG Devices

The default graph size for the SVG output is 600 x 800 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 Chapter 15, “Graphics Options and Device Parameters Dictionary,” on page 327 and “SAS System Options and SVG Output” on page 84.

SAS System Options and SVG Output

Because the SVG devices are Universal Printer shortcut devices, there are several SAS system options that affect the way the SVG devices generate output. These options include SVGHEIGHT=, SVGWIDTH=, SVGVIEWBOX=, SVGCONTROLBUTTONS, and PAPERSIZE=, among others. These options and their interactions are described in several topics in *SAS Language Reference: Concepts* under “Creating Scalable Vector Graphics Using Universal Printing” . Before reviewing the topics that deal with the various system options, you should review the topic “SVG Terminology” .

Topics dealing primarily with SAS system options are as follows:

- "SAS System Options That Effect Stand-alone SVG Documents"
- "Scaling an SVG Document to the Viewport"
- "Setting the ViewBox"
- "Preserving the Aspect Ratio"
- "Interaction between SAS SVG System Options and the SVG Tag Attributes"

Viewing and Modifying Device Entries

As described in "What Is a SAS/GRAPH Device?" on page 68, 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. See "Device Categories And Modifying Default Output Attributes" on page 72 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 1126 or ask your on-site SAS support personnel.

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

- 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 in the Output window the contents of the PSCOLOR device entry:

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

- Run the GDEVICE procedure in windowing mode. The following statements open the GDEVICE directory window that 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 1136.

Modifying Device Entry Parameters

Use the GDEVICE procedure to modify the properties of an existing device entry. See Chapter 38, "The GDEVICE Procedure," on page 1125.

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 they can display.

Some output devices cannot display color. If you try to increase the size of the display past the device's capability or if you specify colors for a device that cannot display them, you will get unpredictable results. 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 unless 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 1142 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 85 and Chapter 38, "The GDEVICE Procedure," on page 1125 for more information.

Related Topics

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

"Devices" on page xvii
describes changes in device support for the current release.

Chapter 7, "SAS/GRAPH Output," on page 87
provides general information about graphics output formats and the SAS/GRAPH output process, setting the size and resolution of your graphics output, previewing on one device how output will look on another device, sending output directly to a printer or other hardcopy device, and replaying output.

Chapter 16, "Introducing SAS/GRAPH Output for the Web," on page 439
describes the options available for creating a Web presentation. Several devices can be used to create a Web presentation, including JAVA and ACTIVEX, which create interactive presentations.

Chapter 8, "Exporting Your Graphs to Microsoft Office Products," on page 113
describes how to choose a device for output that you want to use in Microsoft Office products.

Chapter 38, "The GDEVICE Procedure," on page 1125
describes how to create and modify devices.

- creating files in graphics formats that can be viewed with a Web browser with other applications (see "Graphics Output Files" on page 92).

CHAPTER 7

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 graphic 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 88.
Image file	A file that contains bitmapped graphic information. Examples include GIF, PNG, and JPEG 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, l, 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 Power Point, you might prefer to create a CGM 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 or Java applet

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

- an HTML file that displays a graph
- an RTF file that contains a graph
- a PCL file that contains a graph
- a PDF file that contains a graph
- a PostScript file that contains a graph
- 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 105). 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, JAVAIMG, ACTIVEX, and ACTXIMG 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 output 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 a 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, SAS/GRAPH procedures that produce graphics output display the output on your computer screen using either the GRAPH window or the direct-display method. 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 110.
- save it in a temporary or permanent SAS catalog for later replay. See “Replaying Your SAS/GRAPH Output” on page 106.
- 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 111.

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" on page 353.

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 4, "Transporting and Converting Graphics Output," on page 1659.
- 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 1662.
- export them to graphics output files using different graphics file formats. For details, see "Exporting Your Output" on page 111.

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 16, "Introducing SAS/GRAPH Output for the Web," on page 439 for more information on 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 outputs to the LISTING destination, which displays your graph on your monitor and creates a GRSEG in the catalog. You can specify a graphics output device other than your monitor for the ODS LISTING destination, or you can specify a different ODS destination and device. For information on the ODS destinations and the devices that each supports, see Chapter 16, "Introducing SAS/GRAPH Output for the Web," on page 439.

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

Table 7.1 SAS/GRAPH Devices and the Output They Generate

Device	External Files
ACTIVEX	This device is used with the ODS HTML and ODS RTF destinations. It generates an HTML or RTF file that contains XML code that is consumed by the ActiveX control. When the HTML or RTF file is viewed in a browser, the SAS/GRAPH output is displayed as an interactive ActiveX control.
ACTXIMG	A PNG file that contains a static image of the graph that is generated with the ACTIVEX device.

Device	External Files
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.
GIF	A GIF file that contains the graph.
JAVA	This device is used with the ODS HTML destination. It generates a JavaScript that ODS includes in the HTML file. When the HTML file is viewed in a browser, the SAS/GRAPH output is displayed as an interactive Java applet.
IBMPCGX	Display device. This device is available on z/OS hosts only.
JAVAIMG	A PNG file that contains a static graph that is generated with the JAVA device.
JPEG	A JPG file that contains the graph.
PCL5	A PCL file that contains the graph.
PDF	A PDF file that contains one or more graphs and tables.
PNG	A PNG file that contains the graph.
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. This device is the default device for the ODS RTF destination.
SVG	An SVG file that contains the graph.
TIFFP	A TIFF file that contains the graph in color.
WIN	Display device. This device is available on Windows hosts only.
XCOLOR	Display device. This device is available on UNIX hosts only.

Graphics Output Files

When you export SAS/GRAPH output, you run the output through a device 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 16, “Introducing SAS/GRAPH Output for the Web,” on page 439.

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 the SAS System, whereas you must use the SAS System to manipulate SAS GRSEGs. The GREPLAY procedure can be used to replay graph entries stored in catalogs and display them in the GRAPH window. △

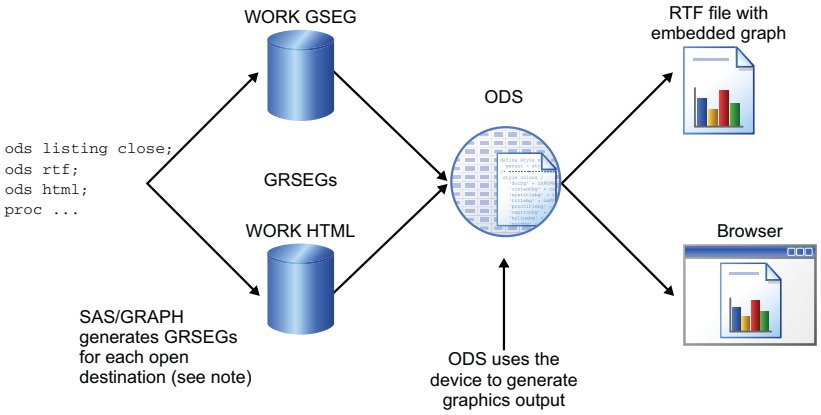
About File Extensions

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” on page 357).

The SAS/GRAPH Output Process

All Devices Except JAVA, JAVAIMG, ACTIVEX, and ACTXIMG

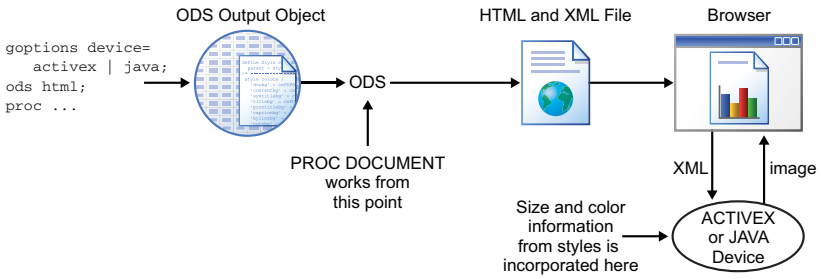
The following diagram illustrates the output process for all of the SAS/GRAPH graphics output devices except JAVA, JAVAIMG, ACTIVEX, and ACTXIMG.



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

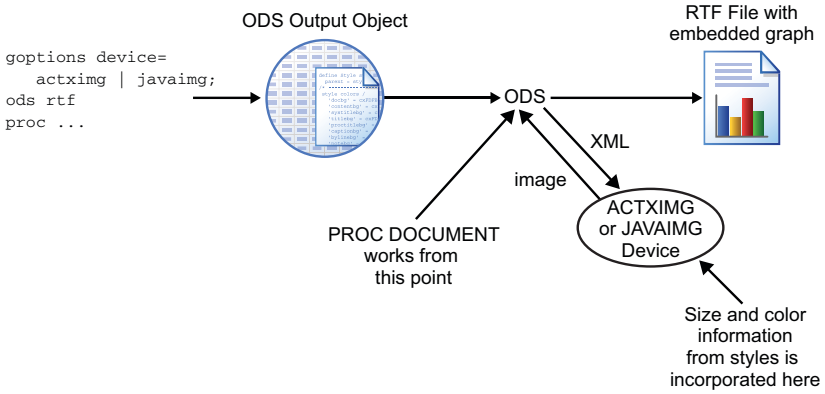
JAVA or ACTIVEX Device

The following diagram illustrates the output process for the JAVA and ACTIVEX graphics output devices.



JAVAIMG or ACTXIMG Device

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



Setting the Size of Your Graph

You can use graphics options to control the size of your graph. Each device uses a default size for the graphics that they generate. You can use the HSIZE= and VSIZE= graphics options to override the default size of your graphics area, or the XPIXELS= and YPIXELS= graphics options to override the default 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). Here is an example that creates a 20 centimeter wide by 10 centimeter high GIF image of a graph.

```
option gstyle;
ods listing style=statistical;
goptions reset=all device=gif hsize=20cm 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

For devices other than the default display devices and the Universal Printing devices, you can use the XPIXELS= and YPIXELS= graphics options to change the default size of the display area for your graph without having to modify the device.

Note: The XPIXELS= and YPIXELS= graphics options are not supported by the default display devices. They are also not supported by Universal Printer devices (including the shortcut devices). The options are partially supported by the ACTIVEX and JAVA devices. △

Setting only the XPIXELS= and YPIXELS= options affects the size of the graph, but does not affect the resolution. Here is an example that creates a 600 pixel wide by 800 pixel high GIF image of a graph.

```
option gstyle;
ods listing style=statistical;
goptions reset=all device=gif 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

To set the resolution of your template-based graphics:

- use the IMAGE_DPI= option in your ODS statement to specify the resolution in DPI.

  See *SAS/GRAPH: Graph Template Language Reference* and *SAS/GRAPH: Statistical Graphics Procedures Guide*.

To set the resolution of your device-based graphics, use one of the following methods:

- For devices other than the default display devices and the Universal Printer devices (including the shortcut devices), use the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options to set the resolution for graphics formats that support variable resolution.
- Use a device variant to set the resolution of your graph to a specific resolution.

Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for Device-Based Graphics

For devices other than the default display devices and the Universal Printer devices, you can use the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options to set the resolution of your graph.

Note: 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 and JAVA devices. △

Note: 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 / XMAX

Table 7.2 on page 96 summarizes the affect of the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options have on the image resolution.

Table 7.2 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.

```
option gstyle;
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;
```

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. Table 7.3 on page 97 lists the GIF device variants that produce images of a specific size.

Table 7.3 GIF Device Variants that Produce Images of a Specific Size

Device Variant	Default Image Size
GIF160	160 x 120
GIF260	260 x 195
GIF373	373 x 280
GIF570	570 x 430
GIF733	733 x 550

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” on page 67.

Controlling Where Your Output is Stored

Specifying the Name and Location of Your ODS Output

By default, ODS output is stored in the default SAS output directory. 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 in the default SAS directory.

```
goptions reset=all;
ods listing close;
ods pdf style=money 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 listing;
```

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 current directory, and the graphics output file to the **images** subdirectory.

```
goptions reset=all;
ods listing close;
ods html style=banker path="./" gpath="images" body="mygraph.html";
proc gchart data=sashelp.prdsale;
   vbar Product / sumvar=actual;
      title1 "First Quarter Sales in Canada";
      where Quarter=1 and Country="CANADA";
   run;
quit;
ods html close;
ods listing;
```

For more information on the PATH=, GPATH=, and BODY= options, see *SAS Output Delivery System: User's Guide* .

Specifying the Name and Location of Your Graphics Output File

When you use the ODS LISTING destination, you can use the GSFNAME= graphics option to send your output to a graphics output file that you specify. 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 102.

Here is an example that shows how to send the output of the GCHART procedure to file mychart.png in the MyGraphs directory.

```
filename graphout "MyGraphs";
goptions reset=all device=png gsfname=graphout;
proc gchart data=sashelp.cars;
   pie Make / name="MYCHART";
      where MSRP <= 15000;
   run;
quit
```

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 filename in the FILENAME statement and omit the NAME= option and get the same result.

If you specify the filename in the FILENAME statement, you must include the proper file extension. See "About File Extensions" on page 93.

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 listing style=banker;
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;
```

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.

About Filename Indexing

When duplicate names occur in graphics output filenames, SAS/GRAPH procedures use indexing systems to determine unique names for new graphics output files. (Numbers are added to the end of the filename to create new filenames.) Two indexing systems are used: ODS Statistical Graphics indexing and catalog-based indexing. ODS Statistical Graphics indexing is used in all ODS Statistical Graphics output and by the procedures listed in Table 7.4 on page 100. All of the other procedures use catalog-based indexing.

Table 7.4 Filename Indexing Systems Used by SAS/GRAPH Procedures

Procedure Type	Indexing System	How To Control Graphics Filenames	Procedure Name
Device-based	Catalog-based	NAME= option in the procedure action statement	All procedures not listed below.
	ODS Statistical Graphics		GAREABAR GKPI GTILE
Template-based	ODS Statistical Graphics	IMAGENAME= option in the ODS GRAPHICS statement	SGDESIGN SGPLOT SGPANEL SGSCATTER SGRENDER

Note: See "Device-Based Graphics and Template-Based Graphics" on page 6 for a description of the procedure types. △

Because two independent indexing systems are used by the SAS/GRAPH procedures, it is possible that graphics output files can be overwritten if you specify the same graphics filename both for procedures that use catalog-based indexing and for procedures that use ODS Statistical Graphics indexing. To avoid this problem, make sure that you specify different names for the procedures that use ODS Statistical Graphics indexing and the procedures that use catalog-based indexing. For example, if your application uses both the GMAP procedure and the GAREABAR procedure, and you are using the NAME= option to specify output filenames, make sure you specify different filenames for each procedure.

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 filename for the graphics output file and omit the NAME= option, the graphics output filename is the name specified in the FILENAME statement, and the GRSEG entry name is the default procedure name. When you specify the filename, make sure that you include the appropriate file extension, such as .cgm, .gif, or .ps.

If you specify an aggregate file storage location instead of a specific filename 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 102 for examples.

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;
proc gchart data=sashelp.cars;
   pie Make / name="MYCHART";
      where MSRP <= 15000;
   run;
quit;
```

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 102 for additional information on 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 a 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";
ods listing style=banker;

proc gchart data=sashelp.cars gout=Mygraphs.cars;
   vbar Make / name="Mychart";
      where MPG_Highway >= 37;
   run;
quit;
```

Table 7.5 on page 101 summarizes the location of the GRSEG based on the NAME= and GOUT= procedure using the GCHART procedure as an example.

Table 7.5 How NAME= and GOUT= Affect the GRSEG Location

NAME=	GOUT=	GRSEG Location
Not specified	Not specified	Gchart in WORK.GSEG
Not specified	CARS	Gchart in WORK.CARS
Not specified	MYGRAPHS.CARS	Gchart in MYGRAPHS.CARS
MYCHART	Not specified	Mychart in WORK.GSEG

NAME=	GOUT=	GRSEG Location
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.

Catalog Name	Content
WORK.GSEG	The GSEGs for ODS LISTING
WORK.HTML	The GSEGs for ODS HTML
WORK.RTF	The GSEGs 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.

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 7.6 on page 103 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 7.6 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 filename: MYGRAPH.GIF
NAME="FRED"	GSFNAME= points to an aggregate file storage location and the catalog is empty.	GRSEG name: FRED external filename: FRED.GIF
NAME="WEATHEROBS"	GSFNAME= points to an aggregate file storage location and the catalog is empty.	GRSEG name:WEATHERO external filename: WEATHEROBS.GIF
NAME= (not specified)	GSFNAME= points to a file named "MYGRAPH.GIF" and the catalog is empty.	GRSEG name: GCHART external filename: MYGRAPH.GIF
NAME= (not specified)	GSFNAME= points to an aggregate file storage location and the catalog is empty.	GRSEG name: GCHART external filename: 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 filename is the complete NAME= value. This is demonstrated by the NAME="WEATHEROBS" example in Table 7.6 on page 103.

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 filename is also incremented, but the filename is the full value of the NAME= option with a number appended. The same number is used for the GRSEG name and the graphics output filename.

If the GSFNAME= graphics option points to a file, the graphics output filename remains the same and the original file is replaced with the new graph by default.

Table 7.7 on page 104 demonstrates how the SAS/GRAPH software increments the GRSEG name and the graphics output filenames when a GRSEG and graphics output file of the same name already exist.

Table 7.7 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 filename: MYGRAPH.GIF
NAME="FRED"	GSFNAME= points to an aggregate file storage location and GRSEG FRED already exists.	GRSEG name: FRED1 external filename: FRED1.GIF
NAME="WEATHEROBS"	GSFNAME= points to an aggregate file storage location and GRSEG WEATHERO already exists.	GRSEG name:WEATHER1 external filename: WEATHEROBS1.GIF
NAME= (not specified)	GSFNAME= points to a file named "MYGRAPH.GIF" and GRSEG GCHART already exists.	GRSEG name: GCHART1 external filename: MYGRAPH.GIF
NAME= (not specified)	GSFNAME= points to an aggregate file storage location and GRSEGs GCHART and GCHART1 already exist.	GRSEG name: GCHART2 external filename: GCHART2.GIF

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 “Introduction” on page 327.

Storing Multiple Graphs in a Single Graphics Output File

If you want to store multiple graphs in a single graphics output file, you can use either the GSFMODE=APPEND and GSFNAME= graphics options, or the GREPLAY procedure.

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. △

A common application of the GSFMODE=APPEND option is in the production of animated GIFs. See "Developing Web Presentations with the GIFANIM Device" on page 519.

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.ps";
```

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 PostScript file for printing.

```
/* Specify graphics output file name */
filename psout "multicharts.ps";

/* Specify style and graphics options */
ods listing style=banker;
goptions reset=all device=pscolor gsfname=psout 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;

/* Enable display, and then replay all of the graphs to psout */
goptions display;
proc greplay
   igout=Work.Mygraphs nofs;
      replay _all_;
   run;
quit;
```

Replaying Your SAS/GRAPH Output

You can use the GREPLAY procedure or the ODS DOCUMENT destination and the DOCUMENT procedure to replay 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 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 to the GRAPH window for viewing:

```
ods listing;
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 105 and Chapter 21, "Generating Web Animation with GIFANIM," on page 519.

For information on the GREPLAY procedure, see Chapter 50, "The GREPLAY Procedure," on page 1473.

Replaying Output Using 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 LISTING.
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 LISTING.

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 pie charts are generated with the JAVA device.

```
/* Create the Mygraphs catalog */
LIBNAME Mygraphs "./";

/* Open the DOCUMENT destination. Specify catalog */
/* Mygraphs.Mydocs for the output and give it write permission */
ods document name=Mygraphs.Mydocs(write);

/* Close the LISTING destination */
ods listing close;

/* Open the HTML destination, and specify the JAVA device. */
ods html style=seaside;
goptions reset=all device=java;

/* Generate the charts */
proc gchart data=sashelp.cars gout=Mygraphs.Mydocs;
   pie Make / other=2;
      title1 "30 MPG or Better";
      where MPG_Highway >= 30;
   run;
   pie Make / other=3;
```

```
      title1 "Between 20 MPG and 29 MPG";
      where MPG_Highway < 30 and MPG_Highway >=20;
   run;
   pie Make / other=3;
      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 LISTING destination */
ods listing;
```

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 example:

```
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\Gchart#2                          Graph
       4 \Gchart#1\Gchart#3                          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.

To replay the output:

1. Close the ODS LISTING destination.
2. Open the ODS destinations that you want to send the output to.
3. Use the DEVICE= graphics option to specify the graphics output device that you want to use to generate the graphs.
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.

   Note: If you want to display all of the graphs, do not specify a path. △

5. Close the ODS destinations that you opened in step 2.

6 Open the ODS LISTING destination.

Here is an example that shows how to play the first and the third graphs in the Mygraphs.Mydocs catalog to the ODS RTF destination using the ACTIVEX device.

```
goptions reset=all device=activex;
ods listing close;
ods rtf style=money;
proc document name=Mygraphs.Mydocs;
   replay \Gchart#1\Gchart#1 / levels=all dest=rtf;
   replay \Gchart#1\Gchart#3 / levels=all dest=rtf;
   run;
quit;
ods rtf close;
ods listing;
```

To replay all of the graphs in the catalog, use one REPLAY statement that does not specify a path. For example:

```
proc document name=Mygraphs.Mydocs;
   replay / levels=all dest=rtf;
   run;
```

For more information on using the ODS DOCUMENT destination and the DOCUMENT procedure, see *SAS Output Delivery System: User's 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" on page 424 for a complete description of TARGETDEVICE=.

Printing Your Graph

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 or to a device port. 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 on the ODS PRINTER statement.

  See the *SAS Language Reference: Concepts* for information on 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, "The GDEVICE Procedure," on page 1125 for information on 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, "The GDEVICE Procedure," on page 1125 for information on modifying device entries.
- Use the Universal Printing interface.

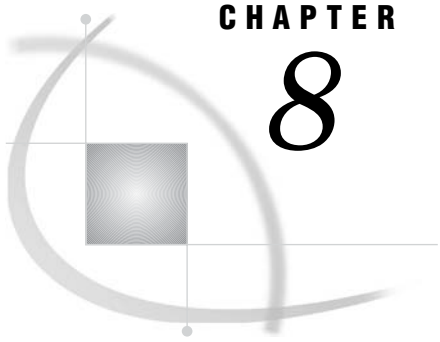
  Note: On Windows platforms, the ODS PRINTER destination uses the Universal Printing interface in addition to the Windows system printers. △

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 107
- Chapter 9, "Writing Your Graphs to a PDF File," on page 123
- Chapter 8, "Exporting Your Graphs to Microsoft Office Products," on page 113



CHAPTER

8

Exporting Your Graphs to Microsoft Office Products

What to Consider When Choosing an Output Format

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
- the fonts you want to use
- 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 graphics format includes:

EMF

WMF

CGM

PNG

JPEG

TIFF

GIF

BMP

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:

Document Format	Compatible Graphics Formats
HTML	PNG, GIF, JPEG, SVG, and ActiveX
RTF	EMF, PNG, JPEG, and ActiveX

Image Resolution and Size

Each of the SAS/GRAPH graphics output devices has a default size and resolution setting for the graphics they generate. For information on the default settings for each device, see “Overview” on page 67. 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 you need in the Microsoft application so that it does not have to be resized after it is imported. See “Setting the Size of Your Graph” on page 94. 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 95

Color Depth

Another consideration when choosing a graphics format is color depth, which 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

CGM

EMF

EPS

PNG

SVG

WMF

See “Overview” on page 67 for information on the color depth supported by each of the graphics output devices.

Fonts

Microsoft Office products use fonts that are native to the Windows operating system, which include TrueType and OpenType fonts. The SAS/GRAPH graphics output devices might support the fonts that you are using in your Microsoft applications. See “Introduction” on page 1643 for information on the fonts that the SAS/GRAPH graphics output devices use.

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 GIFANIM procedure to insert multiple graphs into an animated GIF.

See “Using the GREPLAY Procedure to Store Multiple Graphs in One Graphics Output File” on page 105 and “Exporting Your Output” on page 111 for information on replaying your graphs. See “Developing Web Presentations with the GIFANIM Device” on page 519 for information on using the GIFANIM device.

Ability to Edit: Vector Versus Raster Formats

If you need the ability to edit your graphs using Microsoft or other third-party software, choose a graphics format that enables you to perform the type of editing that you need to do. For vector formats, such as WMF, EMF, SVG, and CGM, you can edit individual text and graphic elements using graphics editing software. Although EPS contains vector graphs, Microsoft products cannot edit an EPS image. 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

The SAS/GRAPH software can generate the following types of graphics output that can be imported into Microsoft products:

EMF and WMF

CGM

PNG

JPEG and TIFF

GIF and BMP

EPS

HTML (PNG)

RTF

ACTIVEX (RTF)

ACTIVEX (RTF)

ACTXIMG (PNG)

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.
- The ACTIVEX device is used with the ODS RTF or ODS HTML destination to create an RTF or HTML file that contains code that is consumed by the ActiveX Control.
- The ACTXIMG and JAVAIMG devices generate a PNG file that contains a static graph that is generated by the ACTIVEX and JAVA devices respectively.
- Procedures that do not support the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG devices produce a GIF file when the ACTIVEX, ACTXIMG, JAVA, or JAVAIMG device is used.

Table 8.1 on page 117 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 on all of the graphics output devices, see “Overview” on page 67.

Table 8.1 Comparison of the Graphics and Document Types

Type	Advantages and Limitations	Devices
EMF and WMF	Advantages: □ Most Windows-based applications recognize the EMF and WMF formats. □ Graphs stored in EMF or WMF 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 and CGM Transparency Limitation" on page 119). □ Only one graph per file is supported.	□ SASEMF and SASWMF □ EMF and WMF
CGM	Advantages: □ Graphs stored in CGM files can be edited after they are imported. □ The image can be resized without a loss of quality. Limitations: □ The format does not support transparency (see "Working Around the EMF and CGM Transparency Limitation" on page 119). □ Because the default CGM filter is not installed by default in Microsoft Office, to import CGM files, you must install the CGM filter (see "About the Default CGM Filter for Microsoft Office" on page 120). □ Although the CGM format supports multiple images per file, not all versions of Microsoft Office can import more than one image per file (see "About the Default CGM Filter for Microsoft Office" on page 120).	□ CGMOFML (landscape) □ CGMOFMP (portrait)

Type	Advantages and Limitations	Devices
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) □ UPNG (no transparency) □ UPNGT (transparency)
JPEG and TIFF	Advantages: □ JPEG is widely used for displaying photographs on the Web. □ Both can store high-resolution graphics. Limitations: □ JPEG uses lossy compression. □ The SAS/GRAPH JPEG device supports only 256 colors. □ TIFF is not a Web graphics format. □ JPEG and TIFF images cannot be resized without a loss of quality.	□ JPEG □ TIFFP (color) □ TIFFB (monochrome)
GIF and BMP	Advantages: □ GIF supports transparent backgrounds. □ GIF can store multiple images per file when it is formatted as an animated GIF. □ Both 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. □ Cannot be resized without a loss of quality.	□ BMP (720x480) □ BMP20 (720 480, BMP 2.0) □ GIF (800x600) □ GIFANIM (1280x1024, multi-image) □ UGIF (Universal Printer)

Type	Advantages and Limitations	Devices
EPS	Advantages: □ Can contain a combination of vector and bitmap objects. □ Can be resized after it is imported into Office 97 or Office 2000. Limitations: □ The images should not be edited after they are imported. □ Because the system display does not use the PostScript language to render the graph, these graphics might be visible only when printed to a PostScript printer. □ Because the preview is created automatically in Office 2002 and later, the image should not be resized after it is imported. □ Although this format can store more than one image per file, an EPS file should contain only one image.	□ UEPS (gray scale) □ UEPSC (color) □ PSEPSF (gray scale) □ PSEPSFA4 (gray scale) □ PSLEPSF (gray scale) □ PSLEPSFC (color)
HTML	Advantages: □ Can store text and graphics. □ In Office 2000 and later, and in Microsoft Word in Office 97, the images are loaded into the document automatically when the HTML is imported. Limitation: In Office 97, the images are not loaded into a PowerPoint or Excel document when the HTML is imported. Only the text and tables are imported.	□ JPEG □ GIF and UGIF □ ACTIVEX □ ACTXIMG and JAVAIMG, which create PNG files □ PNG, PNGT, UPNG, and UPNGT
RTF	Advantages: □ Designed specifically for sharing documents between word processors. □ Can store both text and graphics.	□ JPEG □ ACTIVEX □ ACTXIMG and JAVAIMG, which create PNG files □ PNG, PNGT, UPNG, and UPNGT □ SASEMF and EMF

Working Around the EMF and CGM Transparency Limitation

For the EMF and CGM devices, you can work around the transparency limitation as follows:

□ For EMF, 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.

- For CGM, use the CBACK= graphics to assign a matching background color to the CGM file. The CGM devices do not support the IBACK= graphics option or the IMAGE function. To have an image in the document or slide appear as the background of the graph, edit the graph after it is imported to remove the background created by SAS so that the document background shows through.

About the Default CGM Filter for Microsoft Office

To import CGM files in Microsoft Office, you must install the default CGM filter. For information on the CGM filter and how to install it for your version of Microsoft Office, visit the Microsoft Support Web site:

http://support.microsoft.com

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 8.2 Features that can Enhance Your Graph

Feature in SAS/GRAPH	Reference
Changing the style of the graphic	Chapter 10, "Controlling The Appearance of Your Graphs," on page 133.
Adding annotations to the graph	Chapter 29, "Using Annotate Data Sets," on page 641
Making the graph interactive	Chapter 17, "Creating Interactive Output for ActiveX," on page 453
Adding drill-down links and data tips to the graph	Chapter 27, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 595
Animating the graph	Chapter 21, "Generating Web Animation with GIFANIM," on page 519

Importing Your Graphs into Microsoft Office

This section describes how to import SAS/GRAPH graphics and documents into Microsoft Office 2007 products. For instructions on how to import graphics and documents for other versions of Microsoft Office, contact Technical Support.

Importing Graphs into Microsoft Word

To insert a SAS/GRAPH graphics file into a Microsoft Word 2007 document:

1. If you have not already done so, open your Microsoft Word document and position your cursor where you want to insert your graph.
2. Select the **Insert** tab.

3 On the **Insert** tab, click the **Picture** icon in the Illustrations group. The Insert Picture dialog box opens.
4 In the Insert Picture dialog box, select your graphics output file, and then click **Insert**.

To insert a SAS/GRAPH document into a Microsoft Word 2007 document:

1 Do one of the following based on the type of the document you are importing from:
   - If you are importing from an HTML document, open the document in your Web browser.
   - If you are importing from an RTF document, open the document in Microsoft Word.
2 If you have not already done so, open the target document and position your cursor where you want to insert your graph.
3 In the HTML or RTF document, right-click the graph, and then select **Copy** from the pop-up menu.
4 In the target document, right-click in the page area, and then select **Paste** from the pop-up menu.

If the graph you have imported is an ActiveX graph, you can right-click on your graph in your document and change various attributes of your graph using the pop-up menu. For more information on this menu, select **Help ▸ Graph Control Help** from the pop-up menu.

If the graph you have imported is an animated GIF, you must convert the Microsoft Word document to HTML, and then open the HTML version of your document in your Web browser to play the animated GIF.

Importing Graphs into Microsoft Excel

To insert a SAS/GRAPH graphics file into a Microsoft Excel 2007 spreadsheet:

1 If you have not already done so, open your Microsoft Excel spreadsheet.
2 Locate the cell that you want to import your graph to. Resize the cell to accommodate the graph, if necessary.
3 Select the **Insert** tab.
4 In the **Insert** tab, click **Picture** in the Illustrations group. The Insert Picture dialog box appears.
5 In the Insert Picture dialog box, select your graphics output file, and then click **Insert**.
6 Adjust the size of the graph and cell, if necessary.

To insert a SAS/GRAPH document into a Microsoft Excel 2007 spread sheet:

1 Open the SAS/GRAPH document that you want to import from:
   - If the document is an HTML document, open it in your Web browser or Microsoft Word.
   - If the document is an RTF document, open it in Microsoft Word.
2 If you have not already done so, open your Microsoft Excel spreadsheet.
3 Locate the cell that you want to import your graph to. Resize the cell to accommodate the graph, if necessary.
4 In the HTML or RTF document that you are importing from, right-click your graph, and then select **Copy** from the pop-up menu.
5 In your spread sheet, right-click in the cell that you are importing to, and then select **Paste** from the pop-up menu.

6 Adjust the size of the graph and cell, if necessary.

If the graph you have imported is an ActiveX graph, you can right-click on your graph in your spreadsheet and change various attributes of your graph using the pop-up menu. For more information on this menu, select **Help ▶ Graph Control Help** from the pop-up menu.

Importing Graphs into Microsoft PowerPoint

To insert a SAS/GRAPH graphics file into a Microsoft PowerPoint 2007 presentation:

1 If you have not already done so, open your Microsoft PowerPoint presentation.
2 Locate the slide on which you want to insert your graph. Insert a new slide, if necessary.
3 Click the **Insert** tab.
4 In the **Insert** tab, click **Picture** in the Illustrations group. The Insert Picture dialog box appears.
5 In the Insert Picture dialog box, select your graphics output file, and then click **Insert**.
6 Adjust the size and position of the graph, if necessary.

To insert a SAS/GRAPH document into a Microsoft PowerPoint 2007 presentation:

1 Open the SAS/GRAPH document that you want to import from:
   - If the document is an HTML document, open it in your Web browser or Microsoft Word.
   - If the document is an RTF document, open it in Microsoft Word.
2 If you have not already done so, open your Microsoft PowerPoint presentation.
3 Locate the slide on which you want to insert your graph. Insert a new slide, if necessary.
4 In the HTML or RTF document that you are importing from, right-click the graph, and then select **Copy** from the pop-up menu.
5 In your PowerPoint presentation, right-click in the slide that you are importing to, and then select **Paste** from the pop-up menu.
6 Adjust the size and position of the graph, if necessary.

If the graph you have imported is an ActiveX graph, you can change various attributes of your graph dynamically as follows:

1 Right-click your graph, and then select **SAS Graph v9 Object ▶ Edit** to activate the ActiveX Control.
2 Right-click your graph again, and then select an item from the pop-up menu to change one or more attributes of the graph. You can change the chart type, style, and so on, using this menu. For more information on this menu, select **Help ▶ Graph Control Help** from the pop-up menu.
3 To deactivate the ActiveX Control, deselect your graph.

If the graph you have imported is an animated GIF, you must set the PowerPoint mode to Slide Show to play the animated GIF as follows:

1 In the left panel, select the slide that contains your animated GIF.
2 Click the **Slide Show** tab.
3 On the **Slide Show** tab, click **From Current Slide** in the Start Slide Show group.
4 Verify that your animated GIF plays properly.
5 Press the Esc key to exit the Slide Show mode.

CHAPTER

9

Writing Your Graphs to a PDF File

About Writing Your Graphs to a PDF File

You can use the ODS PDF destination to write your graph output to a PDF Version 1.4 file or a PDF file that is compliant with PDF/A-1b standards and can be archived. You can add multiple graphs to your PDF file with one or more graphs per page. You can also add bookmarks, links, and document metadata in your PDF file, and use system options to change the default page layout of your document.

The ODS PDF destination supports the SAS/GRAPH fonts, the TrueType fonts that are installed with the Base SAS product, 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

Courier/oblique

Courier/bold

Courier/bold/oblique

Helvetica

Helvetica/oblique

Helvetica/bold

Helvetica/bold/
oblique

Times

Times/italic

Times/bold

Times/bold/italic

Symbol

ITC Zapf Dingbats

For more information on fonts, see Chapter 11, “Specifying Fonts in SAS/GRAPH Programs,” on page 155.

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 Chapter 6, “Using Graphics Devices,” on page 67 for information on the PDFA Universal Printer shortcut device. See “Creating a PDF/A-1b-Compliant File that Contains Multiple Graphs Per Page” on page 127 for an example of how to create an archivable PDF file.

Changing the Page Layout

Use the following system options to change the page layout for your PDF document:

- ORIENTATION=PORTRAIT | LANDSCAPE | REVERSEPORTRAIT | REVERSELANDSCAPE
- PAPERSIZE="*paper-size*"
- LEFTMARGIN=*value*
- RIGHTMARGIN= *value*
- TOPMARGIN= *value*
- BOTTOMMARGIN=*value*

See *SAS Language Reference: Dictionary* for information on these system options. See “Creating a Multipage PDF File with Bookmarks and Metadata” on page 125 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

Use the following ODS options to add document metadata to the PDF file:

- AUTHOR="*author-name*"
- KEYWORDS="*word1 word2 ...* "
- SUBJECT="*document-subject*"
- TITLE="*document-title*"

See “Creating a Multipage PDF File with Bookmarks and Metadata” on page 125 for an example of how to add metadata to a PDF file.

Adding Bookmarks for Your Graphs

You can use an ODS PROCLABEL=*label* statement to add bookmarks for your graphs. 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 “Creating a Multipage PDF File with Bookmarks and Metadata” on page 125 for an example.

Changing the Default Compression Level for Your PDF File

You can use the COMPRESS= ODS option 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.

Examples

This section provides the following examples:

Creating a Multipage PDF File with Bookmarks and Metadata

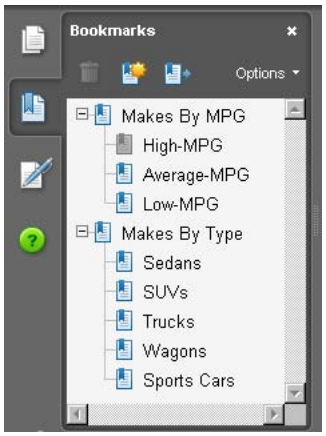
Here is an example that creates a multipage PDF file with bookmarks and metadata using RUN-group processing. Each page displays a single graph in the landscape orientation, and is set up for A4 paper with a 1 cm right, left, and bottom margin, and a 2 cm top margin. The PROCLABEL= ODS option is used to set the top-level bookmark for each category of graphs. The DESCRIPTION= option is used with each procedure to set the text of each subheading bookmark.

```
/* Close the LISTING destination */
ods listing close;

/* Reset the graphics options  */
goptions reset=all;

/* Open the PDF destination */
ods pdf style=seaside
   file="MyDoc.pdf"  /* Output filename */
   compress=0          /* No compression */
   /* Add metadata */
   author="J. L. Cho"
   subject="Auto makers"
   title="Car Makers by MPG and Vehicle Type"
   keywords="automobiles cars MPG sedans trucks wagons SUVs";

/* Modify the PDF page properties */
options orientation=LANDSCAPE
   papersize=A4
   leftmargin=1cm
```

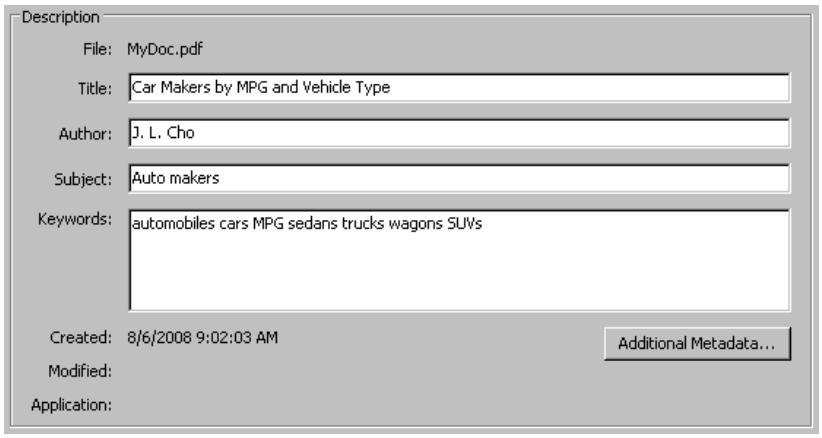
```
   rightmargin=1cm
   bottommargin=1cm
   topmargin=2cm;

/* Set the top-level bookmark for the first set of graphs */
ods proclabel="Makes By MPG";

/* Create the first set of graphs */
proc gchart data=sashelp.cars;
   pie Make / name="HighMPG" other=3
      description="High-MPG"; /* Set subheading text */
      title1 "30 MPG or Better";
      where MPG_Highway >= 30;
   run;
   pie Make / name="MedMPG" other=3
      description="Average-MPG"; /* Set subheading text */
      title1 "Between 20 MPG and 29 MPG";
      where MPG_Highway < 30 and MPG_Highway >= 20;
   run;
   pie Make / name="LowMPG" other=3
      description="Low-MPG"; /* Set subheading text */
      title1 "19 MPG or less";
      where MPG_Highway < 20;
   run;
quit;

/* Set the top-level bookmark for the second set of graphs */
ods proclabel="Makes By Type";

/* Create the second set of graphs */
proc gchart data=sashelp.cars;
   pie Make / name="Sedans" other=3
      description="Sedans"; /* Set subheading text */
      title1 "Sedans";
      where Type = "Sedan";
   run;
   pie Make / name="SUVs" other=3
      description="SUVs"; /* Set subheading text */
      title1 "SUVs";
      where Type="SUV";
   run;
   pie Make / name="Trucks" other=3
      description="Trucks"; /* Set subheading text */
      title1 "Trucks";
      where type="Truck";
   run;
   pie Make / name="Wagons" other=3
      description="Wagons"; /* Set subheading text */
      title1 "Wagons";
      where type="Wagon";
   run;
   pie Make / name="Sports" other=3
      description="Sports Cars"; /* Set subheading text */
      title1 "Sports Cars";
```

```
      where type="Sports";
   run;
quit;

/* Close the PDF destination */
ods pdf close;
ods listing;

/* Reset the graphics options */
goptions reset=all;
```

This creates a PDF file with the bookmarks shown in the following display:



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 display shows the document metadata that is displayed for this example.



Creating a PDF/A-1b-Compliant File that Contains Multiple Graphs Per Page

Here is an example that creates the PDF file FourVbars.pdf, which contains four graphs on one page and can be archived. 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.

```
/* Close the LISTING destination */
ods listing close;

/* Set page options */
options orientation=portrait rightmargin=0.1in leftmargin=0.1in;
goptions reset=all ftext="Helvetica/bold";

/* Open PDF */
ods pdf style=printer
    printer=pdfa         /* Create an archivable PDF */
    file="FourVbars.pdf" /* Output filename */
    startpage=never;     /* Do not insert a pagebreak after each graph */

/* Create a slide for the graphs */
goptions hsize=0 vsize=0;

proc gslide;
    title1 "1997 Quarterly U.S. Sales By State";
run;

/* Size each graph 4in x 4in */
goptions hsize=4in vsize=4in;
title1;

/* Generate the graphs */
proc gchart data=sashelp.prdsal3;
   /* Create the Q1 graph in the top-left quadrant */
   title2 "First Quarter";
   goptions horigin=0 vorigin=5;
   pie State / sumvar=Actual type=mean;
      where country="U.S.A." AND quarter=1 AND Year=1997;
   run;

   /* Create the Q2 graph in the top-right quadrant */
   goptions horigin=4 vorigin=5;
   title2 "Second Quarter";
   pie State / sumvar=Actual type=mean;
      where country="U.S.A." AND quarter=2 AND Year=1997;
   run;

   /* Create the Q3 graph in the bottom-left quadrant */
   title2 "Third Quarter";
   goptions horigin=0 vorigin=0;
   pie State / sumvar=Actual type=mean;
      where country="U.S.A." AND quarter=3 AND Year=1997;
   run;

   /* Create the Q4 graph in the bottom-right quadrant */
   title2 "Fourth Quarter";
   goptions horigin=4 vorigin=0;
   pie State / sumvar=Actual type=mean;
      where country="U.S.A." AND quarter=4 AND Year=1997;
```

```
   run;
quit;

/* Close PDF and reopen LISTING */
ods pdf close;
ods listing;

/* Reset the graphics options */
goptions reset=all;
```

Creating a Multiple-Page PDF File Using BY-Group Processing

Here is an example that uses BY-group processing to create a multiple-page PDF file that contains one graph per page in the landscape orientation.

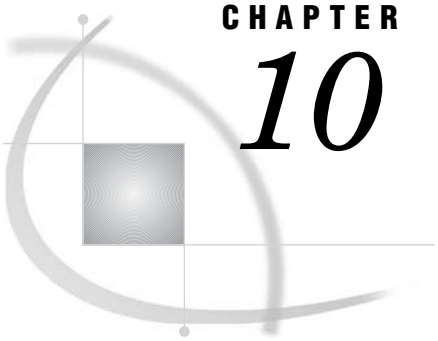
```
/* Specify the landscape page orientation */
options orientation=landscape;

/* Close the LISTING destination */
ods listing close;

/* Reset the options */
goptions reset=all;

/* Open the PDF destination */
ods pdf style=statistical;

/* Create our data set by extracting 1994 data from sashelp.prdsale */
/* and sorting by product */
proc sort data=sashelp.prdsale(where=(Year=1994)) out=work.prdsale;
   by product;
run;

/* Generate the graphs */
title1 "1994 Monthly Sales By Product";
proc gchart data=work.prdsale;
   hbar month /sumvar=actual type=sum sum;
      by product;
   run;
quit;

/* Close the PDF destination */
ods pdf close;

/* Reset the graphics options */
goptions reset=all;

/* Open the LISTING destination */
ods listing;
```

Creating a Multiple-Page PDF File Using the GREPLAY Procedure

Here is an example that uses the GREPLAY procedure to create a PDF file that contains four graphs.

```
/* Specify the landscape page orientation */
options orientation=portrait;

/* Close the LISTING destination */
ods listing close;

/* Reset the options and set NODISPLAY */
goptions reset=all nodisplay;

/* Open the PDF destination */
ods pdf style=statistical file="Mygraph.pdf";

/* Create our data set by extracting 1994 data from sashelp.prdsale */
/* and sorting by quarter */
proc sort data=sashelp.prdsale(where=(Year=1994)) out=work.prdsale;
   by quarter;
run;

/* Delete the old GRSEGs */
proc greplay igout=work.gseg nofs;
   delete _all_;
run;

/* Generate the graphs */
proc gchart data=work.prdsale;
   vbar product /sumvar=actual discrete type=mean mean;
      title1 "1994 Q1 Average Sales By Product";
      where quarter=1;
   run;

      title1 "1994 Q2 Average Sales By Product";
      where quarter=2;
   run;

      title1 "1994 Q3 Average Sales By Product";
      where quarter=3;
   run;

      title1 "1994 Q4 Average Sales By Product";
      where quarter=4;
   run;
quit;

/* Replay the graphs to the PDF file */
goptions display;
proc greplay igout=work.gseg nofs;
   replay _all_;
run;
quit;

/* Close the PDF destination */
ods pdf close;

/* Reset the graphics options */
```

```
goptions reset=all;

/* Open the LISTING destination */
ods listing;
```



CHAPTER

10

Controlling The Appearance of Your Graphs

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 10.1 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" on page 139
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" on page 140
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" on page 142

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 134 and "Turning Off Styles" on page 153 for more information.

Note: This section covers only device-based graphics. See "Device-Based Graphics and Template-Based Graphics" on page 6. △

Style Attributes Versus Device Entry Parameters

The default appearance of SAS/GRAPH output is determined by either style attributes or device entry parameters, depending 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 Java and ActiveX devices. The Java and ActiveX devices always use styles to determine appearance. The setting of the GSTYLE system option has no effect on the Java and ActiveX devices.

Table 10.2 The GSTYLE System Option and Default Appearance

Current Device	GSTYLE	NOGSTYLE
Java or ActiveX device	style	style
All other devices	style	device entry parameters

For information on device entries, see "What Is a SAS/GRAPH Device?" on page 68 and "Viewing and Modifying Device Entries" on page 85. See also "Changing the Appearance of Output to Match That of Earlier SAS Releases" on page 154 and "Turning Off Styles" on page 153.

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: 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, has a default style associated with it. These styles are tailored 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 10.3 on page 135.

Table 10.3 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 139.

Chapter 3, "Getting Started With SAS/GRAPH," on page 39 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 136 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 10.4 on page 136 describes a subset of the styles provided by SAS that are particularly well-suited to displaying graphics.

Table 10.4 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	onochromePrinter	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

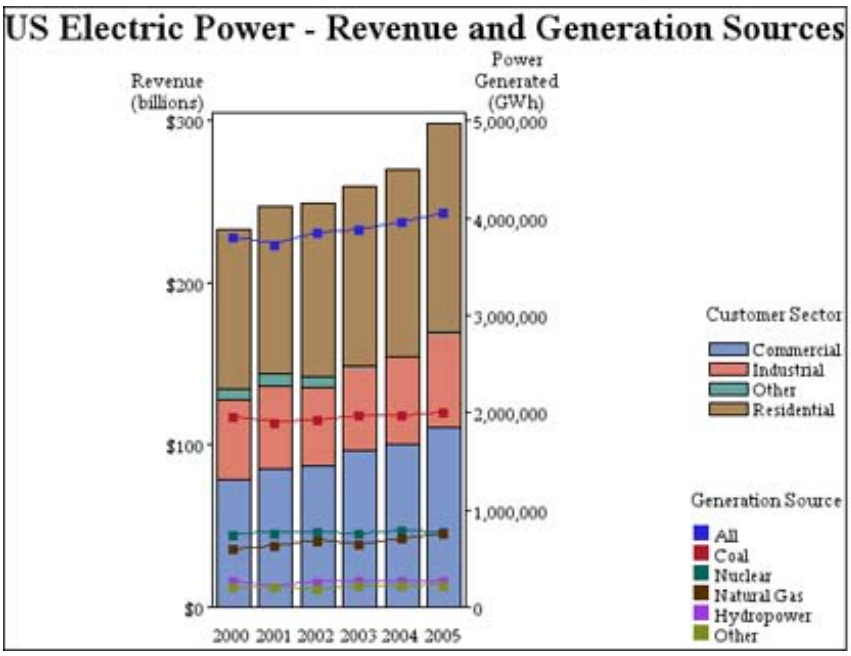
Note: Certain ODS styles map textures onto graph elements. For the Java devices, these textures can be applied to two-dimensional rectangles only. Therefore, styles with textures cannot be applied to three-dimensional bar and pie charts in Java graphs. △

Chapter 3, “Getting Started With SAS/GRAPH,” on page 39 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 136 shows examples of graphs and tables using the Printer, Rtf, Analysis, and Journal styles.

Examples of Output Using Different Styles

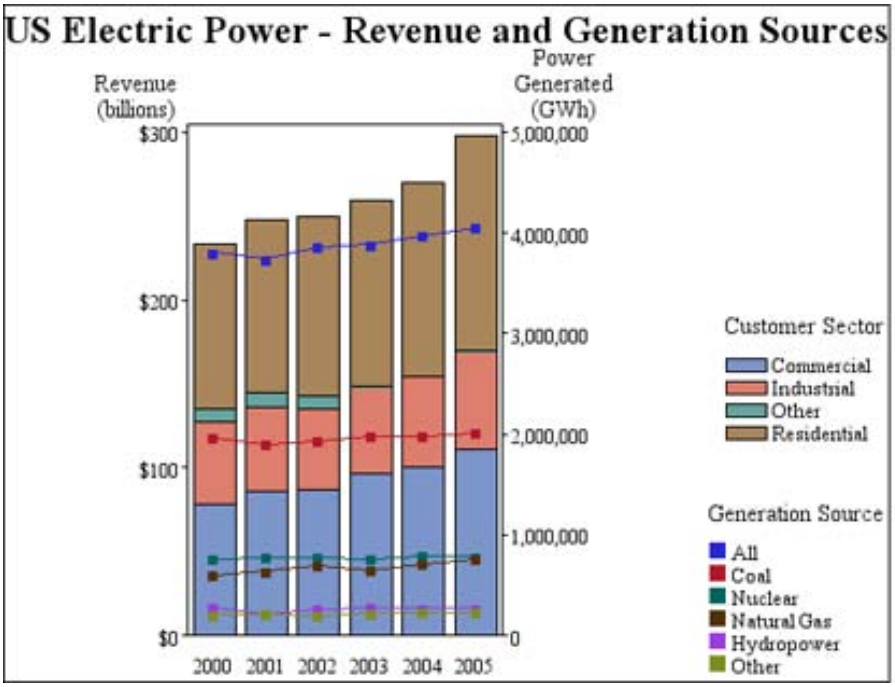
Each of the following sets of output was created using a different style. Additional examples of output in Chapter 3, “Getting Started With SAS/GRAPH,” on page 39.

Figure 10.1 Output Using the Printer Style



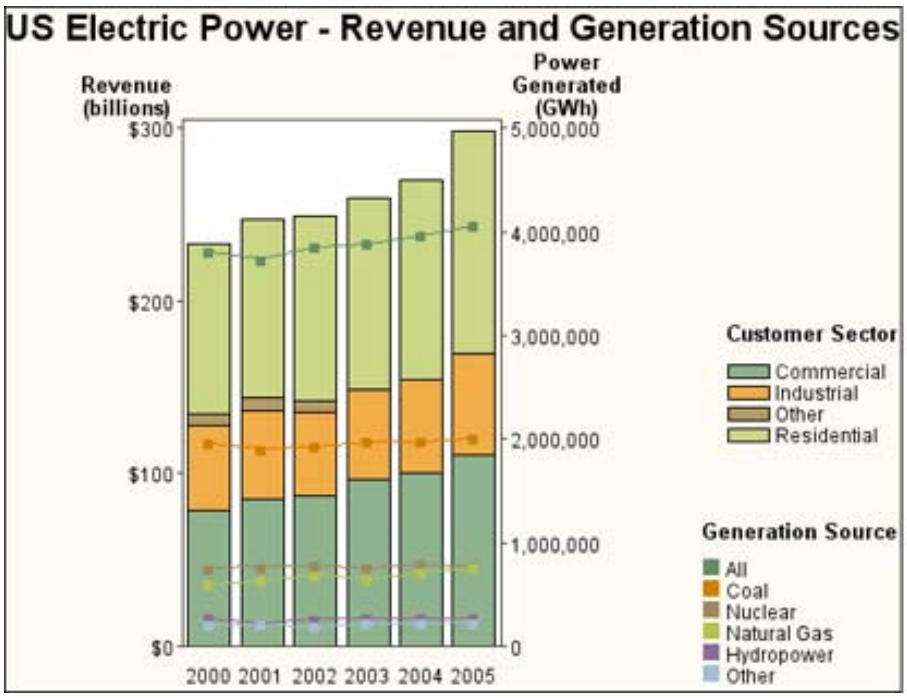
Obs	Customer	Revenue	Year
1	Commercial	$63	1994
2	Industrial	$48	1994
3	Other	$7	1994
4	Residential	$85	1994
5	Commercial	$66	1995

Figure 10.2 Output Using The RTF Style



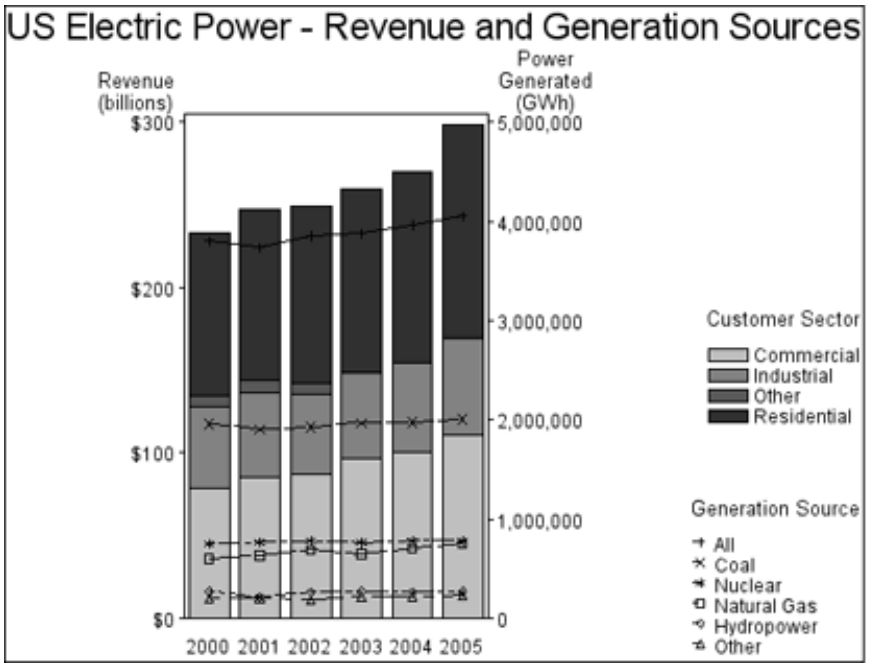
Obs	Customer	Revenue	Year
1	Commercial	$63	1994
2	Industrial	$48	1994
3	Other	$7	1994
4	Residential	$85	1994
5	Commercial	$66	1995

Figure 10.3 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 10.4 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 10.4 on page 138 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 10.3 on page 135. 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 Language Reference: Concepts*. △

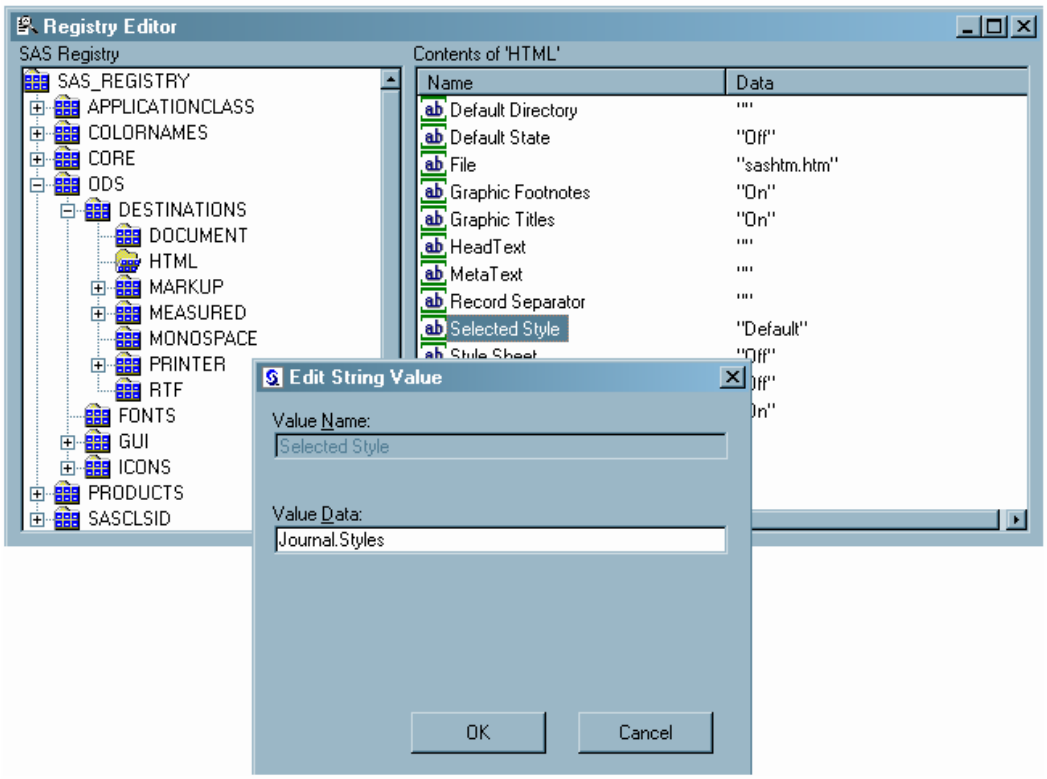
Note: You many have more than one SAS registry. Each site has a SAS registry in SASHELP. Each directory from which you run SAS has an individual registry in SASUSER. If you run SAS from multiple locations, and you want to change default styles via the SAS registry, you might need to change it in multiple locations. For more information, see "The SAS Registry" in *SAS Language Reference: Concepts*. △

For more information on ODS and the SAS registry, see "Changing SAS Registry Settings for ODS" in *SAS Output Delivery System: User's Guide*.

To permanently change the default style for a particular destination:

1. Select **Solutions** ▶ **Accessories** ▶ **Registry Editor**, or issue the command REGEDIT in the SAS command line.
2. Select **ODS** ▶ **Destinations**.
3. Select the destination that you want to change the default style for.
4. Select **Selected Style**, right-click, and select **Modify**. The Edit String Value window appears.
5. Type the style in the Value Data text box and click **OK**.

Display 10.1 SAS Registry Showing Selected Style Setting



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), unless 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 on GOPTIONS, see “GOPTIONS Statement” on page 220 and “Specifying Colors in a GOPTIONS Statement” on page 168. See the examples in Chapter 14, “SAS/GRAPH Statements,” on page 197.

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 6 on page 1441.

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= option for the GCHART procedure includes the following style reference information:

CAXIS=

Style reference: Color attribute of the GraphAxisLines element

If you want to change the color of the same graphical elements that are affected by the CAXIS= option by modifying a style, then you need to modify the Color attribute of the GraphAxisLines element. See “Modifying a Style” on page 142 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 that you send your SAS code to.

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 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 140.

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 definition that you want to view. For example, the Default style definition is the template store for HTML output. Similarly, the Rtf style definition is the template store for RTF output.

To view the actual style definition, double-click on a style name. The style definition 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 on using PROC TEMPLATE to modify and create styles, see "TEMPLATE Procedure: Creating a Style Definition" 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 they control and affect them in only the styles where you change them. See "Style Elements For Use With Device-Based SAS/GRAPH Output" on page 146 for information on 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 145 and "The GraphColors Style Element" on page 144 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 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. See *SAS Output Delivery System: User's Guide* for information on changing SAS registry settings. (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 on the syntax used in the GraphFonts style element, see "Font Specifications In The GraphFonts Style Element" on page 146.)

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 than 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 144 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 10.5 on page 145 lists the style attributes that

are defined in the GraphColors style element and the graphical elements that they affect by default.

Table 10.5 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
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. (See "Device-Based Graphics And Template-Based Graphics" in Chapter 1, "Introduction to SAS/GRAPH Software".)

2 The gtext attribute does not affect text that is not rendered as part of the graph. See also "Controlling Titles and Footnotes with Java and ActiveX Devices in HTML Output" in Chapter 13, "Managing Your Graphics With ODS".

The GraphFonts Style Element

The GraphFonts style element specifies the fonts that are used for different categories of graphical elements. Table 10.6 on page 146 lists the style attributes that are defined in the GraphFonts style element and the graphical elements that they affect by default.

Table 10.6 GraphFonts Attributes For Device-Based Output

GraphFonts Attributes*	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

* 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. See *SAS Output Delivery System: User's Guide* for information on changing SAS registry settings.

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 san-serif font family. If SAS cannot find the specific fonts listed, then it looks for a font in the san-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 Language Reference: Concepts* for information on registering TrueType fonts. See *Base SAS Procedures Guide* for information on the FONTREG procedure. △

Style Elements For Use With Device-Based 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 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 142 and "Example: Modifying a Style Element" on page 142. For complete documentation on the style attributes that can be specified in each style element, see "Style Attributes and Their Values" in the section "TEMPLATE Procedure: Creating a Style Definition" in *SAS Output Delivery System: User's Guide*.

Table 10.7 Device-Based Graph Style Elements: General Graph Appearance

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
DropShadowStyle	Used with text types	Color	GraphColors("gshadow")
Graph	Graph size and outer border appearance	OutputWidth	Not set
		OutputHeight	Not set
		BorderColor	Inherited
		BorderWidth	Inherited
		CellPadding	0
		CellSpacing	Inherited
GraphAxisLines	X, Y, and Z axis lines	Color	GraphColors("gaxis")
		LineStyle	1
		LineThickness	1px
GraphBackground	Background of the graph	Transparency	Not set
		BackgroundColor	Colors("docbg")
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackgroundImage	Not set
		Image	Not set
		VerticalAlign	Not set
		TextAlign	Not set
GraphBorderLines	Border around graph wall, legend border, borders to complete axis frame	Color	GraphColors("gborderlines")
		LineThickness	1px
		LineStyle	1
GraphCharts	All charts within the graph	Transparency	Not set
		BackgroundColor	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackgroundImage	Not set
		Image	Not set
		VerticalAlign	Not set
		TextAlign	Not set

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphDataText	Text font and color for point and line labels	Font or *font-attributes**	GraphFonts("GraphDataFont")
		Color	Not set
			GraphColors("gtext")
GraphFloor	3D floor	BackgroundColor	GraphColors("gfloor")
		Transparency	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackgroundImage	Not set
		Image	Not set
		VerticalAlign	Not set
		TextAlign	Not set
GraphFootnoteText	Text font and color for footnotes	Font or *font-attributes**	GraphFonts("GraphFootnoteFont")
		Color	Not set
			GraphColors("gtext")
GraphGridLines	Horizontal and vertical grid lines drawn at major tick marks	Color	GraphColors("ggrid")
		LineStyle	1
		LineThickness	1px
		Transparency	.5
		displayopts	"Auto"
GraphGridLines	Horizontal and vertical grid lines drawn at major tick marks	Color	GraphColors("ggrid")
		LineStyle	1
		LineThickness	1px
		Transparency	.5
		displayopts	"Auto"
GraphLegendBackground	Background color of the legend	Color	Colors("glegend")
		Transparency	Not set
GraphOutlines	Outline properties for fill areas such as bars, pie slices, and box plots.	Color	GraphColors("goutlines")
		LineStyle	1
		LineThickness	1px
GraphTitleText	Text font and color for titles	Font or *font-attributes**	GraphFonts("GraphTitleFont")
		Color	Not set
			GraphColors("gtext")

Style Element	Portion of Graph Affected	Recognized Attributes	Attribute Values in DEFAULT Style
GraphValueText	Text font and color for axis tick values and legend values	Font or *font-attributes**	GraphFonts("GraphValueFont")
		Color	Not set
			GraphColors("gtext")
GraphWalls	Vertical walls bounded by axes	Transparency	Not set
		BackgroundColor	GraphColors("gwalls")
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackgroundImage	Not set
		Image	Not set

* *Font-attributes* can be one of the following: FONTFAMILY=, FONTSIZE=, FONTSTYLE=, FONTWEIGHT=.

Table 10.8 Style Elements Affecting Device-Based Non-Grouped Graphical Data Representation

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
ThreeColorAltRamp	Line contours, markers, and data labels with segmented range color response	StartColor	GraphColors("gconramp3start")
		NeutralColor	GraphColors("gconramp3cneutral")
		EndColor	GraphColors("gconramp3end")
ThreeColorRamp	Gradient contours, surfaces, markers, nad data labels with continuous color response	StartColor	GraphColors("gramp3cstart")
		NeutralColor	GraphColors("gramp3cneutral")
		EndColor	GraphColors("gramp3cend")
TwoColorAltRamp	Line contours, markers, and data labels with segmented range color response	StartColor	GraphColors("gconramp2cstart")
		EndColor	GraphColors("gconramp2cend")
TwoColorRamp	Gradient contours, surfaces, markers, and data labels with continuous color response	StartColor	GraphColors("gramp2cstart")
		EndColor	GraphColors("gramp2cend")

Table 10.9 Style Elements Affecting Device-Based Grouped Graphical Data Representation

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
GraphData1	Primitives related to 1st grouped data items. Color applies to filled areas. ContrastColor applies to markers and lines.	Color	GraphColors("gdata1")
		ContrastColor	GraphColors("gcdata1")
		MarkerSymbol	"Circle"
		LineStyle	1
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData2	Primitives related to 2nd grouped data items	Color	GraphColors("gdata2")
		ContrastColor	GraphColors("gcdata2")
		MarkerSymbol	"Plus"
		LineStyle	4
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData3	Primitives related to 3rd grouped data items	Color	GraphColors("gdata3")
		ContrastColor	GraphColors("gcdata3")
		MarkerSymbol	"X"
		LineStyle	8
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
GraphData4	Primitives related to 4th grouped data items	Color	GraphColors("gdata4")
		ContrastColor	GraphColors("gcdata4")
		MarkerSymbol	"triangle"
		LineStyle	5
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData5	Primitives related to 5th grouped data items	Color	GraphColors("gdata5")
		ContrastColor	GraphColors("gcdata5")
		MarkerSymbol	"square"
		LineStyle	14
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData6	Primitives related to 6th grouped data items	Color	GraphColors("gdata6")
		ContrastColor	GraphColors("gcdata6")
		MarkerSymbol	"Asterisk"
		LineStyle	26
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
GraphData7	Primitives related to 7th grouped data items	Color	GraphColors("gdata7")
		ContrastColor	GraphColors("gcdata7")
		MarkerSymbol	"Diamond"
		LineStyle	15
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData8	Primitives related to 8th grouped data items	Color	GraphColors("gdata8")
		ContrastColor	GraphColors("gcdata8")
		MarkerSymbol	Not set
		LineStyle	20
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData9	Primitives related to 9th grouped data items	Color	GraphColors("gdata9")
		ContrastColor	GraphColors("gcdata9")
		MarkerSymbol	Not set
		LineStyle	41
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set

Style Element	Portion of Graph Affected	Default Attributes	Attribute Values in DEFAULT Style
GraphData10	Primitives related to 10th grouped data items	Color	GraphColors("gdata10")
		ContrastColor	GraphColors("gcdata10")
		MarkerSymbol	Not set
		LineStyle	42
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData11	Primitives related to 11th grouped data items	Color	GraphColors("gdata11")
		ContrastColor	GraphColors("gcdata11")
		MarkerSymbol	Not set
		LineStyle	2
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set
GraphData12	Primitives related to 12th grouped data items	Color	GraphColors("gdata12")
		ContrastColor	GraphColors("gcdata12")
		MarkerSymbol	Not set
		LineStyle	Not set
		MarkerSize	Not set
		LineThickness	Not set
		Gradient_Direction	Not set
		StartColor	Not set
		EndColor	Not set
		BackGroundImage	Not set
		Image	Not set

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 on **Gstyle**, select **Modify value**, and select *0=False* as the new value.

Changing the Appearance of Output to Match That of Earlier SAS Releases

SAS/GRAPH 9.2 introduces many new features that significantly change the default appearance of your SAS/GRAPH output. To produce output that looks as if it was 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 153.
- 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 native SAS/GRAPH devices (see "Device Categories And Modifying Default Output Attributes" on page 72). For example, this option works for GIF, TIFFP, JPEG, and ZPNG devices, but it is not applicable to PNG, SVG, or SASPRT* devices.
- Specify DEVICE=ZGIF on the GOPTIONS statement when you are sending output to the HTML destination.
- In other cases where your application specifies a device, specify a compatible Z device driver, if applicable. See "Devices" on page xvii for more information.

CHAPTER

11

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.

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 on these fonts, see Appendix 2, "Using SAS/GRAPH Fonts," on page 1643.

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 That Are Supplied by SAS" on page 156 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 That Are Supplied by SAS

SAS/GRAPH installs and registers a set of TrueType fonts, that are referred to collectively as system fonts. TrueType fonts that are shipped with SAS are listed in Table 11.1 on page 156.

You can use these fonts in your SAS programs by assigning the font name to font options, enclosed in quotes. For example, you can specify the following:

```
goptions ftext="Thorndale AMT";
```

Table 11.1 TrueType Fonts Supplied by SAS

Albany AMT*	Thorndale Duospace WT SC	GungsuhChe
Cumberland AMT*	Thorndale Duospace WT TC	Dotum
Thorndale AMT*	Arial Symbol*	DotumChe
Symbol MT	Times New Roman Symbol*	Gulim
Monotype Sorts	MS PMincho	GulimChe
Monotype Sans WT J	MS Mincho	NSimSun
Monotype Sans WT K	MS PGothic	SimHei
Monotype Sans WT SC	MS UI Gothic	SimSun
Monotype Sans WT TC	Batang	PMingLiU
Thorndale Duospace WT J	BatangChe	MingLiU
Thorndale Duospace WT K	Gungsuh	HeiT

* Albany AMT, Cumberland AMT, Thorndale AMT, Arial Symbol, and Times New Roman Symbol are font families. Normal, bold, italic, and bold italic versions of these fonts are provided.

For more information about using TrueType fonts with SAS/GRAPH, see *SAS Language Reference: Concepts*.

Determining What Fonts Are Available

The fonts listed in Table 11.1 on page 156 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 on determining what fonts are available, see *SAS Language Reference: Concepts* 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 2, "Using SAS/GRAPH Fonts," on page 1643 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 is installed with SAS (see Table 11.1 on page 156) 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 158. 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 143 and "The GraphFonts Style Element" on page 145 for more information about the font specifications in the styles. See "Style Attributes Versus Device Entry Parameters" on page 134 for more information on the GSTYLE system option.

Table 11.2 on page 158 shows the fonts used when styles are not active (the NOGSTYLE system option is in effect).

Table 11.2 Fonts Used By Default When NOGSTYLE System Option Is In Effect

Device	TITLE1	All Other Text
GIF, JPEG, PNG, TIFF, SVG, SASBMP, SASPRTx, GIFANIM, PCL, PS, PDF	Swiss	Font specified by **<MTmonospace>** subkey in the SAS registry
BMP, ZGIF, ZPNG, ZJPEG, ZTIFF in all environments except z/OS EMF and WMF on Windows Display devices for Graph window: WIN, XCOLOR, IBMPCGX	Swiss	Font specified by DMSFont
SASEMF, SASWMF, EMF and WMF in other operating environments	Swiss	Font specified by **<MTmonospace>** subkey in the SAS registry
JAVAMETA	Swiss	Font specified in the **Chartype 1** field in the device entry
CGM, ZPCL, ZPS, ZPDF	Swiss	Font specified by the device entry

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 1652 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 JAVA or 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 159 and "Methods For Specifying Fonts" on page 163 for information on 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 "The SAS Registry" in *SAS Language Reference: Concepts*.

Specifying a Font

To specify a font in your SAS program, include a font name, enclosed in quotes, 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 159 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 JAVA or JAVAIMG devices. △

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 will be used for the title.

The SAS registry is localized. If you specify a font using a registry subkey, the actual font that is used will be the localized value specified in your registry.

See also "Viewing Font Specifications in the SAS Registry" on page 158 and "Font Specifications In The GraphFonts Style Element" on page 146.

Specifying International Characters (Unicode Encoding)

You can use the **/unicode** modifier with a hexadecimal code to print any character in a font. This modifier can be used only with fonts that support Unicode encoding. Most of the TrueType fonts listed in Table 11.1 on page 156 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 160) 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 Web site at **http://www.unicode.org/charts**. See also *SAS Language Reference: Concepts* for information on printing international characters.

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 typed 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 159 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 that 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 11.1 on page 161. 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 11.1 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;
```

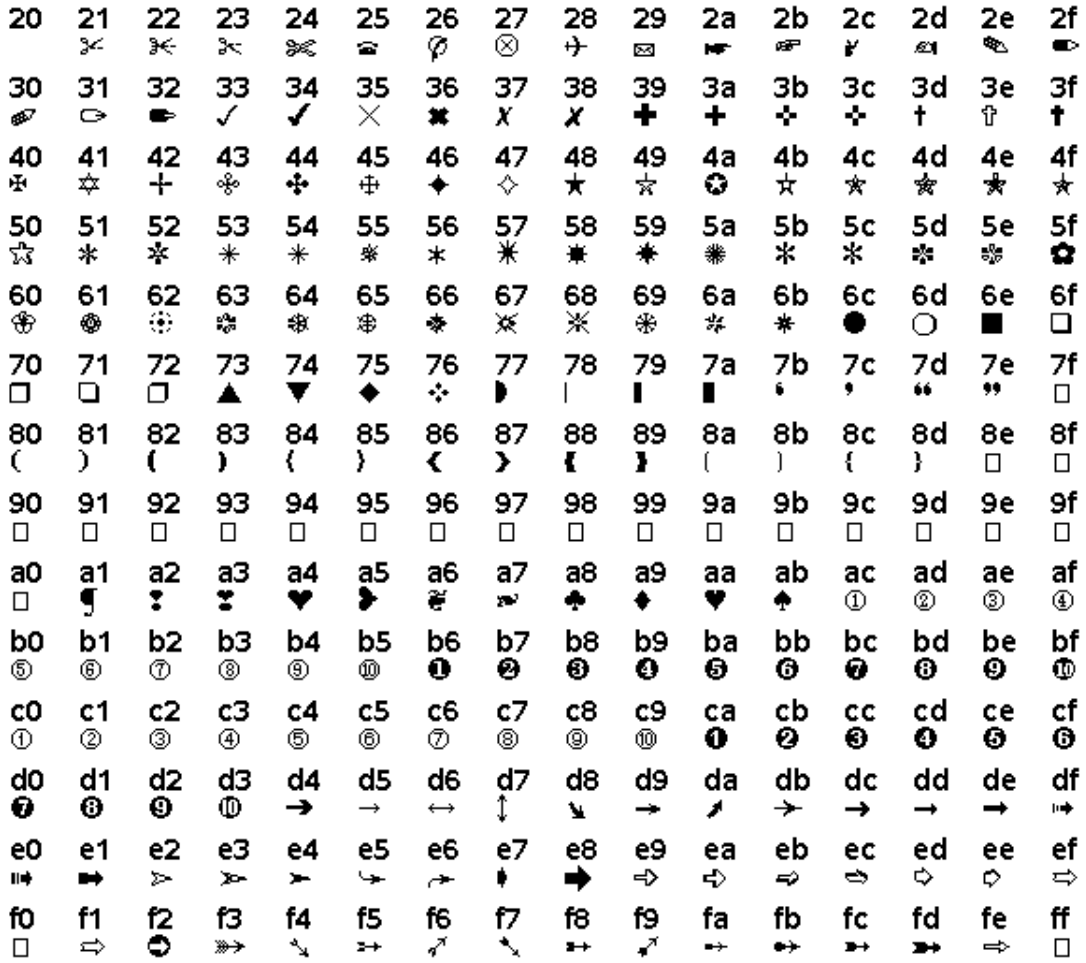
```
      x = 0;
   tempy = tempy - 6;
         end;
 run;

/*****************************************************/
/* 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 11.1 on page 162 and Figure 11.2 on page 163 show the output of the program above for the TrueType fonts Symbol MT and Monotype Sorts.

Figure 11.1 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 11.2 Monotype Sorts Font



Methods For Specifying Fonts

In general, there are four ways to specify fonts. The method 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 on 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 164.

- 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 164.

- 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 165.

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 on page 1116 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 on each of the global statements, see Chapter 14, "SAS/GRAPH Statements," on page 197.

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 on the GOPTIONS statement, see "GOPTIONS Statement" on page 220. Information for specific graphics options is in Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327. △

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 with Java and ActiveX Devices in HTML Output" on page 194 for information on 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 only want to change the fonts used in a few applications, then using the GOPTIONS statement is a better alternative.

For information on changing the font specifications used by the styles, see "Ways to Modify Graph Fonts Or Colors Specified By Styles" on page 143.

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 157

.

CHAPTER

12

SAS/GRAPH Colors and Images

Using SAS/GRAPH Colors and Images

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 134 and "Turning Off Styles" on page 153 for more information.

In either case (using ODS styles or using device parameters), you can override the default colors by specifying options in your SAS/GRAPH program. Whether you are

using ODS styles or you have turned styles off, you can the change these colors as described in “Specifying Colors in SAS/GRAPH Programs” on page 168. You can add images to your output as described in “Specifying Images in SAS/GRAPH Programs” on page 181.

Specifying Colors in SAS/GRAPH Programs

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 Chapter 14, “SAS/GRAPH Statements,” on page 197.
- use options in the GOPTIONS statement that define colors for specific graphics elements. See “Specifying Colors in a GOPTIONS Statement” on page 168.
- define a color list with the GOPTIONS COLORS= option. See “COLORS” on page 340
- specify a different style, modify an existing style, or create a custom style. See Chapter 10, “Controlling The Appearance of Your Graphs,” on page 133 for more information on 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 169 and Chapter 38, “The GDEVICE Procedure,” on page 1125 for more information.

See “Precedence of Appearance Option Specifications” on page 141 for information on 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	Sets the color for
CBACK=	background for graphics output
CBY=	BY lines in graphics output
CPATTERN=	fill patterns
CSYMBOL=	SYMBOL definitions
CTEXT=	all text and the border in graphics output
CTITLE=	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 Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for complete information about each of these graphics options.

Defining and Using a Color List

Introduction to the 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 169.

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 169 for more information.

The color selected from a color list varies depending on the procedure using the color and graphical element it's 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, "The GDEVICE Procedure," on page 1125.

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 170. 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. Any time 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" on page 340 for more information.

Color-Naming Schemes

Introduction to Color-Naming Schemes

The valid color-naming schemes are as follows:

- RGB (red green blue)
- 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 12.1 on page 170 shows examples of each color-naming scheme.

Table 12.1 Examples of Specifying Colors

Color-Naming Scheme	Example
RGB	`COLORS=(cx98FB98 cxDDA0DD cxFFDAB9 cxDB7093 cxB0E0E6)`
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")`

You can also mix color-naming schemes in the same statement, for example:

```
goptions colors=(cxEE0044 "vivid blue" darkgreen);
```

Note: 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 different 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 supported by SAS/GRAPH has its advantages and disadvantages based on how the output is used. For example, if you are creating a

report that will be viewed online only, 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.

The color utility macros enable you to create colors for a specific color-naming scheme. These macros convert color values between color-naming schemes. See "Using the Color Utility Macros" on page 177.

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. △

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. With RGB color codes, a color is defined by its red, green, and blue components. Individual amounts of each color are added together to create the desired color. 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 given as hexadecimal numbers in the range 00 through FF (0% to 100%), where lower values are darker and higher values are lighter. This scheme allows for up to 256 levels of each color component (over 16 million different colors).

Table 12.2 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 "Using the SAS Registry to Control Color" in *SAS Language Reference: Concepts* for information on viewing the RGB combinations that match predefined SAS color names.

CMYK Color Codes

CMYK is a color-naming scheme 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 Look-Up Table, you can use the CMYK color-naming scheme. Specify colors in terms of their cyan, magenta, yellow, and black components. Color names are of the form *ccmmyykk*, where the following is true:

- *cc* is the cyan component.
- *mm* is the magenta component.
- *yy* is the yellow component.
- *kk* is the black component.

The components are given as hexadecimal numbers in the range 00 through FF, where higher values are darker and lower values are brighter. This scheme allows for up to 256 levels of each color component. Quotation marks are required when the color value starts with a number instead of a letter.

Table 12.3 Examples of CMYK Color Values

Color	CMYK Value
red	00FFFF00
green	FF00FF00
blue	FFFF0000
white	00000000
process black (using cyan, magenta, and yellow ink)	FFFFFF00
pure black (using only black ink)	000000FF

Note: You can specify a CMY value by making the *kk*, the color's black component, zero (00). △

CMYK color specifications are for devices that support four colors. If a CMYK color is used on a three-color device, the device processes the color specification. The resulting colors might not be as expected. Different CMYK colors might map to the same device color because a four-color space supports more colors than a three-color space.

HLS Color Codes

The HLS color-naming scheme follows the Tektronix Color Standard illustrated in Figure 12.1 on page 174. To make the HLS color model consistent with the HSV coordinate system, Tektronix places blue at zero degrees. With the HLS color naming-scheme, you specify colors in terms of hue, lightness, and saturation levels. HLS color names are of the form H*hhhllss*, where the following is true:

- H indicates that this is an HLS color specification.
- *hhh* is the hue component.
- *ll* is the lightness component.
- *ss* is the saturation component.

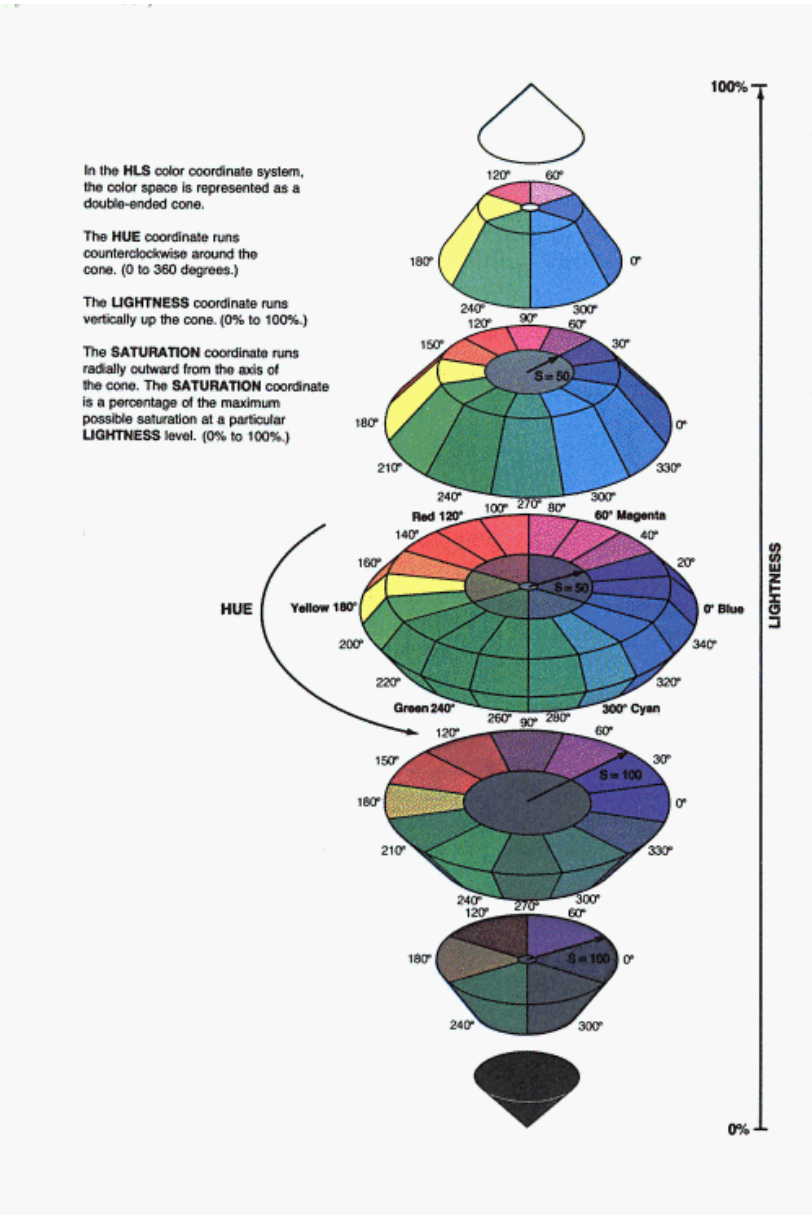
The components are given as hexadecimal numbers. The hue component has the range of 000 through 168 hexadecimal (168 hexadecimal is equivalent to 360 decimal). Both the lightness and saturation components are hexadecimal and scaled to a range of 0 to 255 expressed with values of 00 through FF (0% to 100%). Thus, they provide 256 levels for each component.

Table 12.4 Examples of HLS Color Codes

Color	HLS Color Code
red	H07880FF
green	H0F080FF
blue	H00080FF
light gray	H000BB00
white*	H*xxx*FF00, such as H000FF00
black*	H*xxx*0000 such as H0000000

* 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.

Figure 12.1 Tektronix Color Standard



HSV (or HSB) Color Codes

Specify the HSV color-naming scheme 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 that this is an HSV color specification.
- *hhh* is the hue component.
- *ss* is the saturation component.
- *vv* is value or brightness component.

The components are given as hexadecimal numbers. The hue component has the range of 000 through 168 hexadecimal (168 hexadecimal is equivalent to 360 decimal). Both the saturation and value (brightness) components are hexadecimal, scaled to a

range of 0 to 255, and expressed with values of 00 through FF. Thus, they provide 256 levels for each component.

Table 12.5 Examples of HSV (or HSB) Color Codes

Color	HSV Color Code
red	V000FFFF
green	V078FFFF
blue	V0F0FFFF
light gray*	V*xxx*00BB such as V07900BB
white*	V*xxx*00FF such as V07900FF
black*	V*xxx*00000 such as V0790000

* 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* can be any hue.

Gray-Scale Color Codes

Specify the lightness or darkness of gray using the word GRAY and a lightness value. Gray-scale color codes are of the form GRAY*ll*. The value *ll* is the lightness of the gray and is given as a hexadecimal number in the range 00 through FF. This scheme allows for 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 175). Its value is CX808080. Invalid color specifications are mapped to GRAY. △

Table 12.6 Examples of Gray-Scale Color Codes

Color	Gray-Scale Color Codes	RGB equivalent
white	GRAYFF	CXFFFFFF
light gray	GRAYC0	CXC0C0C0
dark gray	GRAY40	CX404040
black	GRAY00	CX000000

SAS Color Names and RGB Values in the SAS Registry

SAS provides, in the SAS Registry, 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.

You can also create your own color values by adding them to the registry. For more information on viewing and modifying the list of color names, see "Using the SAS Registry to Control Color" in *SAS Language Reference: Concepts*.

Color Naming System Values

With CNS, you specify a color value by specifying lightness, saturation, and hue, in that order, using the terms shown in the following table.

Table 12.7 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:

- The lightness values black and white should not be used with saturation or hue values.
- If not specified, 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 you want is black, white, or some form of gray, you must specify at least one hue.

One or two hue values can be used in the CNS color name. When using 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*. When two hues are used, the resulting color 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. If you are using a color with an **`ish`** suffix, this color must precede the color without the **`ish`** suffix.

Color names can be written 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 are all valid color specifications:

- 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. To use a CNS color value when the color name is also in the SAS Registry, do the following:

- Include a space to separate the words.
- Enclose the entire color name in quotation marks.

△

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.

The %COLORMAC macro contains several subcomponent macros that can be used to construct and convert color values for the different color-naming schemes supported by SAS. The %HELPCLR macro provides information about the %COLORMAC subcomponent macros. The following table shows information displayed in your SAS log when you call the %HELPCLR macro from the command line.

Table 12.8 Using the %HELPCLR macro

Use...	To...
%HELPCLR;	List the color utility macro names with help information.
%HELPCLR(ALL);	Display the short descriptions and examples for each of the color utility macros.
%HELPCLR(*macroname*);	Obtain a short description and an example of a specific color utilities macro. Replace *macroname* with the name of the color utility macro you are interested in.

When the color utility macros are invoked, the calculated color value is directed to the SAS log. The calculated color can also be used to perform in-place substitutions in the code.

Table 12.9 %CMY(*cyan, magenta, yellow*);

Description	Usage Example

Replace *cyan, magenta, yellow* with numeric values to create an RGB color value. The numeric values that are used in place of *cyan, magenta, yellow* indicate the percentage of each color to be included in the RGB value.	Entering the following code into your Program Editor:
	`%COLORMAC;`
	`data _null_;`
	`put "%CMY(100,0,100)";`
	`run;`
	Returns the RGB value CX00FF00 which is green.

Table 12.10 %CMYK(*cyan, magenta, yellow, black*);

Description	Usage Example

Replace *cyan, magenta, yellow, black* with numeric values to create a CMYK color value. The numeric values that are used in place of *cyan, magenta, yellow, black* indicate the percentage of each color to include in the CMYK color value. See "CMYK Color Codes" on page 171 for more information on the color value produced by using this macro.	Entering the following code into your Program Editor:
	`%COLORMAC;`
	`data _null_;`
	`put %CMYK(0,46,16,31);`
	`run;`
	Returns the CMYK value 0075294F which is purple.

Note: In the PUT statement, %CMYK(*cyan, magenta, yellow, black*), should not be placed in quotations. △

Table 12.11 %CNS (*colorname*);

Description	Usage Example

Replace *colorname* with a color-naming scheme color name to create an HLS color value. See "HLS Color Codes" on page 172 for more information on HLS color values. For more information on valid color-naming scheme color names see "Color Naming System Values" on page 176 or enter the following into the command-line of the Program Editor:
`%HELPCLR(CNS);`
	Entering the following code into your Program Editor:
	`%COLORMAC;`
	`data _null_;`
	`put "%CNS(GRAYISH REDDISH PURPLE)";`
	`run;`
	Returns the HLS value H04B8040 which is grayish reddish purple.

Note: The %CNS macro accepts only CNS color names where a space is used to separate the words in the color name. △

Table 12.12 %HLS(*hue, lightness, saturation*);

Description	Usage Example

Replace *hue, lightness, saturation* with numeric values to create an HLS color value. *Hue* should be replaced with any value from 0 to 360. *Lightness* and *saturation* indicate a percentage to be included in the HLS color values. See "HLS Color Codes" on page 172 for more information.	Entering the following code into your Program Editor:
	`%COLORMAC;`
	`data _null_;`
	`put "%HLS(0,50,100)";`
	`run;`
	Returns the HLS value H00080FF which is blue.

Table 12.13 %HSV(*hue, saturation, value*);

Description	Usage Example

Replace *hue, saturation, value* with numeric values to create an HLS value from HSV components. *Hue* should be replaced with any value from 0 to 360. *Saturation* and *value* (brightness) indicate a percentage to be included in the HLS color value. See "HSV (or HSB) Color Codes" on page 174 and "HLS Color Codes" on page 172 for more information.	Entering the following code into your Program Editor:
	`%COLORMAC;`
	`data _null_;`
	`put "%HSV(0,100,75)";`
	`run;`
	Returns the HSV value V000FFBF which is dark red.

Table 12.14 %RGB(*red, green, blue*);

Description	Usage Example

Replace *red, green, blue* with numeric values to create an RGB color value from RGB color components. The numeric values that are used in place of *red, green, blue* indicate the percentage of each color to be included in the RGB color value. See "RGB Color Codes" on page 171 for more information.	Entering the following code into your Program Editor:
	`%COLORMAC;`
	`data _null_;`
	`put "%RGB(100,100,0)";`
	`run;`
	Returns the RGB value CXFFFF00 which is yellow.

Table 12.15 %HLS2RGB(*hls*);

Description	Usage Example

Replace *hls* with an HLS color value to create an RGB color value. See "HLS Color Codes" on page 172 and "RGB Color Codes" on page 171 for more information.	Entering the following code into your Program Editor:
	`%COLORMAC;`
	`data _null_;`
	`put "%HLS2RGB(H04B8040)";`
	`run;`
	Returns the RGB value CX9F5F8F which is grayish reddish purple.

Table 12.16 %RGB2HLS(*rgb*);

Description	Usage Example

Replace *rgb* with an RGB color value to create an HLS color value. See "RGB Color Codes" on page 171 and "HLS Color Codes" on page 172 for more information.	Entering the following code into your Program Editor:
	`%COLORMAC;`
	`data _null_;`
	`put "%RGB2HLS(CX9F5F8F)";`
	`run;`
	Returns the HLS value H04C7F40 which is grayish reddish purple.

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. △

For additional information on color-naming schemes. See *Effective Color Displays: Theory and Practice* by David Travis and *Computer Graphics: Principles and Practice* by Foley, van Dam, Feiner, and Hughes.

Processing Limitations For Colors

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 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.

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" on page 339) 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 display as the color you specify.

You can use the TARGETDEVICE= graphics option to preview the way 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" on page 424 for complete information about the TARGETDEVICE= graphics option. △

Specifying Images in SAS/GRAPH Programs

SAS/GRAPH enables you to display images as part of your graph. You can place an image in the background area of a graph, in the backplane of graphs that support frames, or on the bars of two-dimensional bar charts. You can also apply images at specified graph-coordinate positions using the Annotate facility or the DATA Step Graphics Interface (DSGI).

The images 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.

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.

File Type	Description
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.
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:

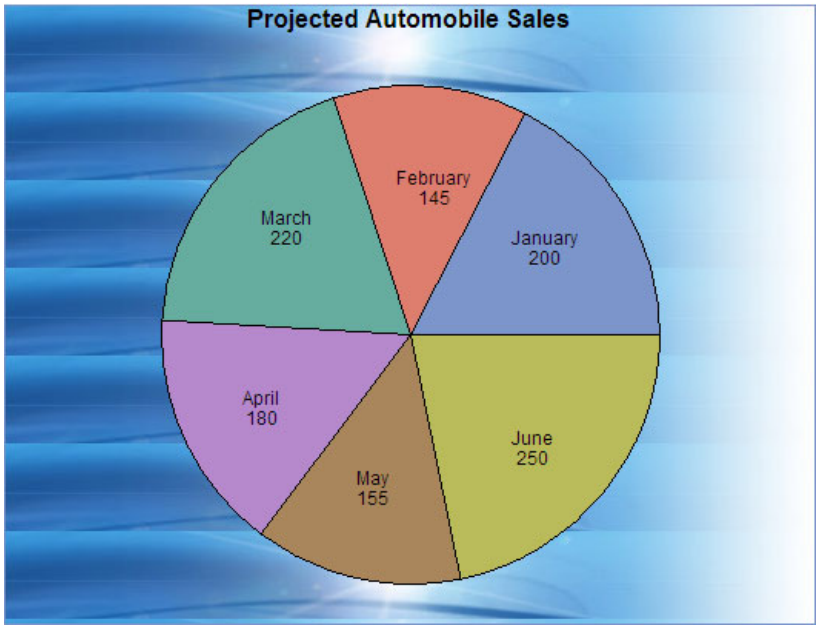
```
goptions iback="external-image-file" | fileref;
```

For example, the following program creates a pie chart with a background image:

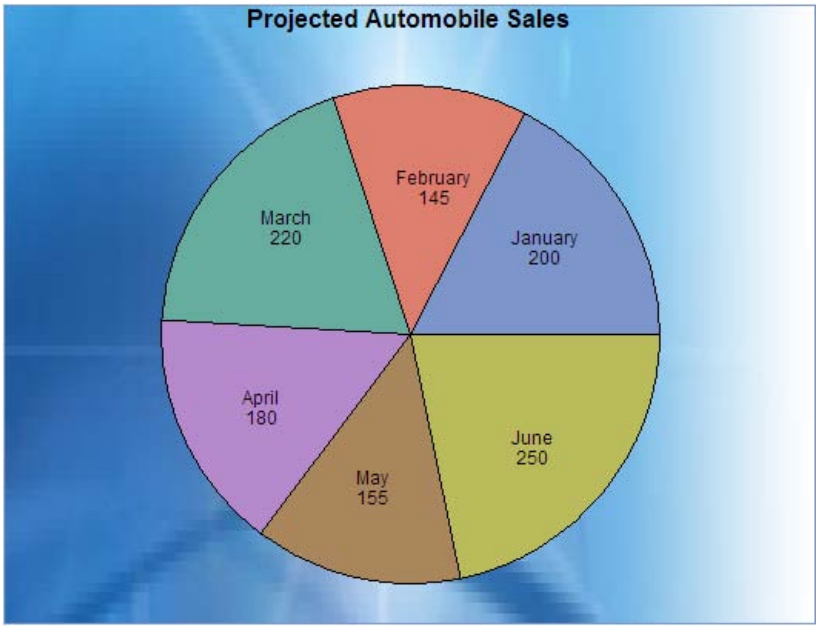
```
goptions reset=all
          htitle=1.25
         colors=(cx7c95ca cxde7d6f  cx66ada0
                 cxb689cd cxa9865b cxbabc5c)
         iback="external-image-file";
title "Projected Automobile Sales";
data sales;
    input Month Amount;
    informat month monyy.;
   datalines;
jan08 200
feb08 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:

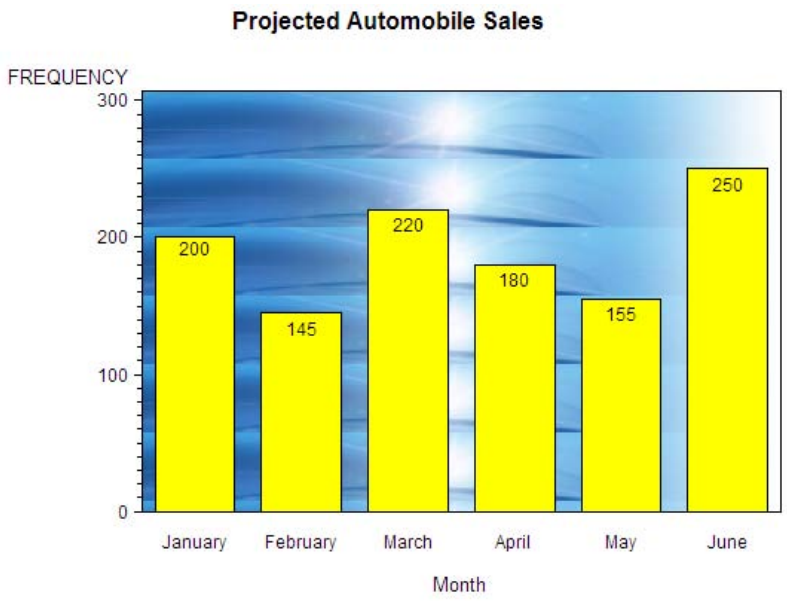
```
iframe=fileref | "external-image-file";
```

For example, the following program creates a vertical bar chart and adds an image to the graph frame:

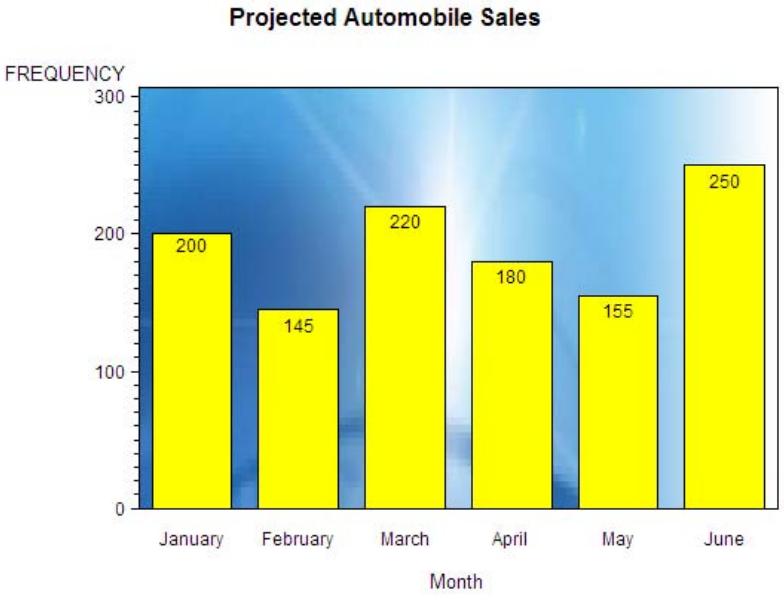
```
goptions reset=all htitle=1.25 colors=(yellow cxde7d6f);
title "Projected Automobile Sales";
data sales;
    input Month Amount;
    informat month monyy.;
   datalines;
jan08 200
feb08 145
mar08 220
apr08 180
may08 155
jun08 250
;
proc sort;
   by month;
proc gchart;
   format month monname8.;
   vbar month / discrete freq=amount inside=freq
               coutline=black  iframe="external-image-file";
run;
```

```
quit;
```

Because the default value for the IMAGESTYLE= graphics 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 in three-dimensional bar charts if you are using the ACTIVEX device.

On 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;
```

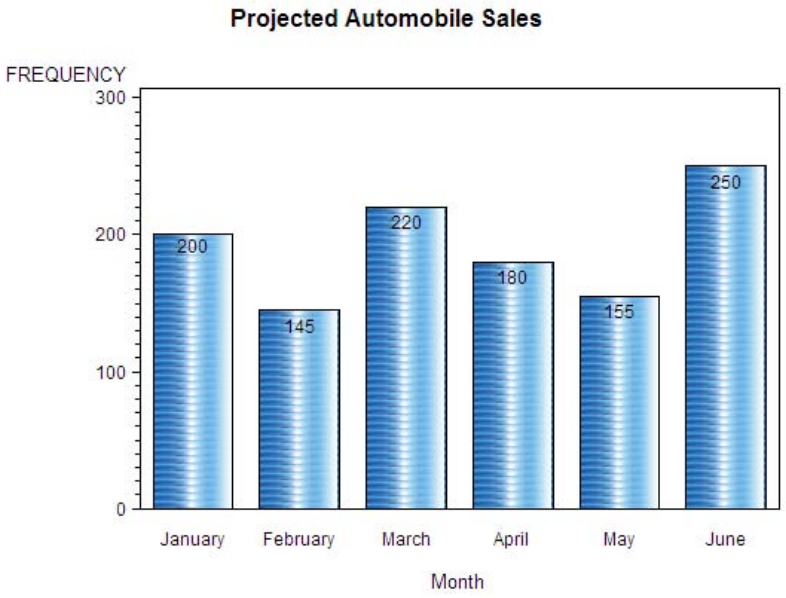
To tile subsequent images, reset the PATTERN statement or by specify IMAGESTYLE=TILE.

Note: Images are supported on bar charts generated by the HBAR and VBAR statements. If an image is specified on 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:

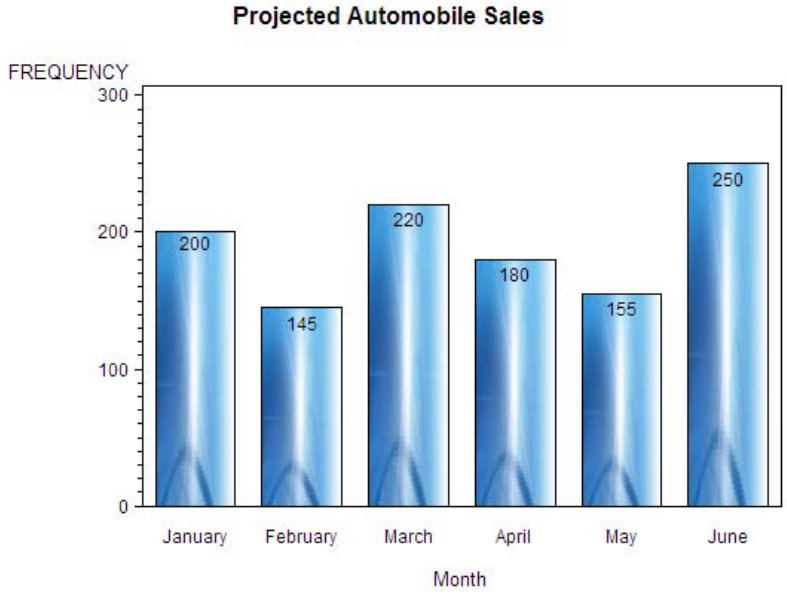
```
goptions reset=all htitle=1.25 colors=(yellow cxde7d6f);
title "Projected Automobile Sales";
data sales;
    input Month Amount;
    informat month monyy.;
   datalines;
jan08 200
feb08 145
mar08 220
apr08 180
may08 155
jun08 250
;
proc sort;
   by month;
pattern1 image="external-image-file";
proc 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

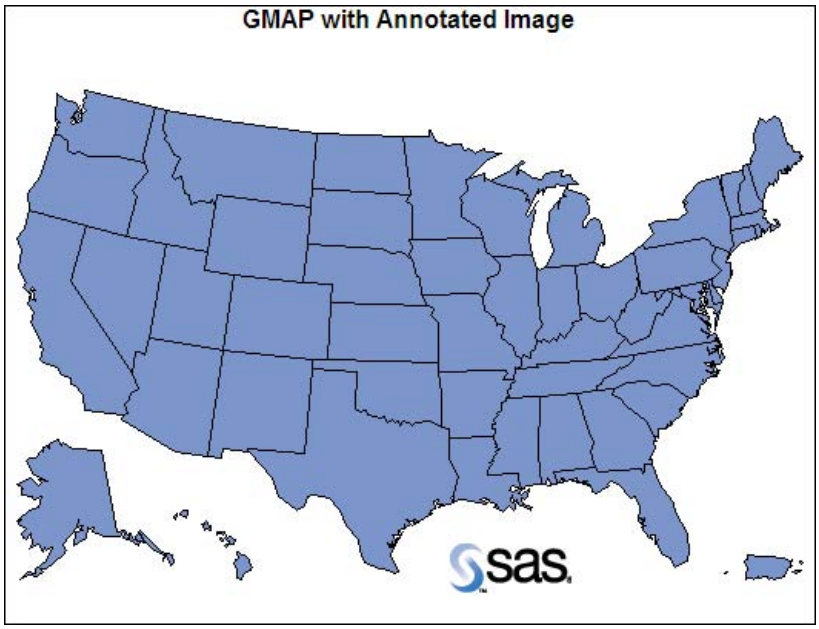
The Annotate facility enables you to display an image at the coordinate location that you specify with the X and Y variables. To display an image, do the following:

- 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.

```
goptions reset=all border htitle=1.25
    hsize=5.5in vsize=4.2in;
data my_anno;
   length function $8;
   xsys="3";  ysys="3"; when="a";
function="move";  x=55;  y=55;  output;
function="image";  style="fit";  imgpath="external-image-file";
         x=x+15;  y=y+18;  output;
run;
title1 "GMAP with Annotated Image";
proc gmap data=maps.us map=maps.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, set the STYLE variable equal to **"tile"**.

Displaying Images using DSGI

Using the DATA Step Graphics Interface (DSGI), you can display an image in a designated position. To display an image, specify the file specification for the image file in quotation marks on the GDRAW('IMAGE',...) function.

This code displays the image in the screen coordinates (20, 20) to (40, 40). The last parameter, FIT, indicates how to display the image.

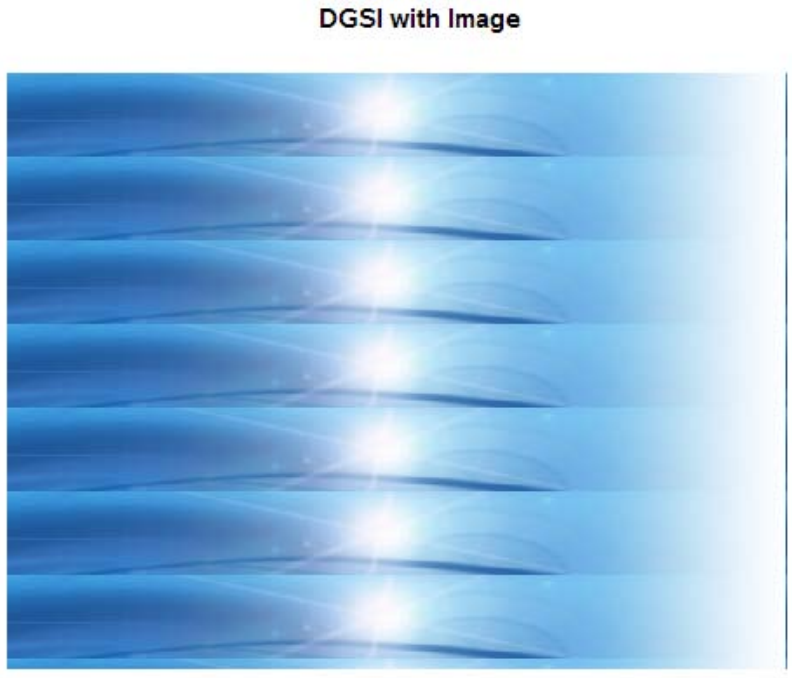
```
rc=gdraw("image", "external-image-file", 20, 20, 40, 40, "fit");
```

"Image File Types Supported by SAS/GRAPH" on page 181 shows the supported image file formats.

```
goptions reset=all
   ftext="Albany AMT/bold" htitle=1.25
   hsize=5.5in vsize=4.2in;
title "DSGI with Image";
data image;
   rc=ginit();
   rc=graph("clear");
   rc=gdraw("image","external-image-file",
            5, 5, 90, 90,"tile");
   rc=graph("update");
   rc=gterm();
run;
quit;
```

If you specify the TILE keyword for the GDRAW('IMAGE',...) function, the image is copied as many times as needed to fill the specified area.

```
rc=gdraw("image","external-image-file",
           5, 5, 90, 90,"tile");
```



If you specify the FIT keyword for the GDRAW('IMAGE',...) function, the image is stretched to fit within the entire area.

```
rc=gdraw("image","external-image-file",
           5, 5, 90, 90,"fit");
```

DGSI with Image

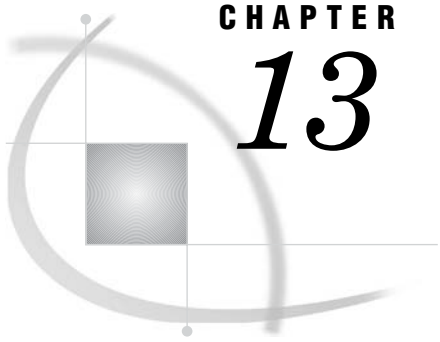
Disabling and Enabling Image Output

The NOIMAGEPRINT graphics option disables image output without removing code from your SAS/GRAPH program. It is useful for printing output without images.

```
goptions noimageprint;
```

To enable image output, reset the GOPTIONS statement or specify the IMAGEPRINT graphics option.

```
goptions imageprint;
```



CHAPTER

13

Managing Your Graphics With ODS

Introduction

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 will send output to each destination. The LISTING 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 on 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 13.1 Relevant Destination Table

Destinations	Results	Default Style	Default ImgFmt	Default DPI
DOCUMENT	ODS document	N/A	N/A	N/A
LISTING	SAS output listing	Listing	PNG	100
OUTPUT	SAS data set	N/A	N/A	N/A
HTML	HTML file for online viewing	Default	PNG	100
LATEX [1]	LaTeX file	Default	PostScript	200
PRINTER	printable output in one of three different formats: PCL, PDF, or PS (PostScript)	Printer for PDF and PS, monochromePrinter for PCL	Embedded PNG	150
RTF	output written in Rich Text Format for use with Microsoft Word 2000	RTF	Embedded PNG	200
Measured RTF		RTF	Embedded	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 LISTING destination is open. You can use an ODS destination statement, such as ODS HTML, to open a different destination. You can also specify options, such as the HTML filename or the path to an output directory, on the ODS destination statement.

ODS *destination <option(s)*;>

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, as well as specifying a different style, and specifying the appropriate image resolution in DPI for your output images. For example, the following ODS HTML statement:

- opens the HTML destination
- specifies that images be written to the directory **`C:\myfiles\images`**
- specifies that the path to the images is specified as **`http://www.sas.com/images/`*`image-filename`*** in the HTML file

- specifies that other output files (for example, the HTML file) be written to the directory **C:\myfiles**
- specifies that the name of the initial HTML file that is displayed is **barGraph.htm**
- changes the style to Analysis.

```
ods html path="c:\myfiles\"
         gpath="c:\myfiles\images" (url="http://www.sas.com/images/")
         body="barGraph.htm"
         style=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 SAS current folder is the same directory in which you start your SAS session. If you are running SAS with the windowing environment in the Windows operating system, then the current folder is displayed in the status bar at the bottom of the main SAS window.

If you do not specify a filename for your output, then SAS provides a default file that is determined by the ODS destination. This file is saved in the SAS current folder. You can check the SAS log to verify the name of the file in which your output is saved.

For complete documentation on ODS destinations, see *SAS Output Delivery System: User's Guide*.

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

GPATH= *location* (URL= '*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= *location* (URL= '*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 current working directory. For complete documentation on PATH= option, see the ODS LISTING statement, ODS MARKUP statement, or TAGSET.RTF statement 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 on using the STYLE= option with SAS/GRAPH output, see Chapter 10, "Controlling The Appearance of Your Graphs," on page 133.

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, and you are less likely to encounter unexpected results.

Controlling Titles and Footnotes with Java and ActiveX Devices in HTML Output

When you use ODS to send your graphs to an HTML destination, you can choose whether titles and footnotes are rendered as part of the HTML body file, as they are with tabular output, or the graphical image that appears in the Web page.

Where titles and footnotes are rendered determines how you control their font, size, and color.

Controlling Where Titles and Footnotes are Rendered

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

For the JAVA, JAVAIMG, ACTIVEX, and ACTXIMG device drivers, titles and footnotes are always rendered as part of the HTML body file. The GTITLE and GFOOTNOTE options are ignored for these drivers.

For all other devices, the GTITLE and GFOOTNOTE options determine where the titles and footnotes are rendered. The default settings, GTITLE and GFOOTNOTE, render titles and footnotes as part of the graphic image. If you want titles and footnotes to appear within the HTML body file and not as part of the graphical image, you must specify the NOGTITLE or NOGFOOTNOTE option, as in the following example.

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

If the title or footnote is being output through an ODS markup destination (such as HTML) and the corresponding ODS option NOGTITLE or NOGFOOTNOTE is specified, then the title or footnote is rendered in the body of the HTML file rather than in the graphic itself. Specifying NOGTITLE or NOGFOOTNOTE 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.

Controlling the Text Font, Size, and Color

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, then 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. SAS looks for global options such as CTEXT= and FTITLE= in the GOPTIONS statement. For more information, see "Using Graphics Options with ODS (USEGOPT)" on page 195.
3. SAS looks for information specified in the current style.

When titles and footnotes are rendered as part of the graphic image, SAS looks first for options on the TITLE and FOOTNOTE statement and then for options in the GOPTIONS statement. When titles and footnotes are rendered as part of the graphic image, you do not need to specify the ODS USEGOPT statement.

When titles and footnotes are rendered as part of the body of the HTML file, font sizes that are specified as a percentage are interpreted as a percentage of the size specified by the current style. When titles and footnotes are rendered as part of the image, fonts sizes that are specified as a percentage are interpreted as a percentage of graphics output area. For more information about specifying fonts and font sizes, refer to

- "FTEXT" on page 363 and "FTITLE" on page 363
- "HTEXT" on page 385 and "HTITLE" on page 385
- "GUNIT" on page 378
- "TITLE, FOOTNOTE, and NOTE Statements" on page 279.

Using Graphics Options with ODS (USEGOPT)

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, ODS does not recognize the settings for the following graphics options unless you also specify the ODS USEGOPT statement:

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

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 (ASTRONOMY). The title for the second graph uses the font size and color specified by the HTITLE and CTEXT options.

```
ods html file="myout.htm" style=astronomy;
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;
```

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.

CHAPTER

14

SAS/GRAPH Statements

Overview

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 Language Reference: Dictionary*.

In addition, SAS/GRAPH has its own set of statements that affect only graphics output generated by the SAS/GRAPH procedures and the graphics facilities Annotate and DSGI. Most of these statements are *global statements*. That is, they can be specified anywhere in your program and remain in effect until explicitly changed 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.

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, you can specify one of the device drivers GIF, ACTIVEX, or JAVA. ACTIVEX and JAVA are available only with GCHART, GCONTOUR, GMAP, GPLOT, and G3D. With the GIF device driver, the graphics output is stored in GIF files. With the ACTIVEX device driver, graphics output is stored as XML input to ActiveX controls. With the JAVA device driver, graphics output is stored as XML input to Java applets. 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 on the BY, LABEL, OPTIONS, and WHERE statements in Base SAS software, see *SAS Language Reference: Dictionary*.

AXIS Statement

Controls the location, values, and appearance of the axes in plots and charts.

Used by: GCHART, GBARLINE, GCONTOUR, GPLOT, GRADAR, G3D procedures

Restriction: For the G3D procedure, the AXIS statement is supported by the JAVA and ActiveX devices only.

Type: Global

Syntax

AXIS<1...99> <*options*>;

option(s) can be one or more options from any or all of the following categories:

- axis scale options:

  INTERVAL=EVEN | UNEVEN | PARTIAL

  LOGBASE=*base* | E | PI

  LOGSTYLE=EXPAND | POWER

  ORDER=(*value-list*)

- appearance options:

  COLOR=*axis-color*

  LENGTH=*axis-length <units >*

  NOBRACKETS

  NOPLANE

  OFFSET=(*<n1 ><,n2 >)<units >* | (*<n1<units>><,n2<units >>*)

  ORIGIN=(*<x><,y >)<units>* | (*<x<units >><,y<units>>*)

  STAGGER

  STYLE=*line-type*

  WIDTH=*thickness-factor*

- tick mark options:

  MAJOR=(*tick-mark-suboption(s))*| NONE

  MINOR=(*tick-mark-suboption(s))*| NONE

- text options:

  LABEL=(*text-argument(s))*| NONE

  REFLABEL=(*text-argument(s))*| NONE

  SPLIT="*split-char*"

  VALUE=(*text-argument(s))*| NONE

Description

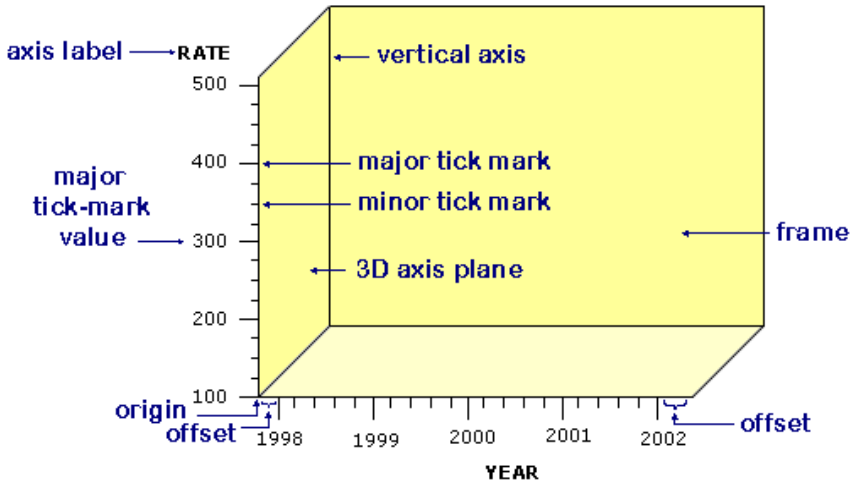
AXIS statements specify the following characteristics of an axis:

- the way 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 14.1 on page 200 illustrates the terms associated with the various parts of axes.

Figure 14.1 Parts of Axes



Options

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 searched for in this order:

1. The GUNIT= option in a GOPTIONS statement
2. the default unit, CELLS.

COLOR=*axis-color*
specifies the color for all axis components (the axis line, all tick marks, 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 14.1

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

Option	Graph Element	Style Reference
calling procedure:		
CTEXT=	all axis text (AXIS label and major tick mark value descriptions)	GraphLabelText
CAXIS=	axis line and major and 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. If the CTEXT= option is not used, the color of all axis components is the color of the default style.

Alias: C=

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294

INTERVAL=EVEN | UNEVEN | PARTIAL
The INTERVAL option affects the LOGBASE option in the AXIS statement. Specifying the option INTERVAL=UNEVEN and LOGBASE=10, permits non-base10 values to be specified for the ORDER option, while retaining a logarithmic scale for the axis.

Note: PARTIAL is an alias for UNEVEN. They have the same effect. △

Restriction: Not supported by Java and ActiveX

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 quotes. Separate multiple strings with blanks.

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=*text-color*

FONT=*font* | NONE

HEIGHT=*text-height <units >*

JUSTIFY=LEFT | CENTER | RIGHT

ROTATE=*degrees*

See "Text Description Suboptions" on page 210 for a complete description.

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

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294, "Example 2. Specifying Logarithmic Axes" on page 297 , and "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309

Restriction: Partially supported by Java and ActiveX.

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 may produce unexpected results.

This option is not supported by the GRADAR Procedure.

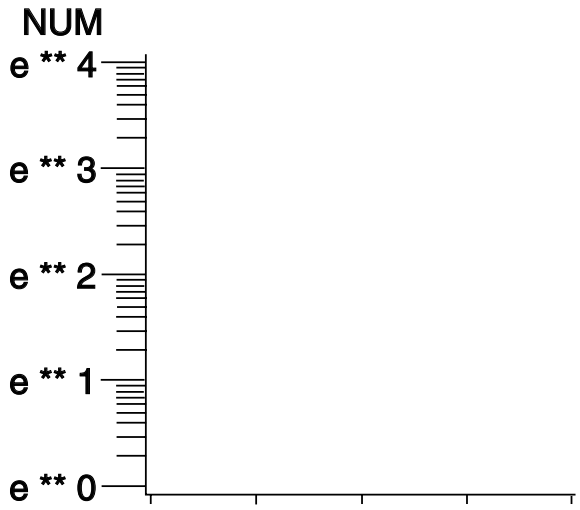
Style Reference: Color attribute of the GraphLabelText graph element.

Restriction: Not supported by Java.

Featured in: "Example 2. Specifying Logarithmic Axes" on page 297 and "Example 9. Combining Graphs and Reports in a Web Page" on page 315 .

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 14.2 on page 202.

Figure 14.2 Axis Generated with LOGBASE=10 and LOGSTYLE=EXPAND



This option is not supported by the GRADAR Procedure.

Featured in: "Example 2. Specifying Logarithmic Axes" on page 297

Restriction: Not supported by Java

LOGSTYLE=EXPAND | POWER
specifies whether the values displayed on the logarithmic axis are the values of the base or the values of the power. LOGSTYLE= is meaningful only when you use LOGBASE=.

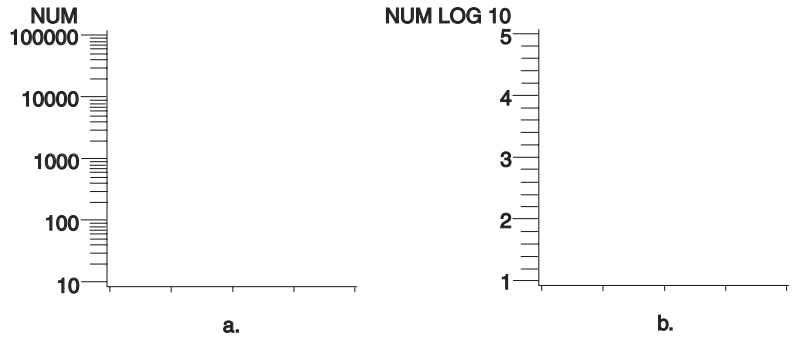
LOGSTYLE=EXPAND specifies that the values displayed are the values of the base raised to successive powers and that the minor tick marks are logarithmically placed. For example, if the base is 10, the values displayed are 10, 100, 1000, 10000, and so on. The default is LOGSTYLE=EXPAND. This statement generates an axis like the one in part (a) of Figure 14.3 on page 203:

```
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 14.3 on page 203:

```
axis logbase=10 logstyle=power;
```

Figure 14.3 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.

This option is not supported by the GRADAR Procedure.

Featured in: "Example 2. Specifying Logarithmic Axes" on page 297

Restriction: Not supported by Java

MAJOR=(*tick-mark-suboption(s)*)| NONE
modifies the major tick marks. *Tick-mark-suboption(s)* 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.
Tick-mark-suboption can be

COLOR=*tick-color*

HEIGHT=*tick-height <units >*

NUMBER=*number-of-ticks*

WIDTH=*thickness-factor*

See "Tick Mark Description Suboptions" on page 214 for complete descriptions. List all suboptions and their values within the parentheses.

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.

Note: By default, tick marks are now 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.

HEIGHT is not supported by Java or ActiveX. WIDTH is not supported by Java. △

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294 , "Example 2. Specifying Logarithmic Axes" on page 297, and "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309

Restriction: Partially supported by Java and ActiveX

MINOR=(*tick-mark-suboption(s)*)| NONE

modifies the minor tick marks that appear between major tick marks. *Tick-mark-suboption(s)* defines the color, number, or size of the minor tick marks. NONE suppresses all minor tick marks. *Tick-mark-suboption* can be

COLOR=*tick-color*

HEIGHT=*tick-height <units >*

NUMBER=*number-of-ticks*

WIDTH=*thickness-factor*

See "Tick Mark Description Suboptions" on page 214 for complete descriptions. List all suboptions and their values within the parentheses.

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.

This option is not supported by the GRADAR Procedure.

HEIGHT is not supported by Java or ActiveX.

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294, "Example 2. Specifying Logarithmic Axes" on page 297, and "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309

Restriction: Partially supported by Java and ActiveX

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.

This option is not supported by the GRADAR Procedure.

See also: GROUP= on page 1025 and GAXIS= on page 1025

Restriction: Not supported by Java and ActiveX

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 backplane, use the NOFRAME option in the procedure.

This option is not supported by the GRADAR Procedure.

Featured in: "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309.

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. This option is not supported by the GRADAR Procedure.

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294

Restriction: Not supported by Java

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:

- For numeric variables, *value-list* is either an explicit list of values or a starting 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)
  ```

  If the specified range is not evenly divisible by the increment value, 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)
```

- *For character variables*, *value-list* is a list of unique character values enclosed in quotes 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 PROC GCHART and the *unformatted* values for PROC GPLOT.

  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.

- *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 quotes 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=("25MAY98"d "04JUL98"d "07SEP98"d)
order=("01JUL97"d to "01AUG97"d)
order=("01JUL97"d to "01JAN98"d by week)
order=("9:25"t to "11:25"t by minute)
order=("04JUN97:12:00:00"dt to
       "10JUN97:12:00:00"dt by dtday)
```

  With SAS date and time values, use a FORMAT statement so that the tick mark values have an understandable form. For more information on SAS date and time values, see the *SAS Language Reference: Dictionary*.

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, if 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. △

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 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, "The GCHART Procedure," on page 989.

The ORDER= option overrides the suboption NUMBER= described in "Tick Mark Description Suboptions" on page 214.

The ORDER= option is not valid with the ASCENDING, DESCENDING, and NOZEROS options used with the bar chart statements in the GCHART procedure.

This option is not supported by the GRADAR procedure.

Note: The Java applet supports the ORDER= option for numeric axes, but does not support the ORDER= option for categorical, character, midpoint, or group axes.

The ActiveX control supports only simple order lists. Non-uniform interval values, such as dates, are not supported. Only maximum and minimum values are supported with a default interval of one day. △

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294, "Example 5. Filling the Area between Plot Lines" on page 304, and "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309

Restriction: Partially supported by Java and ActiveX

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.

This option is not supported by the GRADAR Procedure.

Restriction: Not supported by Java and ActiveX

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 quotes. Separate multiple strings with blank spaces. The strings are applied to the reference lines specified by the VREF or HREF option.

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*
AUTOREF
COLOR=*text-color*
FONT=*font* | NONE
HEIGHT=*text-height* <*units* >
JUSTIFY=LEFT | CENTER | RIGHT
POSITION=TOP| MIDDLE| BOTTOM
ROTATE=*degrees*
T=*n*
See "Text Description Suboptions" on page 210 for a complete description.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

REFLABEL is not supported by the GRADAR Procedure.

Style Reference: Font and Color attributes of the GraphLabelText element

Restriction: Not supported by Java and ActiveX

STAGGER
offsets the axis values on a horizontal axis. This option is useful when values overlap on an axis. When specifying the Java and ActiveX devices, the STAGGER option must sometimes be used in conjunction with the ORDER statement.

SPLIT="*split-char*"
specifies the split character that the AXIS statement uses to break axis values into multiple lines. *Split-char* can be any character value that can be specified in a SAS character variable. The split character must be embedded in the variable values in the data set 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**, and you specify SPLIT=",". The value would appear on the axis as follows:

```
Berlin
Germany
```

Note that the split character itself is not displayed.

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.

This option is not supported by the GRADAR Procedure.

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

Restriction: Not supported by Java and ActiveX

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.

Note: 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 axis *frame* is modified. △

Note: See also: Figure 14.22 on page 277 for examples of the available line types. △

Style Reference: Line style attribute of the GraphAxisLine element

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 quotes 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 Java or ActiveX devices, it is necessary to use the ORDER= option to ensure 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.

To change the value of midpoints in bar charts produced with the GCHART procedure, use the MIDPOINTS= option in the procedure. △

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=*text-color*

FONT=*font* | NONE

HEIGHT=*text-height <units >*

JUSTIFY=LEFT | CENTER | RIGHT

ROTATE=*degrees*

TICK=*n*.

For a complete description, see "Text Description Suboptions" on page 210.

Place text description suboptions before the text strings 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.

Note: If an end user viewing a graph in the Java applet or ActiveX control zooms 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. △

Style Element: Color attribute of the GraphLabelText graph element

Featured in: "Example 2. Specifying Logarithmic Axes" on page 297, "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309, and "Example 9. Combining Graphs and Reports in a Web Page" on page 315

Restriction: Partially supported by Java

WIDTH=*thickness-factor*
specifies the thickness of the axis line. Thickness increases directly with the value of *thickness-factor*. By default, WIDTH=1.

Note: 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 axis *frame* is modified.

Java does not support the WIDTH option. ActiveX ignores the WIDTH option for the vertical axis of an AXIS statement with GPLOT and GCONTOUR. △

Style Reference: LineThickness attribute of the GraphAxisLines element

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294

Restriction: Not supported by Java and partially supported by ActiveX

Text Description Suboptions

Text description suboptions are used by the LABEL=, REFLABEL=, and VALUE= options to change the color, height, justification, font, and angle of either default text or specified text strings. See the LABEL= option on page 201, the REFLABEL= option on page 207, and the VALUE= option on page 209.

ANGLE=*degrees*
A=*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 Java or ActiveX device drivers. △

Alias: A=

Restriction: Partially supported by Java

See also: the ROTATE= suboption on page 213

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

AUTOREF
automatically labels each reference line on an axis with the response value at the reference line's position. The AUTOREF option is used only with the REFLABEL=

option. The automatic labels are applied only to reference lines that do not have 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, however, that if you simultaneously request automatic labeling with a PLOT or BUBBLE statement (using the AUTOHREF or AUTOVREF option), then the automatic labeling can write on top of the custom label 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 Java, ActiveX, and GIF

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 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 for details on 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 style font, NONE.

Alias: F=

Restriction: Partially supported by Java

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 it applies to.

- With the LABEL= option:
  - 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:
  - 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, and LEFTplaces 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 is centered on top of the line, and LEFT places the text string just to the left of the line.

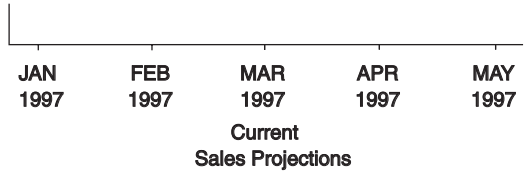
- With the VALUE= option:
  - for numeric variables on a vertical axis, the default is JUSTIFY=RIGHT
  - for character variables on a vertical axis, the default is JUSTIFY=LEFT
  - for all variables on a horizontal axis, the default is JUSTIFY=CENTER.

Note: With output using Java and 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. △

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 14.4 on page 212.

```
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 14.4 The JUSTIFY= suboption



Specify additional suboptions before any string.

Alias: J=L | C | R

Restriction: Not supported by Java

See also: the suboption TICK= on page 213

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, and 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, and BOTTOM places the label at the bottom end of the line.

Restriction: Not supported by Java and 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=*degrees*

Restriction: Partially supported by Java

See also: the suboption ANGLE= on page 210

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 *n*th 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")
```

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)
```

-

Note: The Java and ActiveX device drivers do not support the REFLABEL option. △

With the VALUE= option, the TICK= option specifies the *n*th 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=*n*

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)**.

Tick Mark Description Suboptions

Tick mark description suboptions are used by the MAJOR= and the MINOR= options to change the color, height, width, and number of the tick marks to which they apply. See the MAJOR= and MINOR= options.

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=*tick-mark color*

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=*tick-height <units>*

Restriction: Not supported by Java and 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=*number-of-ticks*

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.

Style Reference: LineThickness attribute of the GraphAxisLines element.

Alias: W=*thickness-factor*

Restriction: Partially supported by Java

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:

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.

BY Statement

Processes data and orders output according to the BY group.

Used by: GAREABAR, GCHART, GBARLINE, GCONTOUR, GMAP, GPLOT, GRADAR, GREDUCE, G3D, G3GRID procedures

Syntax

BY<DESCENDING> *variable*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

Description

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. △

Required Arguments

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.

Options

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. If observations with the same value for the BY variables are not contiguous, the procedure treats each new value it encounters as the first observation in a new BY group and creates a graph for that value, even if it is only one observation.

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, either sort it with the SORT procedure using the same BY statement that you plan to use in the target SAS/GRAPH procedure or 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 309 .

Controlling BY Lines

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 309.

Suppressing the name of the BY variable To 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.

See Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for a complete description of each option.

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 also "Specifying the Catalog Name and Entry Name for Your GRSEGs" on page 100. For an example of incremented catalog names, see "Example 9. Combining Graphs and Reports in a Web Page" on page 315.

Using the BY Statement

This section describes the following:

- the effect of BY-group processing on the GCHART, GMAP, 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 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 on any of these topics, refer to the appropriate chapter.

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 220.
- 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 to produce the same response axis scaling in all horizontal or vertical bar charts in a BY group.

With the PIE and STAR statements, the effect of a BY statement is similar to that of the GROUP= option, except 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.

With the GMAP Procedure By default, BY-group processing affects both the map data set and the response data set. This means that you get separate, individual output for

each map area common to both data sets. For example, if the map data set REGION contains six states and the response data set contains the same six states, and you specify BY STATE in the GMAP procedure, you get six graphs with one state on each graph.

If you use the ALL option in the PROC GMAP statement and you also use the BY statement, you get one output for each map area in the response data set, but that output displays all the map areas in the map data set. Only one map area per output contains response data information; the others are empty. For example, if you create a block map using the data sets REGION and SALES, specify BY STATE, and include the ALL option in the PROC GMAP statement, you get six graphs with six states on each graph. One state per graph has a block; the remaining five are empty. The UNIFORM option applies colors and heights uniformly across all BY-groups.

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.

With the RUN Groups If you use the BY statement with a procedure that processes data and supports RUN-group processing (the GCHART, GMAP, and GPLOT procedures), then 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 SITE:

```
/* first run group*/
proc gplot data=sales;
   title1 "Sales Summary";
   by site;
   plot sales*model_a;
run;

   /* second run group */
   plot sales*model_b;
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, but when you do this, the null BY statement turns off BY-group processing *and* the RUN group generates a graph.

For more information, see “RUN-Group Processing” on page 56.

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 by including the BY variable in the Annotate data set and by using each BY-variable value as a condition for the annotation to be applied to the output for that value.

With TITLE, FOOTNOTE, and NOTE Statements TITLE, FOOTNOTE, and NOTE statements can automatically include the BY variable name, BY variable values, or BY

lines in the text they produce. To insert BY variable information into the text strings used by these statements, use the #BYVAR, #BYVAL, and #BYLINE substitution options. For an example, see "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309.

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, if the BY variable contains four values and there are two response levels for each BY value, the PATTERN1 and PATTERN2 or SYMBOL1 and SYMBOL2 statements are used for each graph. Each BY-group starts over with PATTERN1 or SYMBOL1. The UNIFORM option in the GMAP procedure changes this behavior.

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`**.

If you create sets of graphs from several data sets containing the variable REGION, and if you want the same pattern assigned to the same region each time, you must be sure that REGION always has the same four values. Otherwise, the patterns may 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.

See: "TITLE, FOOTNOTE, and NOTE Statements" on page 279

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

Syntax

GOPTIONS <*options-list*>;

options-list can be one or more options from any or all of the following categories:

- reset option

  RESET=ALL | GLOBAL | *statement-name* | (*statement-name(s)*)

- options that affect the appearance of the display area and the graphics output

  ASPECT=*scaling-factor*
  ALTDESC | NOALTDESC
  AUTOSIZE=ON | OFF | DEFAULT
  BORDER | NOBORDER
  CELL | NOCELL
  GSIZE=*lines*
  HORIGIN=*horizontal-offset* <IN | CM>
  HPOS=*columns*
  HSIZE=*horizontal-size* <IN | CM>
  IBACK= *fileref* | "*external-file*"
  IMAGESTYLE = TILE | FIT
  IMAGEPRINT | NOIMAGEPRINT
  ROTATE=LANDSCAPE | PORTRAIT
  ROTATE | NOROTATE
  TARGETDEVICE=*target-device-entry*
  VORIGIN=*vertical-offset* <IN | CM>
  VPOS=*rows*
  VSIZE=*vertical-size* <IN | CM>
  XMAX=*width* <IN | CM>
  XPIXELS=*width-in-pixels*
  YMAX=*height* <IN | CM>
  YPIXELS=*height-in-pixels*

- options that affect color

  CBACK=*background-color*
  CBY=*BY-line-color*
  COLORS=<(*colors-list* | NONE)>
  CPATTERN=*pattern-color*
  CSYMBOL=*symbol-color*
  CTEXT=*text-color*
  CTITLE=*title-color*
  PENMOUNTS=*active-pen-mounts*
  PENSORT | NOPENSORT

- options that control font selection or text appearance

  CHARTYPE=*hardware-font-chartype*
  FASTTEXT | NOFASTTEXT
  FBY=*BY-line-font*

FCACHE=*number-fonts-open*
FONTRES=NORMAL | PRESENTATION
FTEXT=*text-font*
FTITLE=*title-font*
FTRACK=LOOSE | NONE | NORMAL | TIGHT | TOUCH | V5
HBY=*BY-line-height <units>*
HTEXT=*text-height <units>*
HTITLE=*title-height <units>*
RENDER=APPEND | DISK | MEMORY | NONE | READ
RENDERLIB=*libref*
SIMFONT=*software-font*

- options that set defaults for procedures and global statements
  GUNIT=*units*
  INTERPOL=*interpolation-method*
  OFFSHADOW=(*x <units>*, *y <units>* | (*x*,*y*) *<units>*
  V6COMP | NOV6COMP
- image animation options
  DELAY=*delay-time*
  DISPOSAL=NONE | BACKGROUND | PREVIOUS | UNSPECIFIED
  INTERLACED | NONINTERLACED
  ITERATION=*iteration-count*
  TRANSPARENCY | NOTRANSPARENCY
- options that affect how your SAS/GRAPH program runs
  DISPLAY | NODISPLAY
  ERASE | NOERASE
  GWAIT=*seconds*
  GRAPHRC | NOGRAPHRC
  IMAGEPRINT | NOIMAGEPRINT
  PCLIP | NOPCLIP
  POLYGONCLIP | NOPOLYGONCLIP
- options that control how output is sent to devices or files
  ADMGDF | NOADMGDF
  DEVADDR=*device-address*
  DEVICE=*device-entry*
  DEVMAP=*device-map-name* | NONE
  EXTENSION="*file-type*"
  FILECLOSE=DRIVERTERM | GRAPHEND
  FILEONLY | NOFILEONLY
  GACCESS=*output-format* | "*output-format* > *destination*"
  GEND="*string*" <..."*string-n*">
  GEPILOG="*string*" <..."*string-n*">
  GOUTMODE=APPEND | REPLACE
  GPROLOG="*string*" <..."*string-n*">
  GPROTOCOL=*module-name*
  GSFLEN=*record-length*

GSFMODE=APPEND | PORT | REPLACE
GSFNAME=*fileref*
GSFPROMPT | NOGSFPROMPT
GSTART="*string*" <..."*string-n*">
HANDSHAKE=HARDWARE | NONE | SOFTWARE | XONXOFF
KEYMAP=*map-name* | NONE
POSTGEPILOG="*string*"
POSTGPROLOG="*string*"
PREGEPILOG="*string*"
PREGPROLOG="*string*"
PROMPTCHARS="*prompt-chars-hex-string*"X

□ options that specify hardware capabilities of the device

CHARACTERS | NOCHARACTERS
CIRCLEARC | NOCIRCLEARC
DASH | NODASH
DASHSCALE=*scaling-factor*
FILL | NOFILL
FILLINC=0...9999
LFACTOR=*line-thickness-factor*
PIEFILL | NOPIEFILL
POLYGONFILL | NOPOLYGONFILL
SYMBOL | NOSYMBOL

□ options that control printer hardware features

AUTOCOPY | NOAUTOCOPY
AUTOFEED | NOAUTOFEED
BINDING=DEFAULTEDGE | LONGEDGE | SHORTEDGE
COLLATE | NOCOLLATE
DUPLEX | NODUPLEX
GCOPIES=(<*current-copies*><,*max-copies*>)
PAPERDEST=*bin*
PAPERFEED=*feed-increment* <IN | CM>
PAPERLIMIT=*width* <IN | CM>
PAPERSIZE="*size-name*" | (*width*,*height*)
PAPERSOURCE=*tray*
PAPERTYPE="*type-name*"
PPDFILE=*fileref* | "*external-file*"
REPAINT=*redraw-factor*
REVERSE | NOREVERSE
SPEED=*pen-speed*
UCC="*control-characters-hex-string*"X

□ options that interact with the operating environment

DRVINIT="*system-command(s)*"
DRVTERM="*system-command(s)*"
PREGRAPH="*system-command(s)*"
POSTGRAPH="*system-command(s)*"

PROMPT | NOPROMPT

- options for mainframe systems

  GCLASS=*SYSOUT-class*

  GDDMCOPY=FSCOPY | GSCOPY

  GDDMNICKNAME=*nickname*

  GDDMTOKEN=*token*

  GDEST=*destination*

  GFORMS="*forms-code*"

  GWRITER="*writer-name*"

  TRANTAB=*table* | *user-defined-table*

Description

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 Chapter 38, "The GDEVICE Procedure," on page 1125 for details.

To review the current settings of all graphics options, use the GOPTIONS procedure. See Chapter 44, "The GOPTIONS Procedure," on page 1319 for details.

Options

See Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for a complete description of all graphics options used by the GOPTIONS statement.

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 either specify the option in another GOPTIONS statement, or 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 you end your SAS

session. 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 background color, text height, and text font:

```
goptions cback=blue htext=6 pct ftext=albany;
```

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, maps, and charts.

Used by: GAREABAR, GCHART, GBARLINE, GCONTOUR, GMAP, GPLOT procedures

Type: Global

Syntax

LEGEND<1...99> <*options*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options

  ACROSS=*number-of-columns*

  CBLOCK=*block-color*

  CBORDER=*frame-color*

CFRAME=*background-color*

CSHADOW=*shadow-color*

DOWN=*number-of-rows*

FRAME

FWIDTH=*thickness-factor*

REPEAT=1 | 2 | 3

ROWMAJOR | COLMAJOR

SHAPE=BAR(*width*,*height*) <*units*> | LINE(*length*) <*units*> | SYMBOL(*width*,*height*) <*units*>

- position-options

  MODE=PROTECT | RESERVE | SHARE

  OFFSET=(<*x* >,<*y* >)<*units* > | (<*x* <*units* >>,<*y* <*units* >>)

  ORIGIN=(<*x* >,<*y* >)<*units* > | (<*x* <*units* >>,<*y* <*units* >>)

  POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <INSIDE | OUTSIDE>)

- text-options

  LABEL=(*text-argument(s)*) | NONE

  ORDER=(*value-list*)

  VALUE=(*text-argument(s)*) | NONE

Description

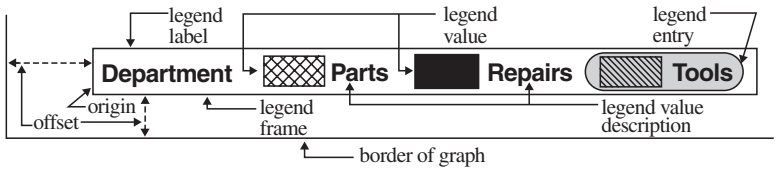
LEGEND statements specify the characteristics of a legend but do not create legends. The characteristics are as follows:

- 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 14.5 Parts of a Legend



Options

When the syntax of an option includes *units*, use one of these:

CELLS	character cells
CM	centimeters

IN	inches
PT	points
PCT	percentage of the graphics output area

Note: The Java applet does not support CM, IN, or PT. △

If you omit *units*, a unit specification is searched for in this order:

1. GUNIT= in a GOPTIONS statement
2. the default unit, CELLS

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.

Featured in: "Example 8. Creating a Simple Web Page with the ODS HTML Statement" on page 313

See also: ROWMAJOR, COLMAJOR

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*).

The CBLOCK= and CSHADOW= options are mutually exclusive. If both are present, SAS/GRAPH software uses the last one specified. The CBLOCK= option is usually used in conjunction with the FRAME, CFRAME=, or CBORDER= options.

The Java applet treats the CBLOCK option like the CSHADOW option.

See also: The OFFSHADOW="OFFSHADOW" on page 394 graphics option and "Creating Drop Shadows and Block Effects" on page 238

Restriction: Not supported by Java.

CBORDER=*frame-color*
draws a colored frame around the legend. This option overrides the FRAME option. CBORDER= can be used in conjunction with the CFRAME= option.

Style Reference: Color attribute of the GraphBorderLines graph element

CFRAME=*background-color*
specifies the background color of the legend. This option overrides the FRAME option. If both the CFRAME= and FRAME= options are specified, only the solid background produced by the CFRAME= option is displayed. The CFRAME= option can be used in conjunction with the CBORDER= option.

Style Reference: Color attribute of the GraphLegendBackground graph element

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*).

The CSHADOW= and CBLOCK= options are mutually exclusive. If both are present, SAS/GRAPH uses the last one specified. The CSHADOW= option is usually specified in conjunction with the FRAME, CFRAME=, or CBORDER= options.

See also: the OFFSHADOW="OFFSHADOW" on page 394 graphics option and "Creating Drop Shadows and Block Effects" on page 238.

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. The ROWMAJOR option is the default.

FRAME
draws a frame around the legend. The color of the frame is the first color in the color list.

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 Java and 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. 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"). *Text-argument(s)* can be one or more of these:

"text-string"
provides up to 256 characters of label text. Enclose each string in quotes. Separate multiple strings with blanks.

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:

COLOR=*text-color*

FONT=*font* | NONE

HEIGHT=*text-height <units>*

JUSTIFY=LEFT | CENTER | RIGHT

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)

Note: The Java applet does not support the POSITION= suboption—it draws legend labels at the top-left of the legend. Also, it does not support multiple values for the JUSTIFY= suboption (only the first is honored). The ActiveX control supports the POSITION= option but does not support multiple values for the JUSTIFY suboption (only the first is honored). △

See "Text Description Suboptions" on page 233 for complete descriptions.

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

Featured in: "Example 3. Rotating Plot Symbols Through the Color List" on page 299 and "Example 8. Creating a Simple Web Page with the ODS HTML Statement" on page 313

Restriction: Partially supported by Java and ActiveX

MODE=PROTECT | RESERVE | SHARE
specifies whether the legend is drawn in the procedure output area or whether legend elements can overlay other graphics elements. MODE= 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.

See also: "Positioning the Legend" on page 237

Restriction: Not supported by Java and ActiveX

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), and *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 since 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,l and the LEGEND is positioned accordingly.

See also: "Positioning the Legend" on page 237 and the option POSITION= on page 230

Restriction: Not supported by Java and ActiveX

ORDER=(*value-list*)
selects or orders the legend values that 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 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 quotes 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 option ORDER= on page 205 in the AXIS statement.

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.

Restriction: Not supported by Java and ActiveX

ORIGIN=(<*x*><,*y*>)<*units*> | (<*x* <*units* >><,*y* <*units*>>)
specifies the *x* coordinate and the *y* coordinate 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.

See also: "Positioning the Legend" on page 237

Restriction: Not supported by Java and ActiveX

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)
positions the legend on the graph. Values for POSITION= are

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 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.

Note: The Java applet defaults to BOTTOM-CENTER and supports all possible combinations of BOTTOM | MIDDLE | TOP with LEFT | CENTER | RIGHT except for MIDDLE-CENTER (which would overwrite the map.) The Java applet does not support INSIDE for positioning. △

See also: OFFSET= option on page 229 and MODE= option on page 228

Restriction: Partially supported by Java

REPEAT=1 | 2 | 3
Use the REPEAT= option to specify how many times the plot symbol is repeated in the legend. Valid values are 1 to 3, with 3 being the default.

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 also: ACROSS=, DOWN=

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 you specify depends on which procedure generates the legend.

BAR(*width*,*height*)<*units*>
is used with the GCHART and GMAP procedures, 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. By default, *width* is 5, *height* is 0.8, and *units* are CELLS. You can specify *units* for the *width*,*height* pair or for the individual coordinates.

LINE(*length*) <*units*>
is used with the GPLOT and GCONTOUR procedures. Each legend value is a line of the length you specify. Plotting symbols are omitted from the legend values. By default, *length* is 5 and *units* are CELLS. You can specify *units* for *length*.

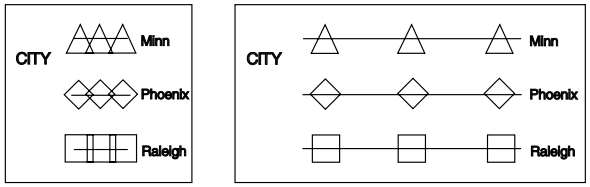
SYMBOL(*width*<*units*>,*height*<*units*>) <*units*>
is used with the GPLOT procedure. Each legend value (*not* each symbol) is the width and height you specify. For example, this specification produces legend values like the ones in Figure 14.6 on page 232(a):

```
shape=symbol(.5,.5)
```

This specification produces legend values like the ones in Figure 14.6 on page 232(b):

```
shape=symbol(2,.5)
```

Figure 14.6 Legend Values Produced with SHAPE= SYMBOL



a. b.

By default, *width* is 5, *height* is 1, and *units* are CELLS. You can specify *units* for the *width*,*height* pair or for the individual coordinates.

Restriction: Not supported by Java and ActiveX

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.

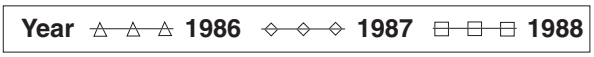
NONE suppresses the value descriptions although the legend values (bars, lines, and so on) are still displayed. (NONE is not supported by Java or ActiveX). *Text-argument(s)* can be one or more of these:

"text-string"
provides up to 256 characters of text for the value description. Enclose each string in quotes. Separate multiple strings with blanks.

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 14.7 on page 232:

```
value=("1986" "1987" "1988")
```

Figure 14.7 Specifying Value Descriptions with the VALUE= Option



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:

COLOR=*text-color*

FONT=*font* | NONE

HEIGHT=*text-height <units >*

JUSTIFY=LEFT | CENTER | RIGHT

TICK=*n*

See "Text Description Suboptions" on page 233 for complete descriptions. Place text description suboptions before the text strings 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.

See also: "Text Description Suboptions" on page 233 and the option ORDER= on page 230

Restriction: Partially supported by Java and ActiveX

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 the LABEL= suboption on page 228 and the VALUE= suboption on page 232.

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=*text-color*

FONT=*font* | NONE
specifies the font for the text. See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 for information on 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=*font* | NONE

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=*text-height <units>*

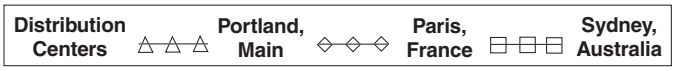
JUSTIFY=LEFT | CENTER | RIGHT
specifies the alignment of the text. The default for character variables is JUSTIFY=LEFT. The default for numeric variables is JUSTIFY=RIGHT.

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 14.8 on page 234:

```
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 14.8 Specifying Multiple Lines of Text with the JUSTIFY= Suboption



Specify additional suboptions before any string.

See also: the suboption TICK= on page 235.

Alias: J=L | C | R

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.

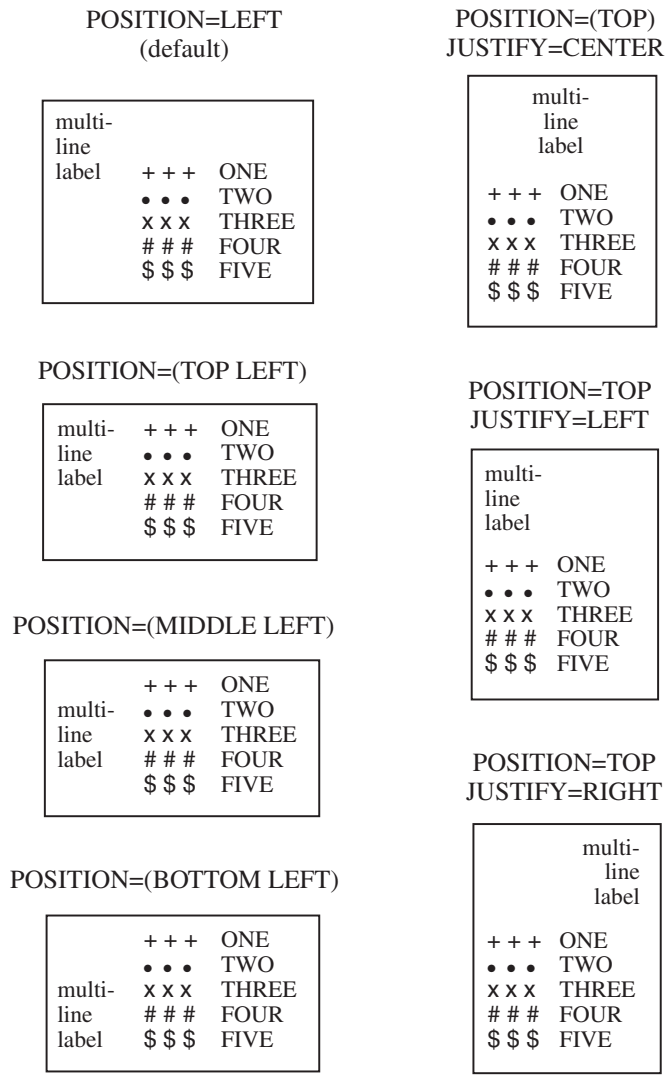
The parentheses are not required if only one value is supplied. If you specify two or three values and omit the parentheses, SAS/GRAPH accepts the first value and ignores the others.

Figure 14.9 on page 235 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 14.9 Using the POSITION= Suboption with Multiple-line Legend Labels



POSITION

In addition, specifying POSITION=RIGHT mirrors the effect of POSITION=LEFT, and specifying POSITION=BOTTOM mirrors the effect of POSITION=TOP.

Restriction: Not supported by Java. Partially supported by ActiveX.

TICK=*n*

specifies the *n*th 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=*n*

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 LOG window, use the GOPTIONS procedure with the LEGEND option:

```
proc goptions legend nolist;
run;
```

Positioning the Legend

By default, the legend shares the procedure output area with the procedure output, such as a map or bar chart. (See "How Graphic Elements are Placed in the Graphics Output Area" on page 65.) 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 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 14.10 on page 237 shows the legend positions inside the axis area.

Figure 14.10 Legend Positions Inside the Axis Area

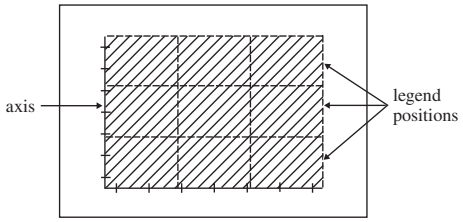
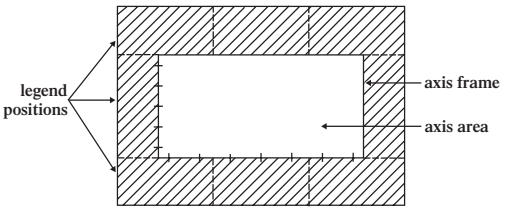


Figure 14.11 on page 237 shows legend positions outside the axis area.

Figure 14.11 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 Graphic Elements By default, the legend is inside the procedure output area and the space it occupies reduces the size of the graph itself. To control the way the legend relates to the other elements of the graph, use the MODE= option. These are values for the MODE= option:

- RESERVE reserve space for the legend outside the axis area and move 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

To 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.

NOTE Statement

Writes lines of text in the output.

See: "TITLE, FOOTNOTE, and NOTE Statements" on page 279

Syntax

NOTE <*text-arguments(s)*>;

ODS HTML Statement

Opens or closes the HTML destination.

Used by: GANNO, GAREABAR, GBARLINE, GCHART, GCONTOUR, GIMPORT, GMAP, GPLOT, GRADAR, GREPLAY, GSLIDE, and G3D procedures

Requirements: On mainframes, either GPATH= or PATH= is required.

Syntax

ODS HTML <(<ID=>*identifier*)> <*action*>;

ODS HTML <(<ID=>*identifier*)> <*option(s)*>;

Description

This section describes the ODS HTML statement as it relates to SAS/GRAPH procedures. For complete information on the ODS HTML statement, see *SAS Output Delivery System: User's Guide*

The ODS HTML statement opens or closes the HTML destination. If 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.

If DEVICE=JAVAMETA, graphics output is produced as metagraphics data. The browser passes the metacodes as a parameter to the Metaview applet. The Metaview applet renders the output defined by the metacodes, and displays the interactive graph in a Web page. For more information on DEVICE=JAVAMETA see "Developing Web Presentations for the Metaview Applet" on page 531.

You can also use the DEVICE=JAVA and DEVICE=ACTIVEX options to create interactive graphics presentations for the Web.

SAS/GRAPH adds datatip text to some graphs depending on the device specified. These datatips 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 datatip text. For more information on using data tips see "Data Tips for Web Presentations" on page 598.

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, the PNG files are stored in the current directory. To specify a destination for all the HTML and PNG files, use the PATH= option. To store thePNG files in a different location than the HTML files, use the GPATH= option to specify a location for thePNG 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 *SAS Output Delivery System: User's Guide*. .

PATTERN Statement

Defines the characteristics of patterns used in graphs.

Used by: GCHART, GBARLINE, GCONTOUR, GMAP, GPLOT procedures; SYMBOL statement; Annotate facility.

Type: Global

Syntax

PATTERN<1...255> <COLOR=*pattern-color* | *_style_*>
<REPEAT=*number-of-times*>
<VALUE=*bar*/*block-pattern* | *map*/*plot-pattern* | *pie*/*star-pattern* >;

- *bar*/*block-pattern* can be one of these:
  EMPTY
  SOLID
  style <*density*>
- *map*/*plot-pattern* can be one of these:
  MEMPTY
  MSOLID
  M*density* <*style* <*angle*>>
- *pie*/*star-pattern* can be one of these:
  PEMPTY
  PSOLID
  P*density* <*style* <*angle*>>

Description

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
GMAP	map areas in choropleth, block, and prism maps; blocks in block maps
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 252 and Chapter 29, "Using Annotate Data Sets," on page 641.

You can use the PATTERN statement to control the fill and color of a pattern, and whether the pattern is repeated. There are three types of patterns:

- bar and block patterns
- map and 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 rectangle, polygon, 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 248.

Options

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 12, "SAS/GRAPH Colors and Images," on page 167 the *SAS/GRAPH: Reference* for more information on specifying colors and images.

Note: ActiveX assigns colors in a different order from Java, so the same data can appear differently with those two drivers. △

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 1053.

CAUTION:
Omitting the COLOR= option in a PATTERN statement can cause the PATTERN statement to generate multiple PATTERN definitions. △

If no color is specified for a PATTERN statement, that is, if neither the COLOR= nor the CPATTERN= option is used, the PATTERN statement rotates the specified fill through each color in the color list before the next PATTERN statement is used. .

Alias: C=*pattern-color*

See also: "Working with PATTERN Statements" on page 249

Featured in: "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309

Restriction: Partially supported by Java and ActiveX

IMAGE= *fileref* | "*external-file*"
specifies an image file that is used to fill one or more bars of a bar chart, as generated by the HBAR, HBAR3D, VBAR, and VBAR3D statements of the GCHART procedure. The format of the external file specification varies across operating environments. See also the IMAGESTYLE= option.

Note: When you specify an image file to fill a bar, the bar is not outlined. Also, the COLOR= and VALUE= options are ignored. △

Note: If an image is specified on 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, if you submit a PIE statement when an image has been specified in a PATTERN statement, 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=JAVA and DEVICE=JAVAIMG, the IMAGE= option works only for the VBAR and HBAR statements. △

See also: For related information, see "Displaying Images on Data Elements" on page 185

Restriction: Partially supported by Java and ActiveX

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, which is the default, repeats the image as needed to fill the bar. The FIT value stretches a single instance of the image to fill the bar.

Restriction: Partially supported by Java and ActiveX

REPEAT=*number-of-times*
specifies the number of times that a PATTERN definition is applied before the next PATTERN definition is used. By default, REPEAT=1.

The behavior of the REPEAT= option depends on the color specification:

- If you use both the COLOR= and the 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.
- If you use the CPATTERN= option in a GOPTIONS statement to specify a single pattern color, and 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.
- If you omit both the COLOR= and CPATTERN= options, and use the REPEAT= option either alone (generates default solids) or with the VALUE= option in a PATTERN statement, the resulting pattern is rotated through each color in the color list, and then the entire group generated by this cycle is 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. Note that in most cases, it is preferable to use LEVELS=1 in the GMAP procedure rather than using this option in the PATTERN statement.

Alias: R=*number-of-times*

See also: "Understanding Pattern Sequences" on page 251

Restriction: Partially supported by Java and ActiveX

VALUE=*bar*/*block-pattern*
specifies patterns for:

- 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.
- the blocks in block maps produced by the BLOCK statement in the GMAP procedure. (The map area from which the block rises takes a map pattern as described on the option VALUE= on page 244). See also "About Block Maps and Patterns" on page 1268.

Values for *bar/block-pattern* are as follows:

EMPTY
E — an empty pattern. Neither the Java applet nor the ActiveX control supports EMPTY.

SOLID
S — a solid pattern (the only valid value for three-dimensional charts).

style<density> — a shaded pattern.

Note: *style<density>* is not supported by the Java or ActiveX device drivers. △

Style specifies the direction of the lines:

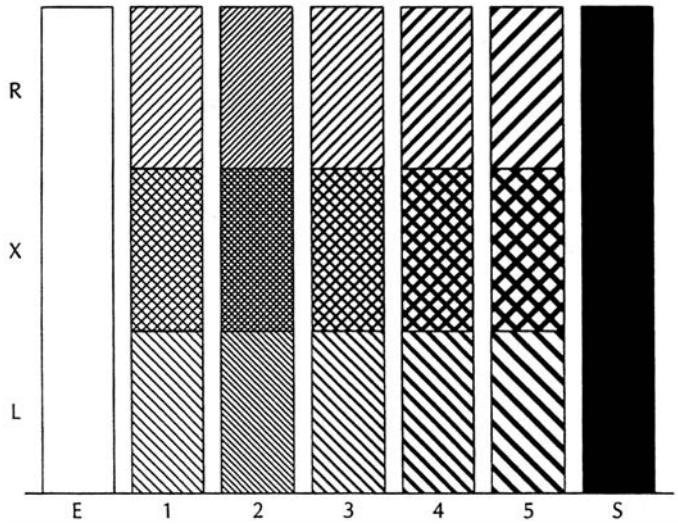
L	left-slanting lines.
R	right-slanting lines.
X	crosshatched lines.

Density specifies the density of the pattern's shading:

1...5	1 produces the lightest shading and 5 produces the heaviest shading.

Figure 14.12 on page 243 shows all of the patterns available for bars and blocks.

Figure 14.12 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=*bar*/*block-pattern*

Restriction: Partially supported by Java and ActiveX

VALUE=*map*/*plot-pattern*
specifies patterns for the following:

- contour levels in contour plots produced by the GCONTOUR procedure
- map area surfaces in block, choropleth, and prism maps produced by the BLOCK, CHORO, AND PRISM statements in the GMAP procedure.
- areas under curves in plots produced by the AREAS= option in the PLOT statement in the GPLOT procedure.

Values for *map*/*plot-pattern* are as follows:

MEMPTY ME	an empty pattern. EMPTY or E are also valid aliases, except when used with the map areas in block maps created by the GMAP procedure.
MSOLID MS	a solid pattern. SOLID or S are also valid aliases, except when used with the map areas in block maps created by the GMAP procedure.

M*density*<*style*<*angle*>> a shaded pattern.

Note: M*density*<*style*<*angle*>> is not supported by the Java or ActiveX device drivers. △

Density specifies the density of the pattern's shading:

1...5	1 produces the lightest shading and 5 produces the heaviest shading.

Style specifies the type of the pattern lines:

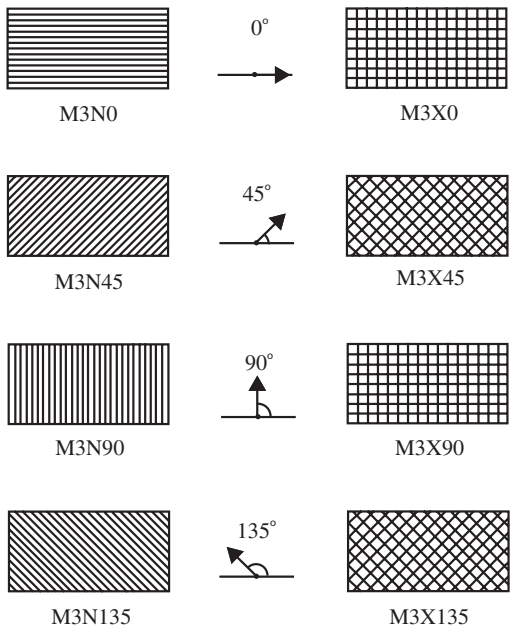
N	parallel lines (the default).
X	crosshatched lines.

Angle specifies the angle of the pattern lines:

0...360	the degrees at which the parallel lines are drawn, measured from the horizontal. By default, *angle* is 0 (lines are horizontal).

Figure 14.13 on page 245 shows some typical map and plot patterns.

Figure 14.13 Map and Plot Patterns



If no valid patterns are available, default map and 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=*map*/*plot-pattern*

Restriction: Partially supported by Java and 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

PEMPTY
PE
: an empty pattern. EMPTY or E are also valid aliases.

PSOLID
PS
: a solid pattern. SOLID or S are also valid aliases.

P*density*<*style*<*angle*>>
: a shaded pattern.

Note: P*density*<*style*<*angle*>> is not supported by the Java or ActiveX device drivers. △

Density specifies the density of the pattern's shading:

1...5	1 produces the lightest shading and 5 produces the heaviest shading.

Style specifies the type of the pattern lines:

N	parallel lines (the default).
X	crosshatched lines.

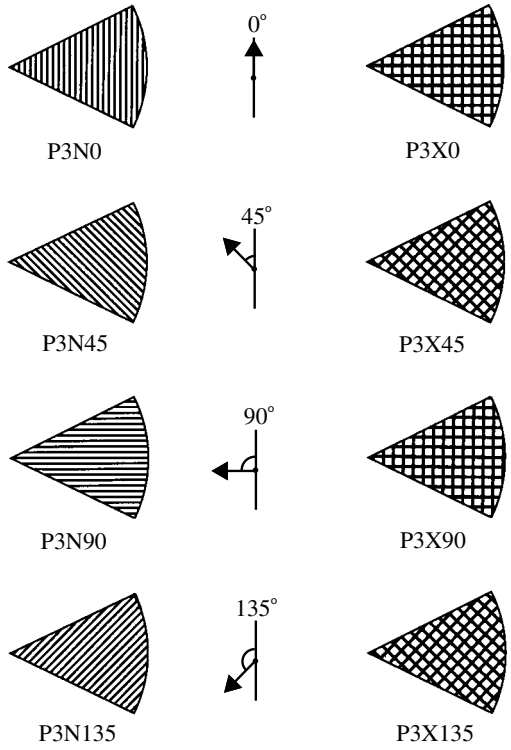
Angle specifies the angle of the pattern lines:

0...360	the angle of the lines, measured in degrees from perpendicular to the radius of the slice. By default, *angle* is 0.

The FILL= option in the PIE and STAR statements in the GCHART procedure overrides VALUE=.

Figure 14.14 on page 246 shows some typical pie and star patterns.

Figure 14.14 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.

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. △

Alias: V=*pie*/*star-pattern*

Restriction: Partially supported by Java and ActiveX

Using the PATTERN Statement

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 on PATTERN definitions, see "Working with PATTERN Statements" on page 249, as well as the description of COLOR= on page 241, VALUE= on page 244, and IMAGE= on page 241 options.

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. If you 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

When a procedure produces a graph that needs one or more patterns, SAS/GRAPH either does one of the following:

- automatically generates the appropriate default patterns and outlines to fill the areas, or
- 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 249 for details about defining patterns.

How Default Patterns and Outlines Are Generated In general, the default pattern that the 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 fill colors, plot line colors, widths, symbols, patterns, and outline colors when producing output. Specifically, SAS/GRAPHuses 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, and rotates it 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.

  Note: The one exception to the default solid pattern is the map area pattern in a block map produced by the GMAP procedure, which uses a hatch fill by default. By default the map areas and their outlines use the first color in the color list,

regardless of whether the list is the default device list or one specified with COLORS= in the GOPTIONS statement. △

- 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 graphics options, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

Working 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 2D chart (IMAGE=)
- how many times to apply the statement before using the next one (REPEAT=.

See "Displaying Images on Data Elements" on page 185 for information on 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 kind of patterns your PATTERN statements generate. Depending on the 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 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 assure that you get exactly the patterns 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 on pattern rotation, see "Understanding Pattern Sequences" on page 251.

Selecting an Appropriate Pattern The type of fill you specify depends on the type of graph you are producing:

With this type of graph	Use this type of fill
bar and block charts (PROC GCHART), block maps (PROC GMAP)	VALUE= bar/block-pattern on page 242
contour plots (PROC GCONTOUR), map area surfaces (PROC GMAP)	VALUE=map/plot-pattern on page 244
pie and star charts (PROC GCHART)	VALUE=pie/star-pattern on page 245

Note: If you specify a fill that is inappropriate for the type of graph you are generating (for example, if you specify VALUE=L1 in a PATTERN statement for a choropleth map), 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.

To change the outline color of any pattern, whether the pattern is default or user-defined, use the COUTLINE= option 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 it does the following:

- With a PATTERN statement that only specifies 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 only specifies 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="CPATTERN" on page 343.

Understanding 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, generating one PATTERN definition 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/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 shows the color and fill of the PATTERN definitions that would be generated if nine patterns were required:

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;
```

Here, **PATTERN1** is used twice and **PATTERN2** cycles through the list of three colors and then repeats this cycle a second time:

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 PROC GBARLINE, PROC GCONTOUR, and PROC GPLOT.

Used by: GBARLINE, GCONTOUR, GPLOT procedures

Type Global

Syntax

SYMBOL<1...255> <COLOR=*symbol-color* | *_style_*>
<MODE=EXCLUDE | INCLUDE> <REPEAT=*number-of-times*>
<STEP=*distance*<*units*>> <*appearance-option(s)*>
<*interpolation-option*> <SINGULAR=*n*>;

appearance-options can be one or more of these:

BWIDTH=*box-width*

CI=*line-color* | *_style_*
CO=*color*
CV=*value-color* | *_style_*
FONT=*font*
HEIGHT=*symbol-height*<*units*>
LINE=*line-type*
POINTLABEL<=(*label-description(s)*) | NONE>
VALUE=*special-symbol* | *text-string* | NONE
WIDTH=*thickness-factor*

interpolation-option can be one of these:

- general methods
  - INTERPOL=JOIN
  - INTERPOL=*map*/*plot-pattern*
  - INTERPOL=NEEDLE
  - INTERPOL=NONE
  - INTERPOL=STEP<*placement*><J><S>
- high-low interpolation methods
  - INTERPOL=BOX<*option(s)*><00...25>
  - INTERPOL=HILO<C><*option(s)*>
  - INTERPOL=STD<1 | 2 | 3><*variance*><*option(s)*>
- regression interpolation methods
  - INTERPOL=R<*type*><0><CLM | CLI<50...99>>
- spline interpolation methods
  - INTERPOL=L<*degree*><P><S>
  - INTERPOL=SM<*nn*><P><S>
  - INTERPOL=SPLINE<P><S>

Description

SYMBOL statements create SYMBOL definitions, which are used by the GPLOT, GBARLINE and GCONTOUR procedures.

For the GPLOT and GBARLINE procedure, 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
- the 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.

Options

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 searched for in this order:

1. the GUNIT= option in a GOPTIONS statement
2. the default unit, CELLS.

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 and omit the BWIDTH= option, the width of the box changes accordingly.

Featured in: "Example 4. Creating and Modifying Box Plots" on page 302.

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 also: "Using Color" on page 275

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294

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 also: "Using Color" on page 275

Featured in: "Example 5. Filling the Area between Plot Lines" on page 304 and "Example 4. Creating and Modifying Box Plots" on page 302

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.

If you do not use a SYMBOL statement to specify a color for each symbol, but you do specify a color list in a GOPTIONS statement, then Java and ActiveX assign colors to symbols differently than other devices. To ensure consistency on all devices, 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.

Note: Neither the Java applet nor the ActiveX control supports using COLOR= with PROC GCONTOUR. △

Style Reference: Color attribute of the GraphLabelText style element.

Alias: C=*symbol-color*

See also: "Using Color" on page 275

Restriction: Partially supported by Java and ActiveX

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.

Note: Neither the Java applet nor the ActiveX control supports using the CV= option with PROC GCONTOUR. △

See also: "Using Color" on page 275

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294, "Example 5. Filling the Area between Plot Lines" on page 304, and "Example 4. Creating and Modifying Box Plots" on page 302

Restriction: Partially supported by Java and ActiveX

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 quotes 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 Figure 14.21 on page 271. To use symbols from the special symbol table, you must omit the FONT= option.

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=*font*

See also: the VALUE= option on page 269, "Specifying Plot Symbols" on page 274, and "Specifying Special Characters Using Character and Hexadecimal Codes" on page 160.

Featured in: Example 2 on page 1116

Restriction: Not supported by Java and ActiveX

HEIGHT=*symbol-height<units>*
specifies the height in number of units of plot symbols (GPLOT, GBARLINE) or contour labels (GCONTOUR).

Note: 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 graph styles. For more information on graph styles, see *SAS Output Delivery System: User's Guide*. △

Note: With the Java device driver, the minimum height is two pixels; with ActiveX a symbol can be so small as to be invisible.

Neither the Java applet nor the ActiveX control supports HEIGHT= with PROC GCONTOUR. △

Alias: H=*symbol-height<units>*

See also: the option SHAPE= on page 231 in the LEGEND statement

Featured in: "Example 4. Creating and Modifying Box Plots" on page 302and "Example 3. Rotating Plot Symbols Through the Color List" on page 299

Restriction: Partially supported by Java and ActiveX

INTERPOL=BOX<*option(s)*><00...25>
produces box and whisker plots. 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). By default, INTERPOL=BOX. 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 the 75th sample percentiles.) Any value more extreme than this is marked with a plot symbol.

Values for *option(s)* 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 high/low extremes. INTERPOL=BOX00 is *not* the same as the default, INTERPOL=BOX.

01 1st percentile low, 99th high

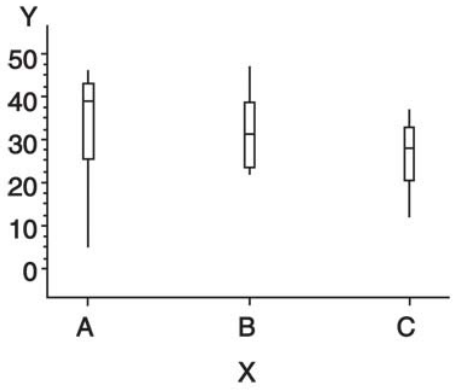
05 5th percentile low, 95th high

10 10th percentile low, 90th high

25 25th percentile low, 75th high; since the box extends from the 25th to the 75th percentile, no whiskers are produced.

Figure 14.15 on page 257 shows the type of plot INTERPOL=BOX produces.

Figure 14.15 Box Plot



Note: If you use the HAXIS= or VAXIS= options in the PLOT statement or the ORDER= option in an AXIS definition 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= option on page 266 △

You cannot use the GPLOT procedure PLOT statement option AREAS= with INTERPOL=BOX.

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 BWIDTH=, 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;
```

Note: When using DEVICE=JAVA and 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=BOX<*option(s)*><00...25>

Featured in: "Example 4. Creating and Modifying Box Plots" on page 302

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 14.16 on page 259.

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.

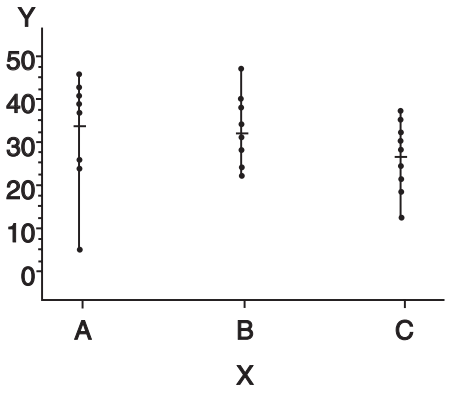
T — adds tops and bottoms to each line.

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 14.16 on page 259 shows the type of plot INTERPOL=HILO produces. Plot symbols in the form of dots have been added to this figure.

Figure 14.16 High-Low Plot



To increase the thickness of all lines generated by the INTERPOL=HILO option, use the WIDTH= option.

Note: If you use the HAXIS= or VAXIS= options in the PLOT statement or the ORDER= option in an AXIS definition 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 option MODE= on page 266. △

When using DEVICE=JAVA and 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=HILO<C><*option*>

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294

Restriction: Partially supported by Java

INTERPOL=JOIN

connects data points with straight lines. Points are connected in the order 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=JOIN

See also: the SKIPMISS on page 1358 option and "Missing Values" on page 1331

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.

INTERPOL=*map*/*plot-pattern*
I=*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
ME
an empty pattern. EMPTY and E are valid aliases.
The Java applet does not support this option.

MSOLID
MS
a solid pattern. SOLID and S are valid aliases

M*density*<*style*<*angle*>>
a shaded pattern. (The Java applet does not support this option.)
Density specifies the density of the pattern's shading:

1...5	1 produces the lightest shading and 5 produces the heaviest.

Style specifies the direction of pattern lines:

N	parallel lines (the default)
X	crosshatched lines.

Angle specifies the starting angle for parallel or crosshatched lines:

0...360	the degree at which the parallel lines are drawn. By default, *angle* is 0 (lines are parallel to the horizontal axis).

The INTERPOL=*map*/*plot-pattern* option only works 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.

Alias: I=L<*degree*><P><S>

See also: the "PATTERN Statement" on page 240

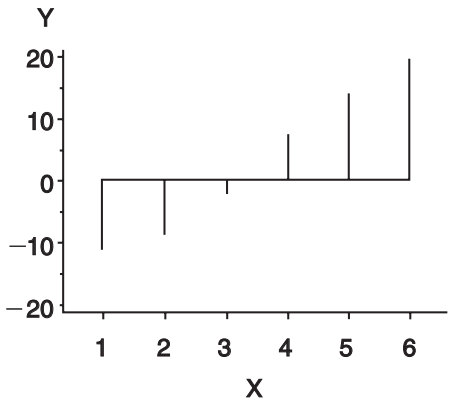
Featured in: "Example 5. Filling the Area between Plot Lines" on page 304

Restriction: Partially supported by Java

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 14.17 on page 261 shows the type of plot INTERPOL=NEEDLE produces. Plot symbols are not displayed in this figure.

Figure 14.17 Needle Plot



You cannot use the PLOT statement option AREAS= with INTERPOL=NEEDLE.

Alias: I=NEEDLE

INTERPOL=NONE
I=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.

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, regression lines are not forced through plot origins and confidence limits are not displayed.

Type specifies 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, the SAS/GRAPH uses a quadratic equation by default for the interpolation whereas the Java and ActiveX device drivers might pick a cubic solution to use. △

By default, *type* is L. The regression line is drawn in the line type specified in the LINE= option. By default, the type of the regression line is 1.

Note: You must specify *type* if you use either 0, or CLI, or CLM. △

To force the regression line through a (0,0) origin, specify:

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: To 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

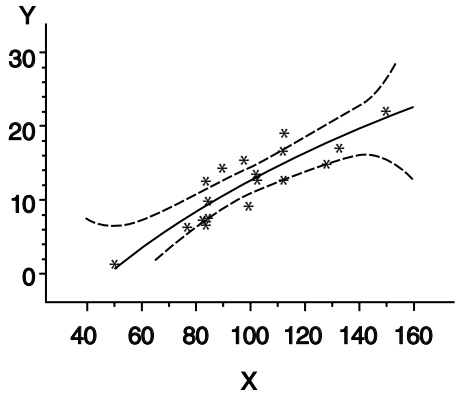
CLI displays confidence limits for individual predicted values.

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 14.18 on page 262 shows the type of plot INTERPOL=RCCLM95 produces (cubic regression analysis with 95% confidence limits).

Figure 14.18 Plot of Regression Analysis and Confidence Limits



Alias: I=R<*type*><0><CLM | CLI<50...99>>

Featured in: Example 4 on page 1372

Restriction: Partially supported by Java

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 (Reinsch 1967)*. 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.

Restriction: Not supported by Java

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 smoothes a plot line using a cubic spline method with continuous second derivatives (Pizer 1975)**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.

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.

Note: When points on the graph are close together and a spline interpolation is used, the Java applet is unable to draw some line types correctly. △

△

Alias: I=SPLINE<P><S>

---

* Reinsch, C.H. (1967), "Smoothing by Spline Functions," *Numerische Mathematik*, 10, 177–183.

** Pizer, Stephen M. (1975), *Numerical Computing and Mathematical Analysis*, Chicago: Science Research Associates, Inc., Chapter 4.

INTERPOL=STD<1 | 2 | 3><*variance*><*option(s)*>
specifies that a solid line connect the mean Y value with ± 1, 2, or 3 standard deviations for each X.

Note: By default, two standard deviations are used. △

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:

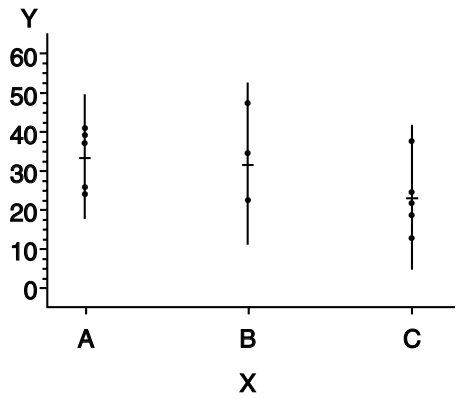
M	computes $s_{\overline{y}}$,
P	computes sample variances using a pooled estimate, as in a one-way ANOVA model.

In addition, specify one of these values for *option(s)*:

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.
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 14.19 on page 264 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 14.19 Plot of Standard Deviations



Note: By default, the vertical axis ranges from the minimum to the maximum Y value in the data. If the requested number of standard deviations from the mean covers a range of values that exceeds the maximum or is less than the minimum, 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. △

If you restrict the range of axis values by using the HAXIS= or VAXIS= option in a PLOT statement or 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= option on page 266 option.

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=JAVA and 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=STD<1 | 2 | 3><*variance*><*option(s)*>

Restriction: Partially supported by Java

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.

Note: When a step is retraced in order to locate its center point, the GIF, JPEG, PNG, ACTXIMG, Java, and JAVAIMG devices treat this as effectively not drawing that part of the step at all. ActiveX, however, draws each part of the step—resulting in a somewhat different graph. △

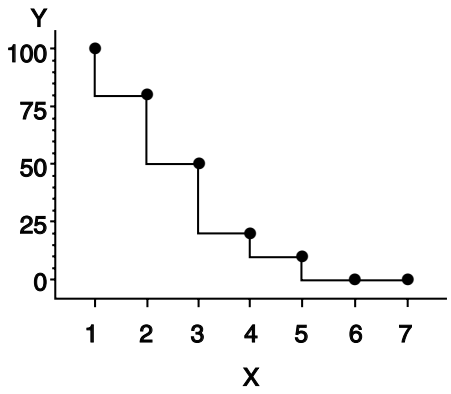
In addition, specify one or both of these:

J produces steps joined with a vertical line.

S sorts unordered data by the independent variable before plotting.

Figure 14.20 on page 265 shows the type of plot INTERPOL=STEPJR produces. Plot symbols in the form of dots have been added to this figure.

Figure 14.20 Step Plot



Alias: I=STEP<*placement*><J><S>

LINE=*line-type*
L=*line-type*

specifies the line type of the plot line in the GPLOT procedure, or the contour line in the GCONTOUR procedure:

1 a solid line.

2...46 a dashed line.

Line types are shown in Figure 14.22 on page 277. By default, LINE=1.

Note: This option overrides the LineStyle attribute in graph styles. Neither the Java applet nor ActiveX control supports GCONTOUR. △

Restriction: Partially supported by Java and ActiveX

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.

If you control the range of values displayed on an axis by using HAXIS= and VAXIS= in the GPLOT procedure, or ORDER= in an AXIS definition, any data points that lie outside the range of the axes are discarded before interpolation is applied to the data. 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.

Restriction: Not supported by Java and partially supported by ActiveX

See also: "Values Out of Range" on page 1331

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 descriptions labels points with the Y value. NONE suppresses the point labels. *Label-description(s)* can be used to change the variable whose values are used to label points, and to change features of the label text, such as the color, font, or size of the text.

Note: If you do not specify a color on a SYMBOL statement, the symbol definition is rotated through the color list before the next SYMBOL statement is used. Thus, if 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. △

Label-description(s) can be one or more of these:

COLOR=*text-color*

specifies the color of the label text. The default is the first color from the color list.

Alias: C=*text-color*

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 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 for details on 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=*font* | NONE

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=*text-height <units >*

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.

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.

Alias: J=C | L | R

"*#var*" | "*#x*:*#y* <$*char*>" | "*#y*:*#x* <$*char*>"
specifies the variable or variables whose values label the plot points. The variable specification must be enclosed in either single or double quotation marks. The first specified variable must be prefixed with a pound sign (#). If a second variable is specified, it must be prefixed with a colon and a pound sign (:#). When you specify both the X and Y 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 the X and Y variables, in either order. The order that you specify X and Y in the variable specification determines the order that the values are displayed in the label. The following option specifies variables HEIGHT and WEIGHT; in the label, the value for HEIGHT is displayed, followed by the value for WEIGHT:

```
POINTLABEL=("#height:#weight")
```

By default when you specify both the X and Y variables, a colon (:) displays in the label to separate the values in each label. To change the character that displays 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.

Note: Specifying a delimiting character with the $ only changes the character that displays in the label. It does not change the syntax of the

variable specification, which requires a colon and pound sign (:#) to precede the second variable. △

Note: There is a sixteen 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. △

When creating output using the ActiveX or Java devices, the variables that you specify in the POINTLABEL= option must be for the plot's X and Y variables. Specifying any other variables causes unexpected labeling.

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.

Restriction: Partially supported by Java and ActiveX

REPEAT=*number-of-times*
specifies the number of times that a SYMBOL definition is applied before the next SYMBOL definition is used. By default, REPEAT=1.

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, and then the entire group generated by this cycle 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.

  Neither the Java applet nor ActiveX control supports GCONTOUR.

Alias: R=*number-of-times*

See also: "Using the SYMBOL Statement" on page 272

Restriction: Partially supported by Java and ActiveX

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. By default, STEP=65PCT.

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 percent is ignored and 10 percent is used instead. △

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 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 will never be less.

In general, to increase the number of labels from the default, reduce the value of *distance*.

If the procedure cannot write the label at a particular location on the contour, for example because 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, since this forces the procedure to select smoother labeling locations, which might not be available on some contours.

Featured in: Example 2 on page 1116

Restriction: Not supported by Java and ActiveX

VALUE=*special-symbol* | *text-string* | 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 symbol. The default symbol is a square if you use the ActiveX or Java devices and a PLUS sign for other devices. If you specify a SYMBOL statement, but do not specify the VALUE= option, plot symbols are suppressed.

  Note: 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. △
- 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=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 Figure 14.21 on page 271. The special symbol table can be used only if the FONT= option is not used or a null value is specified:

```
font=,
```

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 quotes.

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 quotes or blanks, enclose the string in single quotes.

For example, if you specify this statement, the plot symbol is the word "plus" instead of the symbol +:

```
symbol font=swiss value=plus;
```

Java and ActiveX support the following characters from the marker font for *special-symbol*:

Table 14.2 Marker-font symbols supported by Java and ActiveX

Character	Aliases
Marker	Cone, Pyramid, Default
Square	Cube
Star	
Circle	Sphere, Dot, Balloon
Plus	Cross
Flag	Y
X	
Prism	Z
Spade	“
Heart	#
Diamond	$
Club	%
Hexagon	Paw
Cylinder	Hash

Note: If you do not use a SYMBOL statement to specify a color for each symbol, but you do specify a color list in a GOPTIONS statement, then Java and ActiveX assign colors to symbols differently than do the other device drivers. To ensure consistency on all devices, 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. △

Note: The VALUE option overrides the MarkerSymbol attribute in graph styles. △

See also: the option FONT= on page 256 and “Specifying Plot Symbols” on page 274.

Alias: V=*special-symbol* | *text-string* | NONE

Featured in: “Example 3. Rotating Plot Symbols Through the Color List” on page 299, “Example 4. Creating and Modifying Box Plots” on page 302, and Example 2 on page 1116

Restriction: Partially supported by Java and ActiveX

WIDTH=*thickness-factor*
specifies the thickness of interpolated lines (GPLOT) or contour lines (GCONTOUR), where *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.

Note: 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 BWIDTH=.

Java and ActiveX do not provide the same measure of control for width as SAS/GRAPH device drivers. Measurements are translated to pixels rather than a percentage. For DEVICE=JAVA and DEVICE=ACTIVEX the maximum width is 6. △

Style Reference: LineThickness attribute of the GraphAxisLines element

Alias: W=*thickness-factor*

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294 and "Example 4. Creating and Modifying Box Plots" on page 302

Restriction: Partially supported by Java and ActiveX

Figure 14.21 Special Symbols for Plotting Data Points

VALUE=	Plot Symbol	VALUE=	Plot Symbol
PLUS	+	% (percent)	♣
X	×	& (ampersand)	♧
STAR	✳	' (single quote)	⚜
SQUARE	□	= (equals)	☆
DIAMOND	◇	- (hyphen)	⊙
TRIANGLE	△	@ (at)	♁
HASH	#	* (asterisk)	♀
Y	Y	+ (plus)	⊕
Z	Z	> (greater than)	♂
PAW	∴	. (period)	♃
POINT	·	< (less than)	♄
DOT	●	, (comma)	☉
CIRCLE	○	/ (slash)	♆
_ (underscore)	⊏	? (question mark)	♇
" (double quote)	♤	((left parenthesis)	☾
# (pound sign)	♡	) (right parenthesis)	☌
$ (dollar sign)	◇	: (colon)	✳

Note: The words or special characters in the VALUE= column are entered exactly as shown. △

Using the SYMBOL Statement

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 273 and “Using Generated Symbol Sequences” on page 277.

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. If you 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 which 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

If you specify consecutively numbered SYMBOL statements and you 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 on 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 two definitions:

  ```
  symbol1 value=star color=green;
  symbol2 value=square color=yellow;
  ```

- 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 on specifying colors for symbol definitions, see “Using Color” on page 275.

If you do not use color to limit a SYMBOL statement to a single symbol definition, SAS/GRAPH generates multiple symbol definitions from that statement by rotating the current definition through the color list (for more details, see “Using Generated Symbol Sequences” on page 277). Because SAS/GRAPH uses symbol definitions in the order they are generated, this means that the *n*th symbol definition applied to a graph does not necessarily correspond to the SYMBOL*n* statement.

For example, assuming that no color is specified on the CSYMBOL= graphics option, these statements generate four definitions:

```
goptions colors=(red blue green);
symbol1 value=star;
symbol2 value=square color=yellow;
```

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, and SYMBOL2 defines the 4th:

Sequence Number	Source	Characteristics: Color	Symbol
1	SYMBOL1	red	star
2	SYMBOL1	blue	star

Sequence Number	Source	Characteristics: Color	Symbol
3	SYMBOL1	green	star
4	SYMBOL2	yellow	square

In this case, if a graph needs only three symbols, the SYMBOL2 definition is not used.

To make the *n*th applied symbol definition correspond to the SYMBOL*n* statement, limit each SYMBOL statement to a single color, using one of the techniques listed at the beginning of this section.

Setting Definitions for PROC GPLOT and PROC GBARLINE

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 Figure 14.21 on page 271
- characters from symbol fonts
- text strings

By default, the plot symbol is the + symbol. To specify a special symbol, use the VALUE= option to specify a name or a character from Figure 14.21 on page 271:

```
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 277.

To use plot symbols other than those in Figure 14.21 on page 271, 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= on page 269 and FONT= on page 256.)

This table illustrates some of the ways you can define a plot symbol:

Definition	Plot Symbol
symbol1 value=plus;	+
symbol2 value=+;	⊕
symbol3 font=swiss value=plus;	plus

Definition	Plot Symbol
symbol4 font=marker value=U;	
symbol5value=""";	

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 If you want 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) and 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

Generally, there are two ways to explicitly specify color for SYMBOL statements:

- specify colors on the SYMBOL statements
- specify a 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 277.

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 this statement to assign 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 277.

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;
```

In this example, both SYMBOL statements use green.

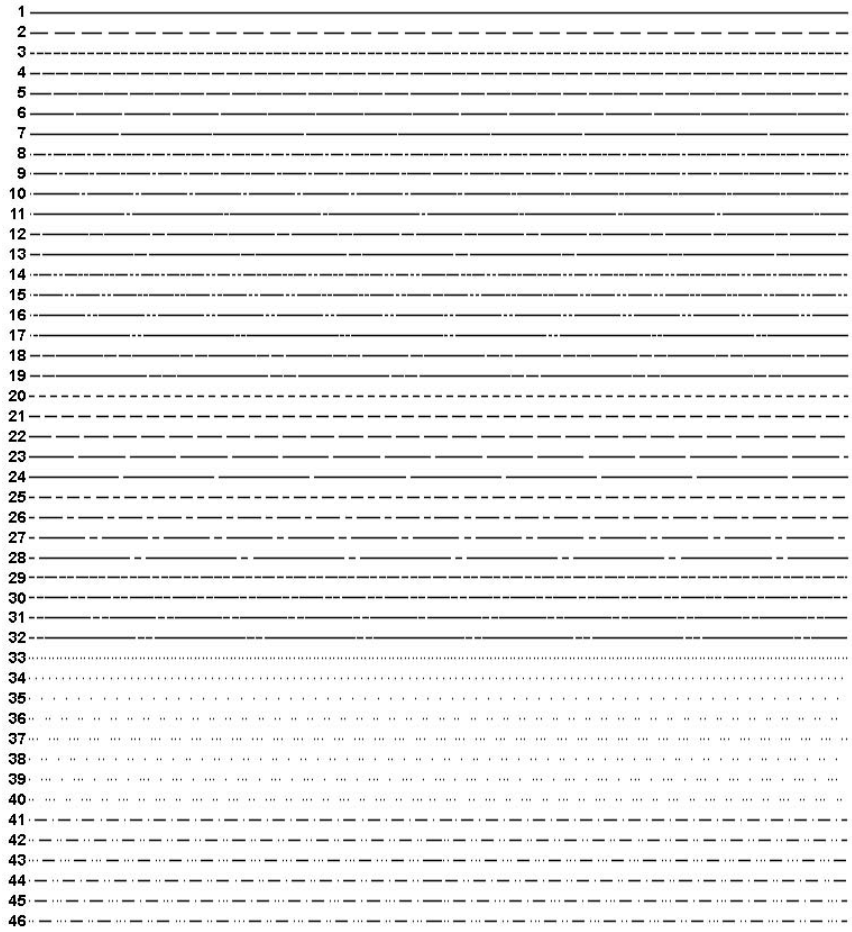
CSYMBOL= is overridden by any of the SYMBOL statement color options. See “Using Color” on page 275 for details.

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 14.22 on page 277 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 14.22 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

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 line type for the GCONTOUR procedure (assuming that GCONTOUR does not specify the needed line types). In this case, a default plot symbol or 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.

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 and 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.

If a color has been specified with the CSYMBOL= option in the GOPTIONS statement, 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 If a SYMBOL statement does not specify color, and if the CSYMBOL= graphics option is not used, the symbol definition is rotated through every color in the color list before the next SYMBOL definition is used:

```
goptions colors=(blue red green);
symbol1 cv=red i=join;
symbol2 i=spline v=dot;
symbol3 cv=green v=star;
```

Here, 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:

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

Sequence Number	Source	Characteristics: Color	Symbol	Interpolation
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. For example, these statements 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;
```

Here, 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:

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

TITLE, FOOTNOTE, and NOTE Statements

Control the content, appearance, and placement of text.

Used by: GANNO, GAREABAR, GBARLINE, GCHART, GCONTOUR, GFONT, GIMPORT, GMAP, GPLOT, GRADAR, GREPLAY, GSLIDE, G3D, and G3GRID
Global: TITLE and FOOTNOTE
Local: NOTE

Syntax

TITLE<1...10> <*text-argument(s)*>;

FOOTNOTE<1...10> <*text-argument(s)*>;

NOTE <*text-arguments(s)*>;
text-argument(s) can be one or more of these:
"*text-string*"
text-options (text options must precede text-string.)

text-options can be one or more of the following, in any order:

- appearance options
  - COLOR=*color*
  - FONT=*font*
  - HEIGHT=*text-height*<*units*>
- placement and spacing options
  - JUSTIFY=LEFT | CENTER | RIGHT
  - LSPACE=*line-space*<*units*>
  - MOVE=(*x,y*)<*units*>
  - WRAP
- baseline angling and character rotation options
  - ANGLE=*degrees*
  - LANGLE=*degrees*
  - ROTATE=*degrees*
- boxing, underlining, and line drawing options
  - BCOLOR=*background-color*
  - BLANK=YES
  - BOX=1...4
  - BSPACE=*box-space*<*units*>
  - DRAW=(*x,y...,x-n,y-n*)<*units*>
  - UNDERLIN=0...3
- linking option
  - LINK= *"URL"*

Options

When the syntax of an option includes *units*, use one of these:

CELLS	character cells
CM	centimeters
IN	inches
PT	points
PCT	percentage 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.

ANGLE=*degrees*
A=*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 and, consequently, might 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 293.

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:

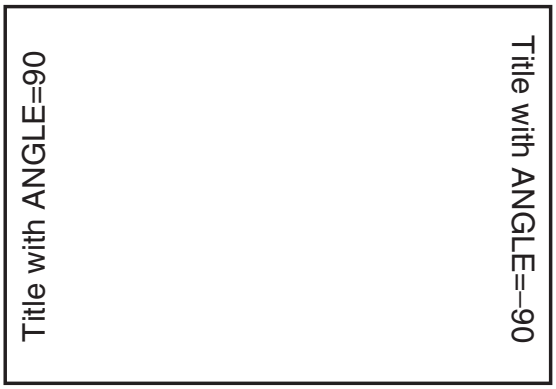
- *With the TITLE statement:*

  Figure 14.23 on page 281 shows how ANGLE=90 degrees or ANGLE=–90 degrees positions and rotates 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 14.23 Positioning Titles with the ANGLE= Option



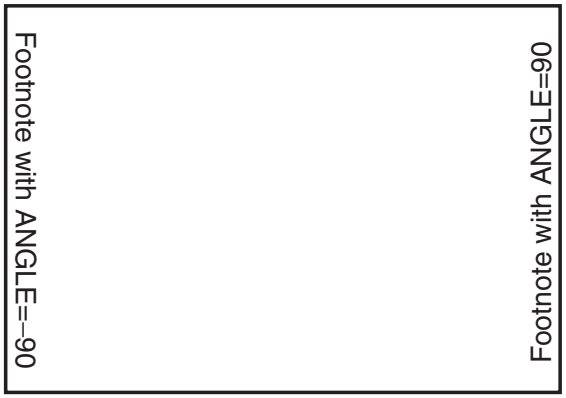
- *With the FOOTNOTE statement:*

  Figure 14.24 on page 282 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 14.24 Positioning Footnotes with the ANGLE=Option



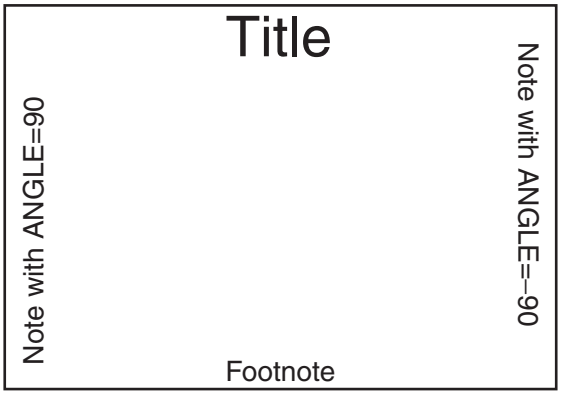
□ *With the NOTE statement:*

Figure 14.25 on page 282 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 14.25 Positioning Notes with the ANGLE= Option



See also: the options LANGLE= on page 287 and ROTATE= on page 290

Featured in: "Example 6. Enhancing Titles" on page 307

Restriction: Not supported by Java and ActiveX

BCOLOR=*background-color*
specifies the background color of a box produced by the BOX= option. If you omit BOX=, BCOLOR= is ignored. 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=.

Note: The BCOLOR= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= optionwith absolute coordinates. See "Using Options That Can Reset Other Options" on page 293 for details. △

Alias: BC=*background-color*

See also: the option BOX= on page 283

Featured in: "Example 6. Enhancing Titles" on page 307.

BLANK=YES
protects the box and its contents from being overwritten by any subsequent graphics elements 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.

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 293 for details. △

Alias: BL=YES

See also: the option BOX= on page 283

Featured in: "Example 6. Enhancing Titles" on page 307

Restriction: Not supported by Java and ActiveX

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 either:

- the color specified by the COLOR= option in the statement
- the default text color.

The COLOR= option affects only the frame of the box. To color the background of the box, use the BCOLOR= option.

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.

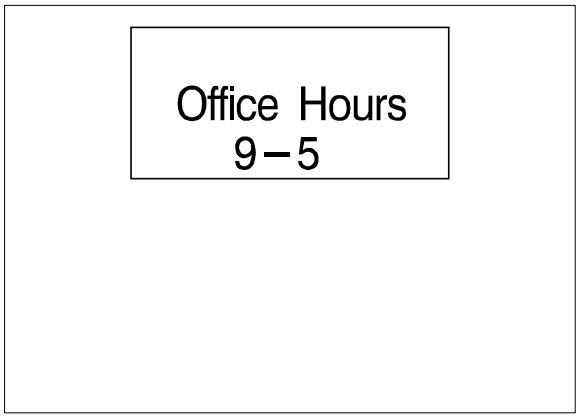
To draw a box around multiple lines of text, you can either

- Use the MOVE= option with relative coordinates to position the lines of text where you want them and enclose them with the BOX= option. For example, this statement produces the boxed note shown in Figure 14.26 on page 284:

```
note font=swiss justify=center box=3
   "Office Hours"      move=(40pct,-12pct) "9-5";
```

- Use the DRAW= option to draw the box and do not use the BOX= option.

Figure 14.26 Using the BOX= Option and the MOVE= Option to Box Multiple Lines of Text



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 293 for details. △

Alias: BO=1...4

See also: the options BCOLOR= on page 283, BLANK= on page 283, and BSPACE= on page 284.

Featured in: "Example 6. Enhancing Titles" on page 307

Restriction: Not supported by Java and ActiveX

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. If the BOX= option is not used, the BSPACE= option is ignored.

The spacing is uniform around the box. For example, BSPACE=.5IN leaves one-half inch of space between the text and the top, bottom, and sides of the box.

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 293 for details. △

Alias: BS=*box-space<units>*

See also: the option BOX= on page 283.

Restriction: Not supported by Java and ActiveX.

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=*color*

See also: the option BCOLOR= on page 283, and "Controlling Titles and Footnotes with Java and ActiveX Devices in HTML Output" on page 194

DRAW=(*x,y...,x-n,y-n*)<*units*>
draws lines anywhere on the graphics output area using *x* and *y* as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

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 and 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=(*x,y...,x-n*, *y-n*)<*units*>

Restriction: Not supported by Java and ActiveX

FONT=*font*
specifies the font for the subsequent text. See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 for details on 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.

Note: Font names greater than eight characters in length must be enclosed in quotation marks. △

Note: If the TITLE or FOOTNOTE is being output through 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. △

Alias: F=*font*

See also: "Controlling Titles and Footnotes with Java and ActiveX Devices in HTML Output" on page 194

Featured in: "Example 6. Enhancing Titles" on page 307

HEIGHT=*text-height<units>*
specifies the height of text characters in number of units. By default, HEIGHT=1. 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 TITLE1 statement:*
  1. the HTITLE= option in a GOPTIONS statement
  2. the HTEXT= option in a GOPTIONS statement
  3. the default value, 2.

  By default, a TITLE1 title is twice the height of all other titles.
- *for all other TITLE statements and the FOOTNOTE and NOTE statements:*
  1. the HTEXT= option in a GOPTIONS statement
  2. the default value, 1.

Note: The Java applet and ActiveX control allow you to control the relative height of text with the HEIGHT= option, but not the absolute height in terms of specific units. △

Alias: H=*text-height<units>*

See also: "Controlling Titles and Footnotes with Java and ActiveX Devices in HTML Output" on page 194

Featured in: "Example 1. Ordering Axis Tick Marks with SAS Date Values" on page 294 and "Example 6. Enhancing Titles" on page 307

Restriction: Partially supported by Java and ActiveX

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. If you produce a block of text by specifying the same justification for multiple text strings, and then change the justification for an additional text string, that text is placed on the same line as the first string specified in the statement.

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 293 for details. △

Alias: J=L | C | R

Featured in: "Example 3. Rotating Plot Symbols Through the Color List" on page 299

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 and consequently can 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 14.27 on page 287 shows how LANGLE=90 degrees and LANGLE=–90 degrees positions and rotates titles.

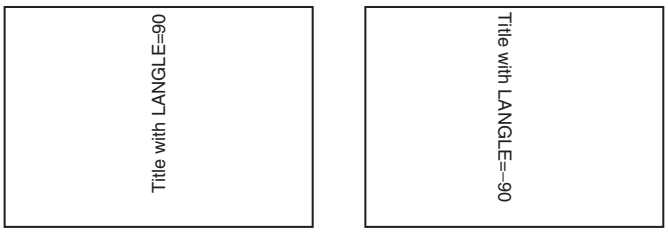
LANGLE=90

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 14.27 Positioning Titles with the LANGLE= Option



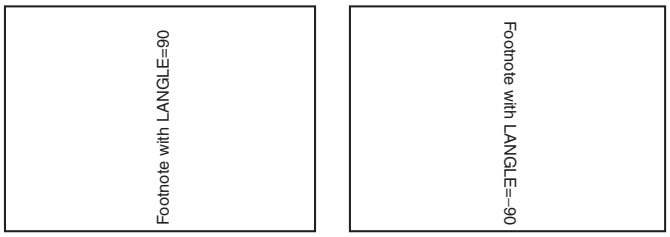
□ *With the FOOTNOTE statement:*

Figure 14.28 on page 288 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 14.28 Positioning Footnotes with the LANGLE= Option



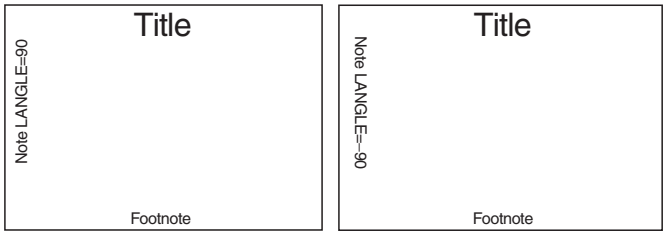
□ *With the NOTE statement:*

Figure 14.29 on page 288 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 14.29 Positioning Notes with the LANGLE= Option



Alias: LA=*degrees*

See also: the option ANGLE= on page 281

Restriction: Not supported by Java and ActiveX

LINK= *"URL"*
specifies a uniform resource locator (URL) that a title or footnote links to.

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 on page 290.

Note: If the title or footnote is being output through an ODS markup destination (such as HTML) and the corresponding ODS option NOGTITLE or NOGFOOTNOTE is 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 also: "Controlling Where Titles and Footnotes are Rendered" on page 194

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. By default, LSPACE=1.

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 293 for details. △

Alias: LS=*line-space <units>*

Restriction: Not supported by Java and ActiveX

MOVE=(*x*,*y*) *<units>*
positions subsequent text or lines anywhere on the graphics output area using *x* and *y* as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

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 relative coordinates begins with 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 and 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.

The MOVE= option overrides a JUSTIFY= option specified for the same text string.

If a NOTE, FOOTNOTE, or TITLE statement uses the MOVE= option to position the text so that the statement does not use its default position, the text of the next NOTE, FOOTNOTE, or TITLE statement occupies the unused position and no blank lines are displayed.

Note: If you specify the MOVE= option with at least one absolute coordinate and 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 293 for details △

Alias: M=(*x*,*y*) <*units*>

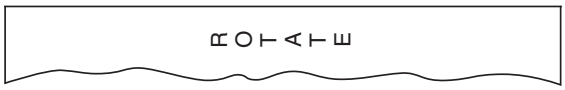
Featured in: "Example 2. Specifying Logarithmic Axes" on page 297 and "Example 6. Enhancing Titles" on page 307

Restriction: Not supported by Java and ActiveX

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 14.30 on page 290 shows how characters are positioned when ROTATE=90 is used with the default (horizontal) baseline.

Figure 14.30 Tilting Characters with the ROTATE= Option



Alias: R=*degrees*

See also: the option ANGLE= on page 281

Featured in: "Example 6. Enhancing Titles" on page 307

Restriction: Not supported by Java and ActiveX

text-string(s)
is one or more strings up to 200 characters. You must enclose text strings in single or double quotation marks. The text appears exactly as you type it in the statement, including uppercase and lowercase characters and blanks.

To use single quotation marks or apostrophes within the title, you can either

- 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";
```

Because FOOTNOTE, NOTE, and TITLE statements concatenate all text strings, the strings must contain the correct spacing. With a series of strings, add blanks at the beginning of a text string rather than at the end, as in this example:

```
note color=red "Sales:" color=blue " 2000";
```

With some fonts, you produce certain characters by specifying a hexadecimal value. A trailing **x** identifies a string as a hexadecimal value. For example, this statement* produces the title **Profits Increase £ 3,000**:

* This statement assumes you are using a U.S. key map.

```
title font=swiss "Profits Increase " "18'x "3,000";
```

For more information see “Specifying Special Characters Using Character and Hexadecimal Codes” on page 160.

In addition, you can embed one or more of the following in the string:

#BYLINE
substitutes the entire BY line without leading or trailing blanks for #BYLINE in the text string, and 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 in the footnote, note, or title 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.

Featured in: “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 309 and “Example 9. Combining Graphs and Reports in a Web Page” on page 315

#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 in the footnote, note, or title 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.

A BY variable name displayed in a title, note, or footnote is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

For more information , see “Substituting BY Line Values in a Text String” on page 294

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. See the option ROTATE= on page 290.

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.

Note: 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 293 for details.

Note: The Java applet and ActiveX control underline text when the UNDERLIN= option is specified, but they do not vary the thickness of the line. △

△

Alias: U=

Featured in: "Example 6. Enhancing Titles" on page 307

Restriction: Partially supported by Java and ActiveX

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.

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 they modify. See "Using Multiple Options" on page 293.

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, 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.

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 they modify. See "Using Multiple Options" on page 293.

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 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";
```

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 Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 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. △

If in a TITLE, FOOTNOTE, or NOTE statement, before the first text string, you use an option that can be reset (such as the UNDERLIN= option) and before the second string 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

To use the #BYVAR and #BYVAL options, insert the option in the text string at the position you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character, 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.

If you use a #BYVAR or #BYVAL specification for a variable that is not named in the BY statement (such as #BYVAL2 when there is only one BY-variable or #BYVAL(ABC) when ABC is not a BY-variable or does not exist), or if there is no BY statement at all, the substitution for #BYVAR or #BYVAL 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 “BY Statement” on page 216.

Note: This feature is not available in the DATA Step Graphics Interface or in the Annotate facility since BY lines are not created in a DATA step. △

Example 1. Ordering Axis Tick Marks with SAS Date Values

Features:

AXIS statement options:

LABEL=
OFFSET=
ORDER=

FOOTNOTE statement option:

JUSTIFY=

SYMBOL statement options:

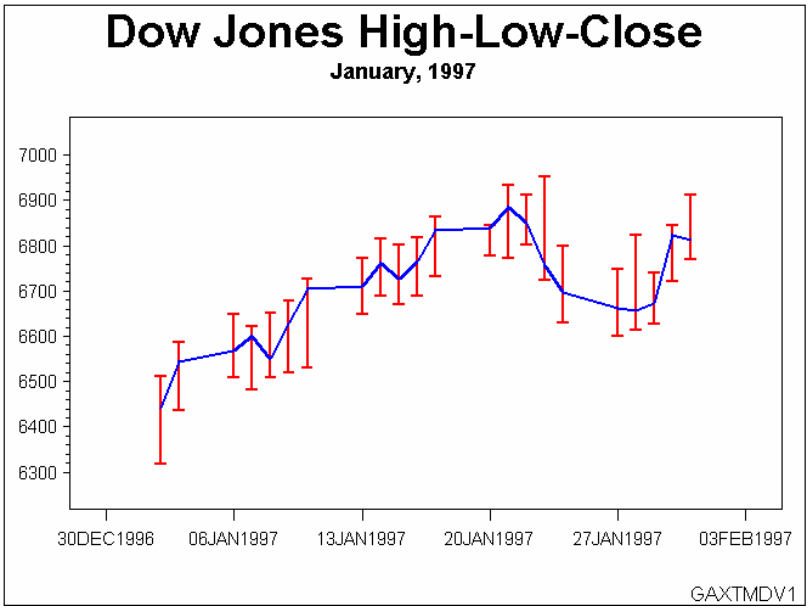
INTERPOL=

WIDTH=

GOPTIONS statement options:

BORDER

Sample library member: GAXTMDV1



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.

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:

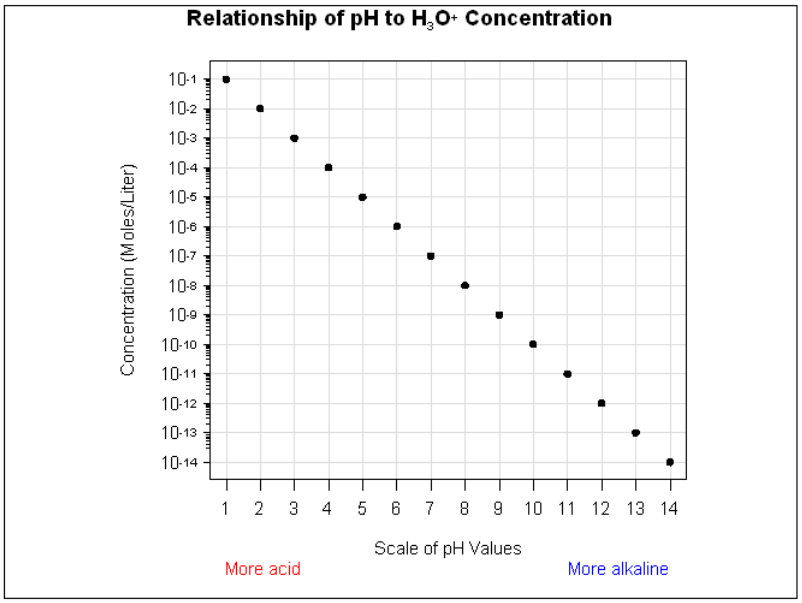
- LABEL=
- LENGTH=
- LOGBASE=
- LOGSTYLE=
- MAJOR=
- MINOR=
- VALUE=

TITLE statement option:

- MOVE=

GOPTIONS statement options:

- GUNIT



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.

In addition, this example shows how the TICK= parameter of the VALUE= option modifies individual tick marks.

The example uses the MOVE= option in a TITLE statement to position the title's subscript and superscript text.

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 CONCENTR option 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 title and footnote. 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=2 "O"
       move=(+0,+.75) h=2 "+"
       move=(-0,-.75) h=2  " Concentration";
```

Define symbol characteristics.

```
symbol value=dot color=black height=2;
```

Define characteristics for horizontal axis. 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 characteristics for vertical axis. 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=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:

GOPTIONS statement options:

COLORS=

LEGEND statement options:

LABEL=

SYMBOL statement options:

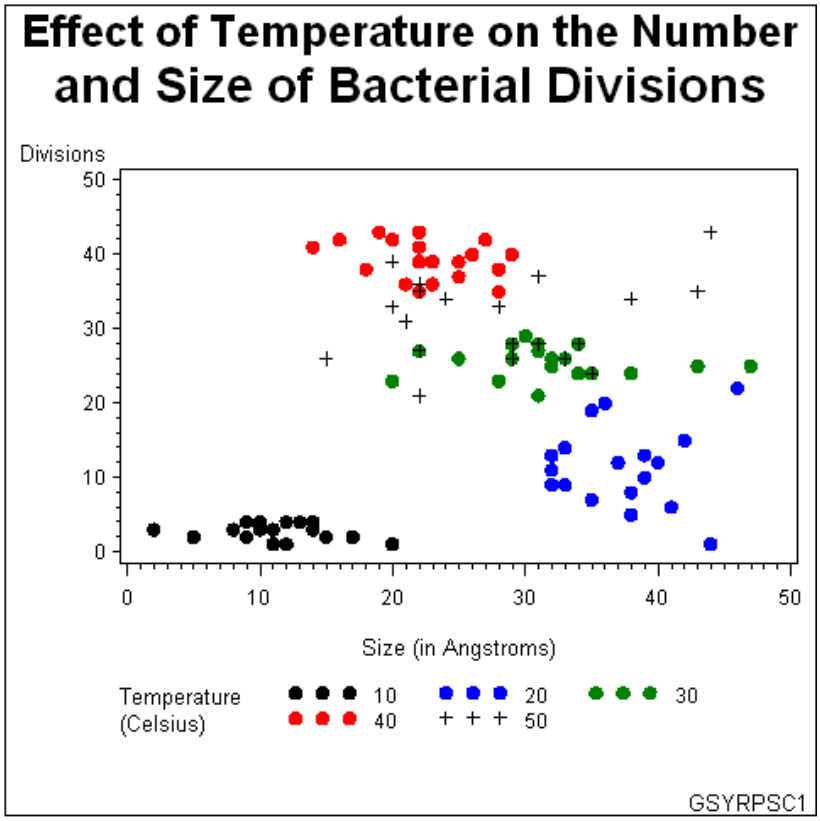
VALUE=

TITLE statement option:

JUSTIFY=

HEIGHT=

Sample library member: GSYRPSC1



This example specifies a plot symbol on 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.

Set the graphics environment. The COLORS= option specifies the color list. This list is used by the SYMBOL 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.*J= breaks the title into two lines. H= specifies the size of the title.

```
title1 "Effect of Temperature on the Number"
       j=c  h=2 "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 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:

SYMBOL statement options:

BWIDTH=

CO=

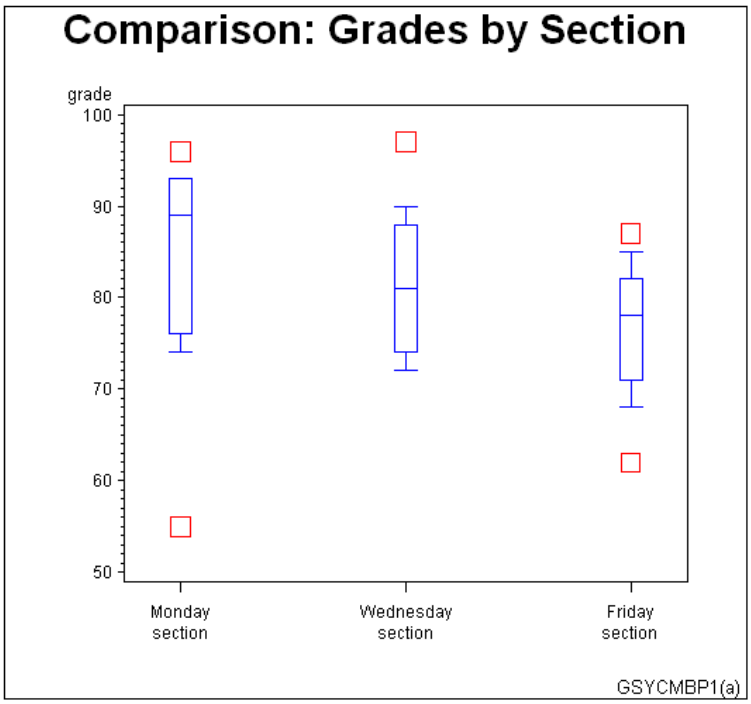
CV=

HEIGHT=

INTERPOL=

VALUE=

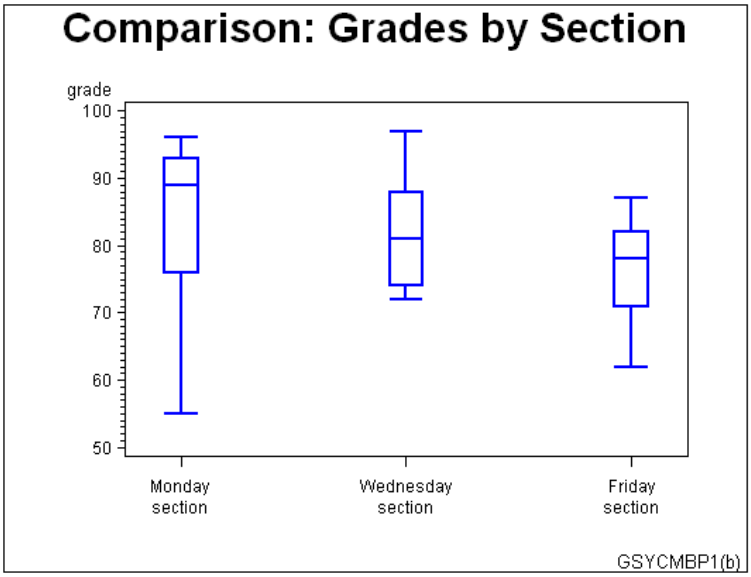
Sample library member: GSYCMBP1



This example shows how to create box plots and how to specify SYMBOL definitions so 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.

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 and 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, which 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.



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:

AXIS statement option:

ORDER=

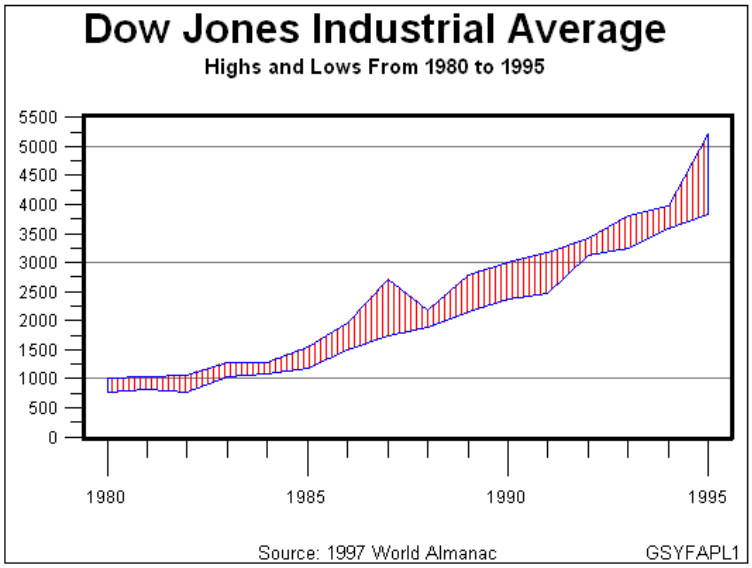
SYMBOL statement options:

CO=

CV=

INTERPOL=

Sample library member: GSYFAPL1



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.

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/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:

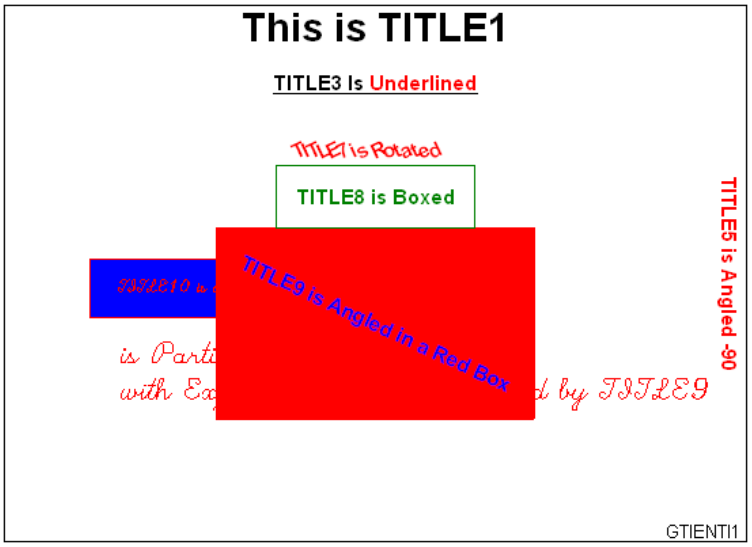
GOPTIONS statement options:

BORDER

TITLE statement options:

BCOLOR=
BLANK=
BOX=
COLOR=
FONT=
HEIGHT=
MOVE=
ROTATE=
UNDERLIN=

Sample library member: GTIENTI1



This example illustrates some ways you can format title text. The same options can be used to format footnotes.

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=red
       " Underlined";
```

Define TITLE5. The ANGLE= option tilts the line of text clockwise 90 degrees and places it at the right edge of the output.

```
title5 color=red
       angle=-90
       "TITLE5 is Angled -90";
```

Define TITLE7. The ROTATE= option rotates each character in the text string at the specified angle. The HEIGHT= option sets the height of the text.

```
title7 height=4
       color=red
       rotate=25
       "TITLE7 is Rotated";
```

Define TITLE8. The BOX= option draws a green box around the text.

```
title8 color=green
       box=1
       "TITLE8 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 color of the box background. The second COLOR= option specifies the text color.

```
title9 color=red
       box=3
       blank=yes
       bcolor=red
       color=blue
       move=(70,20)
       angle=-25
       "TITLE9 is Angled in a Red Box";
```

Define TITLE10. In this statement, the BOX= option draws a box around the first text string. The BOX= option is turned off by the MOVE= option that uses absolute coordinates and causes a text break.

```
title10 color=red
        box=1
        bcolor=blue
        move=(60,20)
        font=script
        "TITLE10 is in Script and "
        move=(60,15)
        height=2
        "is Partially Boxed, Positioned"
        move=(60,10)
                height=2
        "with Explicit Moves, and Overlaid by TITLE9"
                ;
```

Define footnote.

```
footnote justify=right "GTIENTI1  ";
```

Display titles and footnote. All existing titles and footnotes 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=
STYLE=
VALUE=

BY statement

OPTIONS statement option:

NOBYLINE

PATTERN statement option:

COLOR=

TITLE statement:

#BYVAL

Sample library member: GBYGMSC1

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 Display 14.1 on page 312, Display 14.2 on page 312, and Display 14.3 on page 313 following the code, 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 the bars.

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 the character that causes an automatic line break in the axis values.

```
axis2 label=none
      split=" ";
```

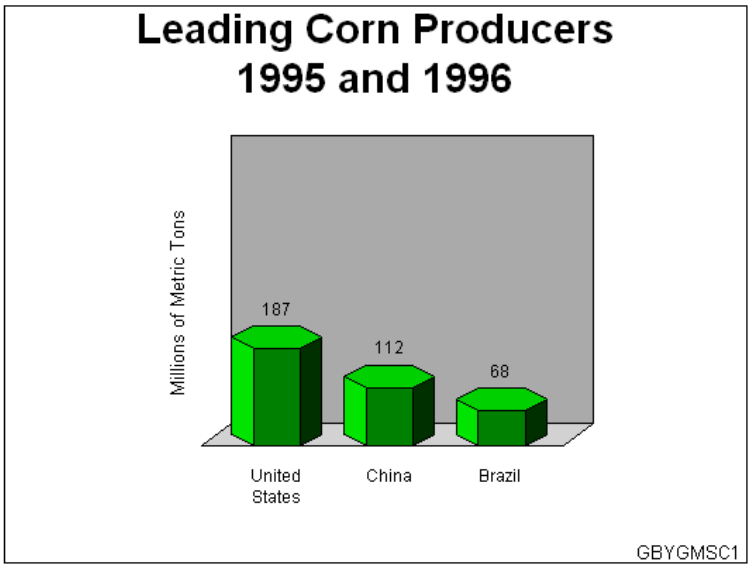
Sort data according to values of BY variable. The data must be sorted before running PROC GCHART with the BY statement.

```
proc sort data=grainldr out=temp;
     by type;
run;
```

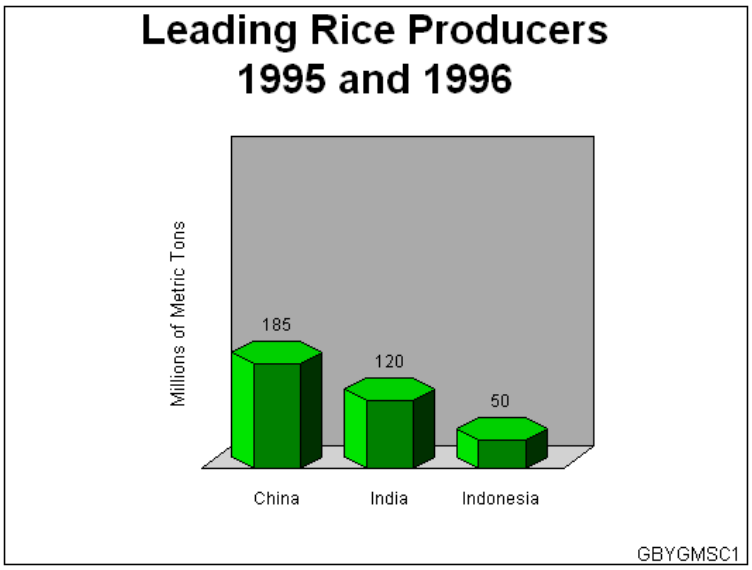
Generate the vertical bar charts using a BY statement. The BY statement produces a chart for each value of SITE. The FORMAT statement assigns the $COUNTRY. format to the chart variable. Assigning AXIS1 to the RAXIS= option causes all three charts to have the same response axis.

```
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;
```

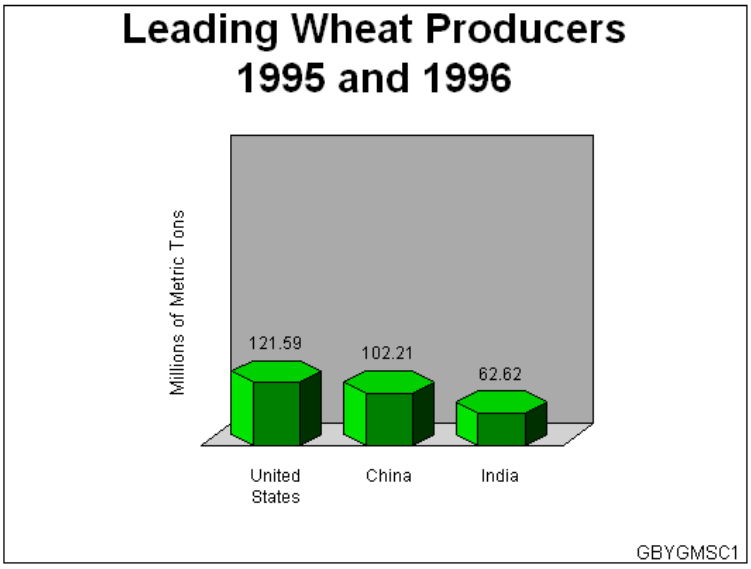
Display 14.1 Output for BY Value Corn



Display 14.2 Output for BY Value Rice



Display 14.3 Output for BY Value Wheat



Example 8. Creating a Simple Web Page with the ODS HTML Statement

Features:

ODS HTML statement options:

BODY=

CLOSE

GOPTIONS statement options:

RESET=

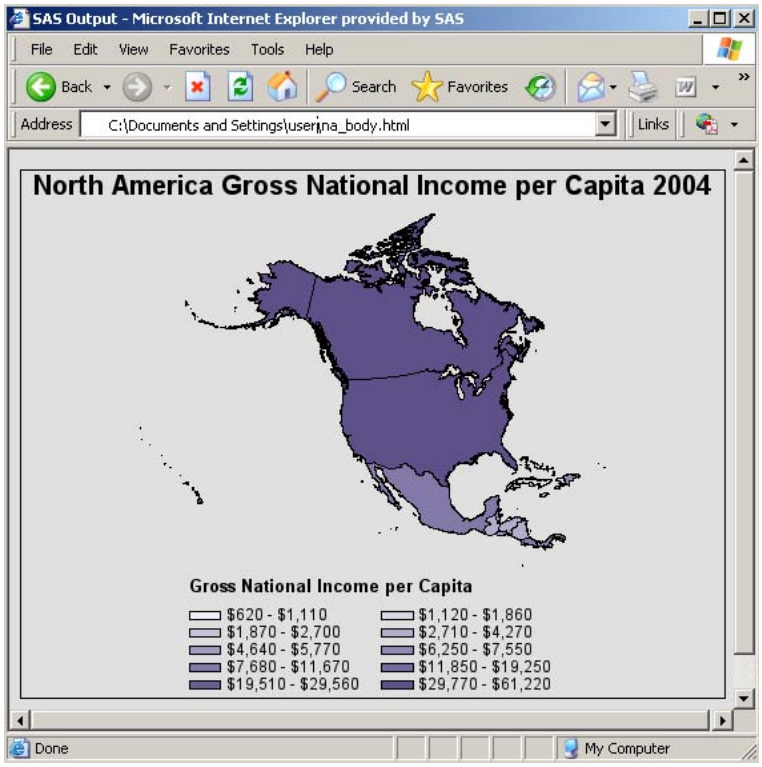
LEGEND statement options:

ACROSS=

LABEL=

Sample library member: GONCSWB1

Display 14.4 Displaying a Map in a Web Page



This example illustrates the simplest way to use the ODS HTML statement to create an HTML file and a GIF file that you can display in a Web browser. It generates one body file that displays one piece of SAS/GRAPH output—a map of average per capita income.

This example also illustrates default pattern behavior with maps and explicit placement of the legend on the graph. It shows how the default solid map pattern uses different shades of the default style color to differentiate between countries.

And it shows how to use a LEGEND statement to arrange and position a legend so it fits well with the graph's layout.

Close the ODS Listing destination for procedure output, and set the graphics environment. To conserve system resources, ODS LISTING CLOSE closes the Listing destination for procedure output. Thus, the graphics output is not displayed in the GRAPH window, although it is written to the graphics catalog and to the GIF files.

```
ods listing close;
goptions reset=all;
```

Open the ODS HTML destination. The BODY= option names the file for storing HTML output.

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

Define title for the map. By default, any defined title is included in the graphics output (GIF file).

```
title "North America Gross National Income per Capita 2004";
```

Define legend characteristics. The ACROSS= option defines the number of columns in the legend. The LABEL= option specifies a legend label and left-justifies it above the legend values.

```
legend across=2
       origin=(8,5)
       mode=share
       label=(position=top
              justify=left
              "Gross National Income per Capita")
;
```

Generate the prism map. Because the NAME= option is omitted, SAS/GRAPH assigns the default name GMAP to the GRSEG entry in the graphics catalog. This is the name that is assigned to the GIF file created by the ODS HTML statement.

```
proc gmap map=maps.namerica data=sashelp.demographics;
      id cont id;
    format gni dollar10.0;
      choro gni / levels=10 legend=legend1;
run;
quit;
```

Close the ODS HTML destination, and open the ODS Listing destination. You must close the HTML destination before you can view the output with a browser. ODS LISTING opens the Listing destination so that the destination is again available for displaying output during this SAS session.

```
ods html close;
ods listing;
```

Example 9. Combining Graphs and Reports in a Web Page

Features:

AXIS statement options:
- LENGTH=
- VALUE=

BY statement

GOPTIONS statement options:
- BORDER
- DEVICE=
- TRANSPARENCY

ODS HTML statement options:
- BODY=
- CONTENTS=
- FRAME=
- PATH=
- NOGTITLE

OPTIONS statement option:
- NOBYLINE

TITLE statement option:

#BYVAL

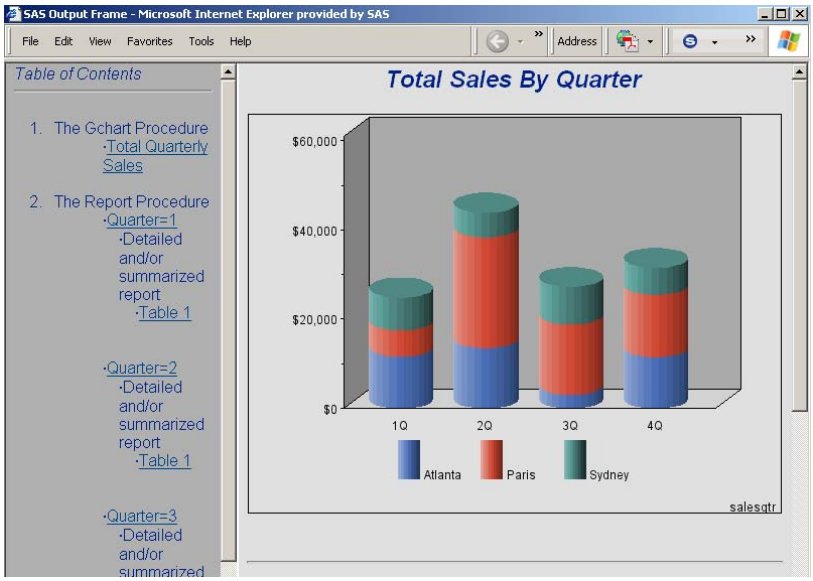
Sample library member: GONCGRW1

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.

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.

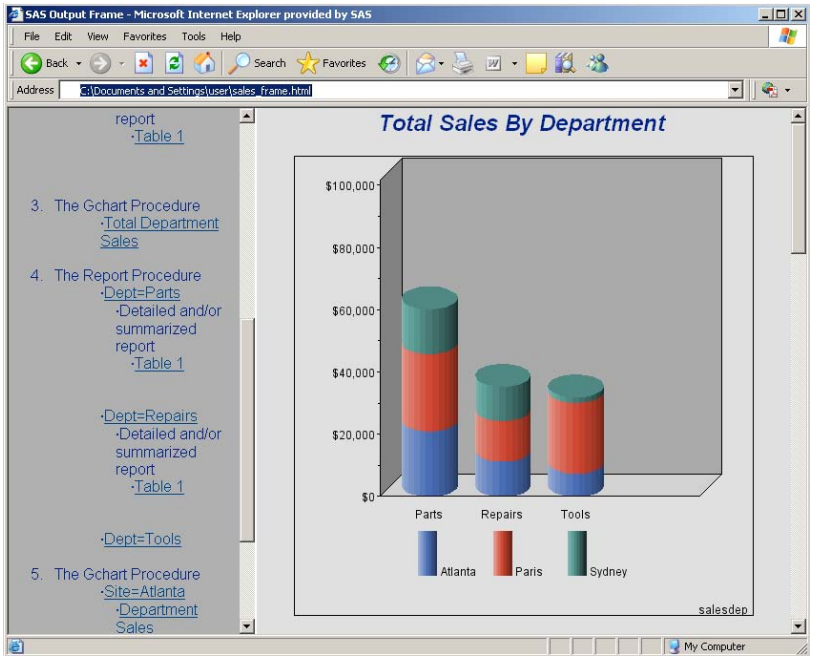
Display 14.5 Browser View of Bar Chart and Quarterly Sales Report



Notice that the chart title is displayed outside the graph as part of the HTML file.

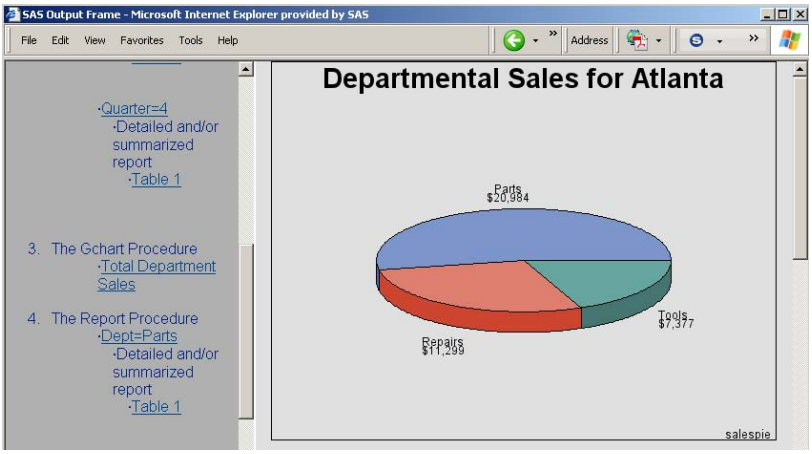
Select the link to *Total Department Sales* to display the second bar chart, as shown in the following figure.

Display 14.6 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.

Display 14.7 Browser View of Pie Charts of Site Sales



Because the pie charts are stored in one file, you can easily see all three by scrolling through the file.

Additional features include AXIS statements that specify the same length for both midpoint axes, so that the bar charts are the same width even though they have a different number of bars.

Close the ODS Listing destination for procedure output, and set the graphics environment. To conserve system resources, ODS LISTING CLOSE closes the Listing destination for procedure output. 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. The

BORDER option is used so that the border around the graphics output area is compatible with the borders that are created for nongraphics output.

```
ods listing close;

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
;
```

Open the ODS HTML destination. 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 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 Display 14.5 on page 316.

```
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;
```

Open 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 Display 14.5 on page 316.

```
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;
```

Open 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.

```
title "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, and open the ODS Listing destination.

```
ods html close;
ods listing;
```

Example 10. Creating a Bar Chart with Drill-Down Functionality for the Web

Features:

GOPTIONS statement option:

RESET=

TRANSPARENCY=

DEVICE=

ODS HTML statement options:

BODY=

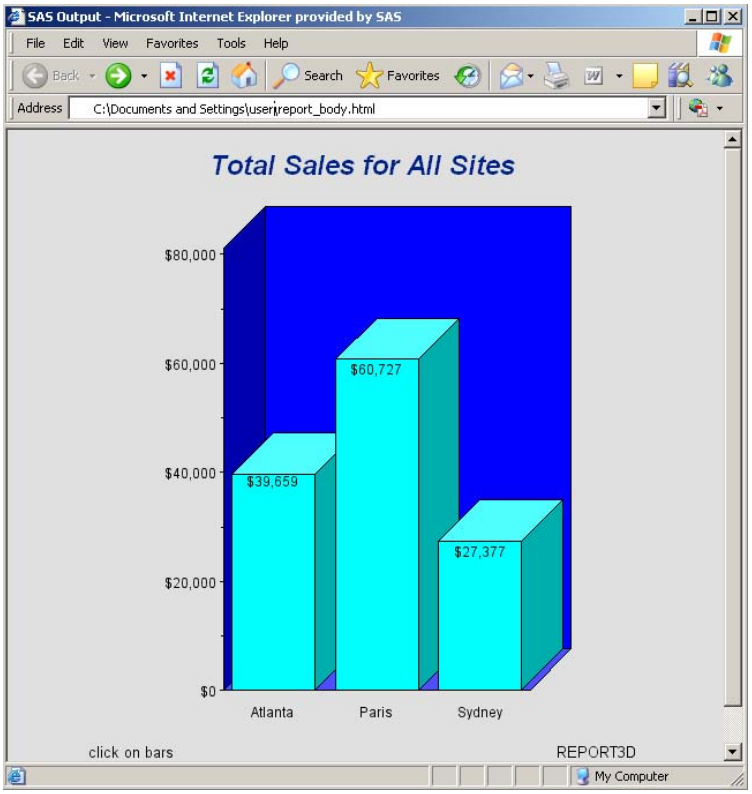
NOGTITLE

PATH=

Sample library member: GONDDCW1

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. The following figure shows the bar chart of sales.

Display 14.8 Vertical Bar Chart of Total Sales



Display 14.9 on page 323 shows the PROC REPORT output that appears when you click on the bar for Atlanta.

Display 14.9 PROC REPORT Output Displayed in a Web Browser

For additional information about this program, see “Details” on page 325.

Close the ODS Listing destination for procedure output, and set the graphics environment. To conserve system resources, ODS LISTING CLOSE closes the Listing destination for procedure output. 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.

```
ods listing close;
goptions reset=all device=gif transparency noborder;
```

Add the HTML variable to TOTALS 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#IDX1'";

else if site="Paris" then
   sitedrill="HREF='report_deptsales.html#IDX2'";

if site="Sydney" then
   sitedrill="HREF='report_deptsales.html#IDX3'";
run;
```

Open the ODS HTML destination. 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
    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 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=blue
                 maxis=axis2
                 raxis=axis1
                 name="report3d ";
run;
quit;
```

Open the file for the PROC REPORT output. Assigning a new body file closes REPORT_BODY.HTML.

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

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;
quit;
```

Clear the footnote.

```
goptions reset=footnote1;
```

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;
quit;
```

Close the ODS HTML destination, and open the ODS Listing destination.

```
ods html close;
ods listing;
```

Details

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. For example,

```
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, which is named **gqcke00k_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.

CHAPTER
15

Graphics Options and Device Parameters Dictionary

Introduction

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 220. For a list of all the device parameters, see "ADD Statement" on page 1129.

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

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 220 and Chapter 38, "The GDEVICE Procedure," on page 1125 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.

Dictionary of Graphics Options and Device Parameters

ACCESSIBLE

Generates descriptive text and summary statistics representing your graphics output.

Used in: GOPTIONS statement

Default: NOACCESSIBLE

Restriction: Only supported by JAVA and ActiveX when used with the ODS HTML output destination.

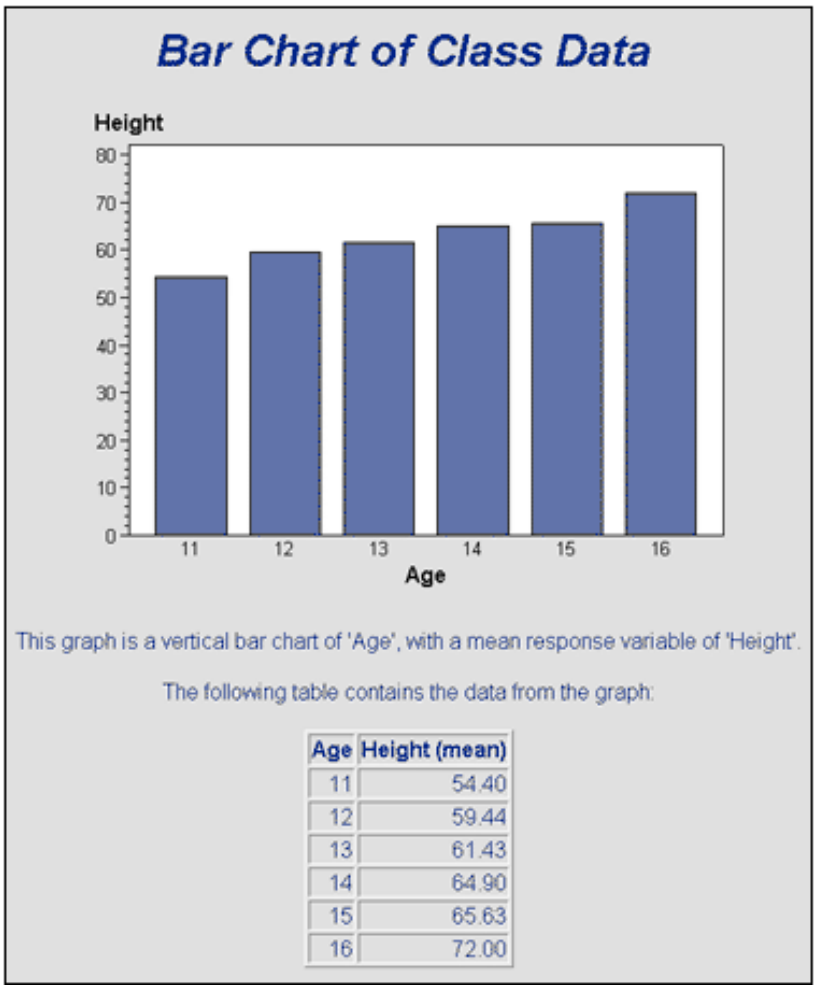
Syntax

ACCESSIBLE | NOACCESSIBLE

ACCESSIBLE
enables you to comply with section 508 of the Rehabilitation Acts and meet usability requirements for disabled users. Specifying the ACCESSIBLE option, when used with the ODS HTML statement, generates descriptive text and data for your graphs. 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.

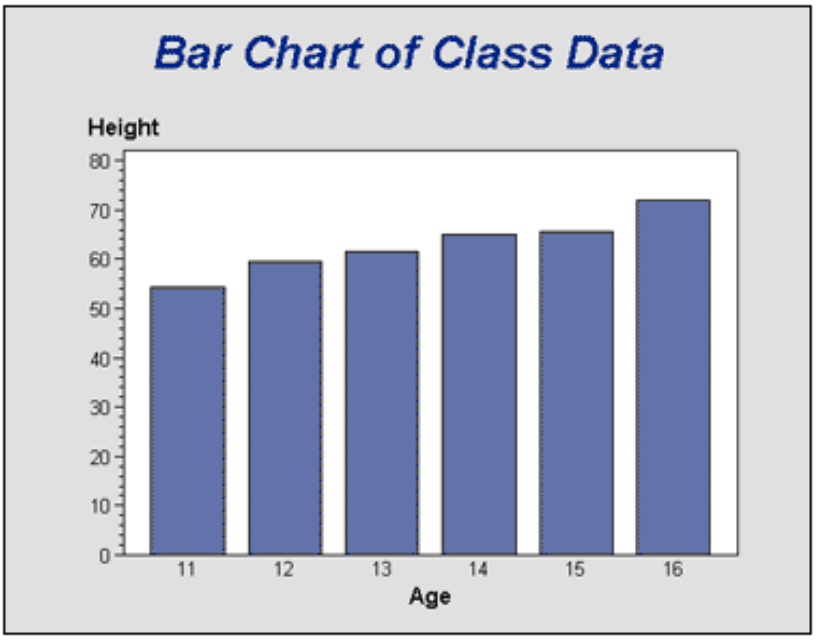
The information and the link are not visible in the output HTML, however both are detected by accessibility aids, such as screen readers. You can also access the information by pressing the tab key and enter. The information will be displayed once you press enter on the link in the footnote. The information will also display if you move your mouse over the location of the left–justified footnote, and click the link when the mouse pointer shape changes.

Figure 15.1 Accessible



NOACCESSIBLE
toggles off the ACCESSIBLE option.

Figure 15.2 Noaccessible



ADMGDF

Specifies whether to write an ADMGDF or GDF file when the GSFNAME= and GSFMODE= graphics options are used with a GDDM device driver.

Used in: GOPTIONS statement

Default: NOADMGDF

Restriction: GDDM device drivers on IBM mainframe systems only

Syntax

ADMGDF | NOADMGDF

ADMGDF
instructs the GDDM device driver to write out an ADMGDF file.

NOADMGDF
instructs the GDDM device driver to write out a GDF file.

ALTDESC

Specifies whether to write the DESCRIPTION= statement text to the ALT= text in an HTML file.

Used in: GOPTIONS statement

Default: ALTDESC

Restriction: Only supported when used with the HTML output destination and the DESCRIPTION= option.

Syntax

ALTDESC | NOALTDESC

ALTDESC
With ODS HTML output, 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

Sets the aspect ratio for graphics elements.

Used in: GOPTIONS statement GDEVICE procedure GDEVICE Detail window

Default: device–dependent

Restriction: not supported by Java or ActiveX

Syntax

ASPECT=*scaling-factor*

scaling-factor
is a non-negative 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

Specifies whether to generate hard copy automatically.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Defaults: GOPTIONS: NOAUTOCOPY; GDEVICE: AUTOCOPY=N
Restrictions: device-dependent; not supported by Java or ActiveX

Syntax

GOPTIONS: AUTOCOPY | NOAUTOCOPY

GDEVICE: AUTOCOPY=Y | N

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 that have printers attached directly to the device.

AUTOFEED

Specifies whether devices with continuous paper or automatic paper feed should roll or feed the paper automatically for the next graph.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Defaults: GOPTIONS: AUTOFEED (if a device is specified): GDEVICE: AUTOFEED=Y
Restrictions: device-dependent; not supported by Java or ActiveX
See also: PPDFILE

Syntax

GOPTIONS: AUTOFEED | NOAUTOFEED

GDEVICE: AUTOFEED=Y | N

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” on page 406.

AUTOSIZE

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 in: GOPTIONS statement

Default: device-dependent

Restriction: not supported by Java or ActiveX

See also: DEVOPTS

Syntax

AUTOSIZE=ON | OFF | DEFAULT

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” on page 350).

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

Specifies which edge of the document is the binding edge.

Used in: 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 also: DUPLEX, PPDFILE

Syntax

BINDING=DEFAULTEDGE | LONGEDGE | SHORTEDGE

Details

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 harware'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

Specifies whether to draw a border around the graphics output area.

Used in: GOPTIONS statement

Default: NOBORDER

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

Specifies the background color of the graphics output.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Gcolors window
Default: as specified in the Gcolors window

Syntax

CBACK=*background-color*

background-color
specifies any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 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 on graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: User's Guide*. △

If you explicitly specify a background color with the CBACK= option, the background color 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

Selects the color of the By lines that appear in the graphics output.

Used in: GOPTIONS statement
Default: (1) CTEXT= graphics option, if used; (2) first color in current color list
Restriction: not supported by Java or ActiveX

Syntax

CBY=*By line-color*

By line-color
specifies any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 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

Controls whether to use cell alignment.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Default: device-dependent
Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: CELL | NOCELL

GDEVICE: CELL=Y | N

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

Specifies whether the device—resident font is used when no font or FONT=NONE is specified in a SAS statement.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Defaults: GOPTIONS: CHARACTERS; GDEVICE: CHARACTERS=Y

Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: CHARACTERS | NOCHARACTERS

GDEVICE: CHARACTERS=Y | N

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 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

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 in: GDEVICE procedure

Default: device-dependent

Syntax

CHARREC=(*charrec-list(s)*)

charrec-list
a list of values that correspond to the fields in the Chartype window. *Charrec-list* has this form:

type, *rows*, *cols*, '*font*', 'Y' | 'N'

type	is the CHARTYPE number and can be an integer from 0 to 9999. (See "CHARTYPE" on page 338 for more information.)
rows	is the number of rows of text in the font that will fit on the display. (See "ROWS" on page 419 for more information.)
cols	is the number of columns of text in the font that will fit on the display. (See "COLS" on page 342 for more information.)
font	is a character string enclosed in quotation marks that contains the name of the corresponding device-resident font. (See "FONT NAME" on page 361 for more information.)
Y	represents a scalable font. A scalable font can be displayed at any size. (See "SCALABLE" on page 419 for more information.)
N	represents a nonscalable font. A nonscalable font can be displayed only at a fixed size. (See "SCALABLE" on page 419 for more information.)

For 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

Selects the number of the default hardware character set.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Default: device-dependent

Restriction: not supported by Java or ActiveX

Syntax

CHARTYPE=*hardware-font-chartype*

hardware-font-chartype
is a nonnegative integer from 0 to 999. *hardware-font-chartype* refers to the actual number for the device-resident font 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

Specifies whether SAS/GRAPH should use the device's hardware circle-drawing capability, if available.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Default: device-dependent

Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: CIRCLEARC | NOCIRCLEARC

GDEVICE: CIRCLEARC=Y | N

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.

hardware drawing is faster, but not all devices have the capability. SAS/GRAPH device drivers do not try to use the capability if the device does not have it.

NOCIRCLEARC
CIRCLEARC=N
causes SAS/GRAPH to use software move and draw commands to draw circles and arcs.

CMAP

Specifies a color map for the device.

Used in: GDEVICE procedure; GDEVICE Colormap window

Syntax

CMAP=('*from-color* : *to-color*' <...,'*from-color-n* : *to-color-n*'>)

from-color
specifies the name 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 12, "SAS/GRAPH Colors and Images," on page 167 for information on 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

Specifies whether to collate the output, if collation is supported by the device.

Used in: GOPTIONS statement; OPTIONS statement

Default: NOCOLLATE

Restriction: hardware–dependent, PostScript printers require a PPD file; not supported by Java or ActiveX

See also: GPROLOG, PPDFILE

Syntax

COLLATE | NOCOLLATE

Details

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 on the appropriate PJL command, consult the Printer Commands section of your printer's user manual.

COLORS

Specifies the foreground colors used to produce your graphics output if you do not specify colors explicitly in program statements.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Gcolors window

Default: device-dependent

Syntax

GOPTIONS: COLORS=<(*colors-list* | NONE)>

GDEVICE: COLORS=(<*colors-list*>)

colors-list
specifies one or more SAS color names. If you specify more than one color, separate each name with a blank. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 for information on 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 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) and 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. △

Featured in: "Example 3. Rotating Plot Symbols Through the Color List" on page 299

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 240 for more information on pattern rotation.

COLORTBL

An eight-character field in the Gcolors window that is not currently implemented. SAS/GRAPH ignores any value entered into this field.

COLORTYPE

Specifies the color space used by the user-written part of the Metagraphics device driver.

Used in: GDEVICE procedure; GDEVICE Metagraphics window

Default: NAME

Syntax

COLORTYPE=NAME | RGB | HLS | GRAY | CMY | CMYK | HSV | HSB

NAME	SAS predefined color names.
RGB	red-green-blue (RGB) color specifications.
HLS	hue-lightness-saturation (HLS) color specifications.
GRAY	gray-scale level.
CMY	cyan-magenta-yellow color specifications.
CMYK	cyan-magenta-yellow-black color specifications.
HSV | HSB	hue-saturation-value color specifications. These specifications are also referred to as hue-saturation-brightness (HSB).

See Chapter 12, “SAS/GRAPH Colors and Images,” on page 167 for a description of these color types.

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

Sets the number of columns that the device-resident font uses.

Used in: GDEVICE Chartype window; GDEVICE procedure; CHARREC= option

Default: 0

See also: CHARREC

Syntax

See "CHARREC" on page 337 for syntax.

Details

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

Selects the default color for PATTERN definitions when a color has not been specified.

Used in: GOPTIONS statement

Default: first color in current color list

Restriction: not supported by Java or ActiveX

Syntax

CPATTERN=*pattern-color*

pattern-color
specifies any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 for information about specifying colors.

Details

CPATTERN= is overridden by any color specification in a PATTERN statement. For details on how CPATTERN= affects the PATTERN statement, see "The Effect of the CPATTERN= Graphics Option" on page 250.

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 240 for more information on pattern rotation.

CSYMBOL

Specifies the default color for SYMBOL definitions when a color has not been specified.

Used in: GOPTIONS statement

Default: first color in current color list

Restriction: not supported by Java or ActiveX

Syntax

CSYMBOL=*symbol-color*

symbol-color
specifies any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 for information about specifying colors.

Details

CSYMBOL= is overridden by any color specification in a SYMBOL statement. See "SYMBOL Statement" on page 252.

CTEXT

Selects the default color for all text and the border.

Used in: GOPTIONS statement

Default: black for Java and ActiveX devices; for other devices, the first color in current color list

See also: CTITLE

Restriction: partially supported by Java

Syntax

CTEXT=*text-color*

text-color
specifies any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 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 195 for more information. △

CTITLE

Selects the default color for all titles, footnotes, and notes, and the border.

Used in: GOPTIONS statement

Default: (1) color specified by CTEXT=, if used; (2) black for Java and ActiveX devices; for other devices, the first color in current color list

See also: CTEXT

Syntax

CTITLE=*title-color*

title-color
specifies any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 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 195 for more information. △

DASH

Specifies whether to use the device's hardware dashed-line capability, if available.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Default: device– dependent

Restriction: not supported by Java or ActiveX

See also: DASHLINE

Syntax

GOPTIONS: DASH | NODASH

GDEVICE: DASH=Y | N

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.

hardware drawing is faster, but not all devices have the capability. SAS/GRAPH device drivers do not try to use the capability if the device does not have it.

NODASH
DASH=N
causes SAS/GRAPH to draw the dashed lines.

DASHLINE

Specifies which dashed lines should be generated by hardware means if possible.

Used in: GDEVICE procedure; GDEVICE Parameters window

Default: device-dependent

See also: DASH

Syntax

DASHLINE='*dashed-line-hex-string*'X

dashed-line-hex-string
is a hexadecimal string 16 characters long that must be completely filled. Each bit in the string corresponds to a line type. See Figure 14.22 on page 277 for line types 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

```
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

Scales the lengths of the dashes in a dashed line.

Used in: GOPTIONS statement

Default: DASHSCALE=1

Restriction: not supported by Java or ActiveX

Syntax

DASHSCALE=*scaling-factor*

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

Controls the amount of time between graphs in the animation sequence.

Used in: GOPTIONS statement

Default: 0

Restriction: GIFANIM driver only; not supported by all browsers

Syntax

DELAY=*delay-time*

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

Provides a description of the device entry.

Alias: DES

Used in: GDEVICE procedure GDEVICE Detail window

Default: none

Syntax

DESCRIPTION='*text-string*'

text-string
is a string up to 256 characters long. This is a comment field and does not affect the graphics output.

DEVADDR

Specifies the location of the device to which the output of device drivers is sent.

Used in: GOPTIONS statement

Default: host dependent

Restriction: IBM mainframe systems only

Syntax

DEVADDR=*device-address*

DEVICE

Specifies the device driver to which SAS/GRAPH sends the procedure output. The device driver controls the format of graphics output.

Alias: DEV

Used in: GOPTIONS statement OPTIONS statement

Default: device–dependent

Syntax

DEVICE=*device-entry*

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 on selecting a device entry and changing device parameters, see Chapter 6, "Using Graphics Devices," on page 67.

For information on using device drivers to display and print graphics output, see Chapter 7, "SAS/GRAPH Output," on page 87.

For information on using device drivers to export graphics output to external files, see "Specifying the Graphics Output File Type for Your Graph" on page 91. For information on using device drivers to create output for the Web, see "Generating Web Presentations" on page 451.

DEVMAP

Specifies the device map to be used when device-resident fonts are used.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host File Options window

Default: device-dependent

Restriction: not supported by Java or ActiveX

Syntax

DEVMAP=*device-map-name* | NONE

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

Specifies the hardware capabilities of the device.

Used in: GDEVICE procedure; GDEVICE Parameters window
Default: device-dependent

Syntax

DEVOPTS='*hardware-capabilities-hex-string*'X

hardware-capabilities-hex-string
is a hexadecimal string 16 characters long that must be completely filled. The following table lists the hardware capabilities of each bit:

Table 15.1 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
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

Bit On	Capability
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	pick 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
43	device can scale images
44	text clipping supported
45	static color device
46	driver does prolog processing
47	driver does epilog processing
48	driver output only uses 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 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 an SAS-supplied driver 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

Specifies the information required by SAS/GRAPH routines to determine the nature of the output device.

Used in: GDEVICE procedure; GDEVICE Host File Options window

Default: device-dependent

Syntax

DEVTYPE=*device-type*

device-type
is a string eight characters long containing either blanks or some token name that is interpreted by the host. *Device-type* can be:

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 on 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

Specifies whether output is displayed on the graphics device but does not affect whether a graph is placed in a catalog.

Used in: GOPTIONS statement

Default: DISPLAY

Restriction: not supported by Java or ActiveX

Syntax

DISPLAY | NODISPLAY

Details

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

Specifies what happens to the graphic after it is displayed.

Used in: GOPTIONS statement

Default: NONE

Restriction: GIFANIM driver only

Syntax

DISPOSAL=NONE | BACKGROUND | PREVIOUS | UNSPECIFIED

NONE
causes the graphic to be left in place after displaying. This is the default.

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.

Details

In Version 6, the ERASE | NOERASE graphics option performed this function for the GIFANIM driver.

DRVINIT

Specifies host commands to be executed before driver initialization.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host Commands window

Restriction: not supported by Java or ActiveX

Syntax

DRVINIT1='*system-command(s)*'

DRVINIT2='*system-command(s)*'

system-command(s)
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. 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

Specifies whether the device can be queried for information about the current device configuration.

Used in: GDEVICE procedure GDEVICE Detail window

Default: device-dependent

Syntax

DRVQRY | NODRVQRY

Details

Generally, this setting is device-dependent and you should not change it.

DRVTERM

Specifies host commands to be executed after the driver terminates.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host Commands window

Restriction: not supported by Java or ActiveX

Syntax

DRVTERM1='*system-command(s)*'

DRVTERM2='*system-command(s)*'

system-command(s)
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. 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

Specifies whether to use duplex printing if available on the device.

Used in: GOPTIONS statement; OPTIONS statement

Default: NODUPLEX

Restriction: duplex printers only

See also: BINDING, GSFMODE, PPDFILE

Syntax

DUPLEX | NODUPLEX

Details

When DUPLEX is on, the driver sets up the printer for duplex operation. Before producing the first graph, set GSFMODE=REPLACE on 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 on 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

Specifies whether to erase graph after display.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Defaults: GOPTIONS: NOERASE; GDEVICE: ERASE=N

Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: ERASE | NOERASE

GDEVICE: ERASE=Y | N

ERASE
ERASE=Y
causes the graph to be erased when you press RETURN after the graph has been displayed.

NOERASE
ERASE=N
causes the graph to remain on the display when you press RETURN 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

Specifies the file extension for an external graphics file.

Used in: GOPTIONS statement
Default: device-dependent
Restriction: not supported by Java or ActiveX
See also: GACCESS, GSFNAME

Syntax

EXTENSION='*file-type*'

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 filename
- if EXTENSION=' 'or EXTENSION= is not used, the driver's default extension is added to the filename
- if the driver has no default extension, SAS/GRAPH uses the default extension .GSF.

FASTTEXT

Specifies whether to use integer-based font processing for faster font rendering.

Used in: GOPTIONS statement
Default: FASTTEXT
Restriction: not supported by Java or ActiveX

Syntax

FASTTEXT | NOFASTTEXT

FBY

Selects the font for By lines.

Used in: GOPTIONS statement
Default: (1) font specified by FTEXT=, if used; (2) device–resident font (3) simulate font
Restriction: not supported by Java or ActiveX
See also: "BY Statement" on page 216

Syntax

FBY=*By line-font*

By line-font
specifies the font for all By lines on the graphics output. See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 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

Specifies the number of system fonts to keep open at one time.

Used in: GOPTIONS statement
Default: FCACHE=3
Restriction: not supported by Java or ActiveX

Syntax

FCACHE=*number-fonts-open*

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

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 in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host File Options window

Default: DRIVERTERM (if a device is specified)

Restriction: not supported by Java or ActiveX

See also: "Specifying the Graphics Output File Type for Your Graph" on page 91

Syntax

FILECLOSE=DRIVERTERM | GRAPHEND

DRIVERTERM
DRIVER
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.

GRAPHEND
GRAPH
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.

FILEONLY

Specifies whether a file or a storage location is the default destination for graphics output.

Used in: GOPTIONS statement

Default: device-dependent

Restriction: FILEONLY ignored if the device requires the output destination to be a storage location; not supported by Java or ActiveX

See also: DEVOPTS, GSFNAME

Syntax

FILEONLY | NOFILEONLY

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 91.

FILL

Specifies whether to use the device's hardware rectangle-fill capability.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Restriction: not supported by Java or ActiveX

Default: device–dependent

Syntax

GOPTIONS: FILL | NOFILL

GDEVICE: FILL=Y | N

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.

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

Specifies the number of pixels to move before drawing the next line in a software fill of a solid area.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Default: device-dependent
Restriction: not supported by Java or ActiveX
See also: FILL, PIEFILL, POLYGONFILL

Syntax

FILLINC= 0...9999

Details

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

Specifies the device–resident font associated with CHARTYPE.

Used in: GDEVICE Chartype window; GDEVICE procedure; CHARREC= option
Required if adding or modifying a CHARREC
See also: CHARREC

Syntax

See “CHARREC” on page 337 for syntax.

Details

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 an SAS-supplied device entry, this parameter already is set for most available device-resident fonts.

FONTRES

Controls the resolution of Bitstream fonts.

Used in: GOPTIONS statement
Default: NORMAL
Restriction: not supported by Java or ActiveX
See also: FASTTEXT, FCACHE, RENDER, RENDERLIB, SWFONTRENDER

Syntax

FONTRES=NORMAL | PRESENTATION

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=saswork
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=saswork
nofasttext
fcache=3
```

FORMAT

Sets the file format of the metacode file produced by the SAS-supplied part of the Metagraphics device driver.

Used in: GDEVICE procedure; GDEVICE Metagraphics window
Default: CHARACTER
Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

FORMAT=CHARACTER | BINARY

Details

A blank field defaults to CHARACTER. For information about Metagraphics drivers, contact Technical Support.

FTEXT

Sets the default font for all text.

Used in: GOPTIONS statement

Default: Default device–resident font (except the first title)

Restriction: partially supported by Java or ActiveX

See also: FTITLE

Syntax

FTEXT=*text-font*

text-font
specifies the font for all text on the graphics output. See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 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 Java 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 195 for more information. △

FTITLE

Selects the default font for the first TITLE line.

Used in: GOPTIONS statement

Default: (1) font specified by FTEXT=, if used; (2) value of the style variable (3)device-resident font (4)simulate font

See also: FTEXT

Syntax

FTITLE=*title-font*

title-font
specifies the font for the TITLE1 statement. See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 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 195 for more information. △

FTRACK

Controls the amount of space between letters in the SAS-supplied Bitstream fonts (Brush, Century, Swiss, and Zapf).

Used in: GOPTIONS statement

Default: TIGHT

Restriction: not supported by Java or ActiveX

Syntax

FTRACK=LOOSE | NONE | NORMAL | TIGHT | TOUCH | V5

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 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

Specifies the format or the destination or both of graphics data written to a device or graphics stream file (GSF).

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host File Options window

Default: device-dependent

Restriction: not supported by Java, ActiveX, or shortcut devices. See Chapter 6, "Using Graphics Devices," on page 67 for more information about devices.

Syntax

GACCESS=*output-format* | '*output-format destination*'

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 36 for more information on 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 on 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 91 for more information on 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. △

GCLASS

Specifies the output class for IBM printers

Used in: GOPTIONS statement

Default: GCLASS=G

Restriction: used only with IBM3287 and IBM3268 device drivers on z/OS systems only

Syntax

GCLASS=*SYSOUT-class*

Details

Specifies the SYSOUT class to which the IBM3287 and IBM3268 device driver output is written.

GCOPIES

Sets the current and maximum number of copies to print.

Used in: GOPTIONS statements; GDEVICE Parameters window; GDEVICE procedure; OPTIONS statement

Defaults: GOPTIONS: GCOPIES=(0,20) GDEVICE: GCOPIES=0

Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: GCOPIES=(<*current-copies*><,*max-copies*>)

GDEVICE: GCOPIES=*current-copies*

current-copies
is a nonnegative integer ranging from 0 through 255, but it cannot exceed the *max-copies* value specified. A value of 0 or 1 produces a single copy.

max-copies
is a nonnegative integer ranging from 1 through 255.

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.

Details

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.

GDDMCOPY

Instructs the driver to issue either an FSCOPY or GSCOPY call to GDDM when AUTOCOPY is in effect.

Used in: GOPTIONS statement
Default: FSCOPY
Restriction: GDDM device drivers on IBM mainframe systems only
See also: AUTOCOPY

Syntax

GDDMCOPY=FSCOPY | GSCOPY

FSCOPY
used when sending output to an IEEE attached plotter.

GSCOPY
used when creating an ADMPRINT file for output on 3287-type printers.

GDDMNICKNAME

Selects a GDDM nickname for the device to which output is sent.

Alias: GDDMN
Used in: GOPTIONS statement
Restriction: GDDM device drivers on IBM mainframe systems only

Syntax

GDDMNICKNAME=*nickname*

Details

Refer to the SAS Help facility for details on using GDDM drivers and options.

GDDMTOKEN

Selects a GDDM token for the device to which output is sent.

Alias: GDDMT
Used in: GOPTIONS statement
Restriction: GDDM device drivers on IBM mainframe systems only

Syntax

GDDMTOKEN=*token*

Details

Refer to the SAS Help facility for details on using GDDM drivers and options.

GDEST

Specifies the JES SYSOUT destination for IBM printers.

Used in: GOPTIONS statement
Default: LOCAL
Restriction: used only with IBM3287 and IBM3268 device drivers on z/OS systems

Syntax

GDEST=*destination*

GEND

Appends an ASCII string to every graphics data record that is sent to a device or file.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Gend window
Restriction: not supported by Java or ActiveX
See also: GSTART

Syntax

GEND='*string*' <...'*string-n*'>

'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 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 the way 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

Sends a string to a device or file after all graphics commands are sent.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Gepilog window
Restriction: not supported by Java or ActiveX
See also: PREGEPILOG, POSTGEPILOG

Syntax

GEPILOG='*string*' <...'*string-n*'>

'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 the way 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. △

GFORMS

Specifies the JES form name for IBM printers.

Used in: GOPTIONS statement

Default: STD

Restriction: used only with IBM3287 and IBM3268 device drivers on z/OS systems only

Syntax

GFORMS=*'forms-code'*

GOUTMODE

Appends to or replaces the graphics output catalog.

Used in: GOPTIONS statement

Default: APPEND

Restriction: not supported by Java or ActiveX

Syntax

GOUTMODE=APPEND | REPLACE

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

Sends a string to device or file before graphics commands are sent.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Gprolog window

Restriction: not supported by Java or ActiveX

See also: PREGPROLOG, POSTGPROLOG

Syntax

GPROLOG='*string*' <...'*string-n*'>

'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 the way 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

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 in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host File Options window

Restriction: not supported by Java or ActiveX

Default: host dependent

Syntax

GPROTOCOL=*module-name*

***module-name* can be one of these**

SASGPADE*

SASGPAGL*

SASGPASC

SASGPAXI*

SASGPCAB*

SASGPCHK*

SASGPDAT*

SASGPDCA*

SASGPHEX

SASGPHYD*

SASGPIDA*

SASGPIDX*

SASGPIMP*

SASGPIOC*

SASGPISI*

SASGPI24*

SASGPLCL*

SASGPNET*

SASGPMIC*

SASGPRTM*

SASGPSCS*

SASGPSTD

SASGPSTE*

SASGPTCX*

SASGPVAT*

SASGP497*

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

Specifies whether to return a step code at graphics procedure termination.

Used in: GOPTIONS statement

Restriction: not supported by Java or ActiveX

Default: GRAPHRC

Syntax

GRAPHRC | NOGRAPHRC

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

Controls the length of records written to the graphics stream file (GSF).

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host File Options window

Default: device-dependent

Restriction: not supported by Java or ActiveX

See also: PROMPTCHARS

Syntax

GSFLEN=*record-length*

record-length
must be a nonnegative integer up to five digits long (0...99999). GSFLEN= specifies the length of the records written by the driver to a GSF or to the device.

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

Specifies the disposition of records written to a graphics stream file (GSF) or to a device or communications port by the device driver.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host File Options window

Default: REPLACE

Restriction: not supported by Java or ActiveX

See also: GACCESS, GSFNAME

Syntax

GSFMODE=APPEND | PORT | REPLACE

APPEND
adds the records to the end of a GSF designated by the GACCESS= or GSFNAME= graphics option or device parameter. 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 on how SAS/GRAPH names catalog entries, see "Specifying the Graphics Output File Type for Your Graph" on page 91.

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 on how SAS/GRAPH names catalog entries, see "Specifying the Graphics Output File Type for Your Graph" on page 91.

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 91 for more information on creating a graphics stream file.

GSFNAME

Specifies the fileref of the file or aggregate file location to which graphics stream file records are written.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host File Options window

Restriction: Not valid for IBM32*xx*, linkable, Metagraphics, Java, or ActiveX drivers.

See also: GACCESS, GSFMODE

Syntax

GSFNAME=*fileref*

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 36 for additional information on 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” on page 365.

See “Specifying the Graphics Output File Type for Your Graph” on page 91 for more information on creating graphics stream files.

GSFPROMPT

Specifies whether to write prompt messages to the graphics stream file (GSF).

Used in: GOPTIONS statement

Default: NOGSFPROMPT

Restriction: not supported by Java or ActiveX

Syntax

GSFPROMPT | NOGSFPROMPT

Details

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

Sets the number of lines of display used for graphics for devices whose displays can be divided into graphics and text areas.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Restriction: not supported by Java or ActiveX

Default: device-dependent

Syntax

GSIZE=*lines*

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

Prefixes every record of graphics data sent to a device or file with a string of characters.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Gstart window

Default: none

Restriction: not supported by Java or ActiveX

See also: GEND

Syntax

GSTART='*string* <...'*string-n*'>

'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 the way 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

Specifies the default unit of measure to use with height specifications.

Used in: GOPTIONS statement

Default: CELLS

Restriction: partially supported by Java or ActiveX

Syntax

GUNIT=*units*

units must be one of

CELLS	character cells
CM	centimeters
IN	inches
PCT	percentage of the graphics output area

PT points (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 then GUNIT is ignored and inches will be used.

GWAIT

Specifies the time between each graph displayed in a series.

Used in: GOPTIONS statement

Default: GWAIT=0

Restriction: not supported by Java or ActiveX

Syntax

GWAIT=*seconds*

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 RETURN 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 (or the RETURN or END key, depending on your device) 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.

GWRITER

Specifies the name of the external writer used with IBM printers.

Used in: GOPTIONS statement

Default: SASWTR

Restriction: Used only with IBM3287 and IBM3268 device drivers on z/OS systems

Syntax

GWRITER='*writer-name*'

HANDSHAKE

Specifies the type of flow control used to regulate the flow of data to a hard copy device.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Default: host dependent
Restriction: not supported by Java or ActiveX

Syntax

HANDSHAKE=HARDWARE | NONE | SOFTWARE | XONXOFF

HARDWARE
HARD
specifies that SAS/GRAPH instruct the device to use the hardware CTS and RTS signals. (This 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 you are using provides its own flow control.

SOFTWARE
SOFT
specifies that SAS/GRAPH use programmed flow control with plotters in eavesdrop mode.

XONXOFF
X
specifies that SAS/GRAPH instruct the device to use ASCII characters DC1 and DC3. (This 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

Specifies the height of By lines generated when you use BY-group processing.

Used in: GOPTIONS statement

Default: One cell unless the HTEXT= option is used

Restriction: not supported by Java or ActiveX

See also: "BY Statement" on page 216

Syntax

HBY=*By line-height <units>*

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 328.

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

Specifies the command that executes a user-supplied program to create HEADER records for the driver.

Used in: GDEVICE procedure; GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

See also: HEADERFILE

Syntax

HEADER='*command*'

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

Specifies the fileref for the file from which the Metagraphics driver reads HEADER records.

Used in: GDEVICE procedure; GDEVICE Metagraphics window
Restriction: Used only with user-supplied Metagraphics drivers.
See also: HEADER

Syntax

HEADERFILE=*fileref*

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 36 for details.

Details

For information about Metagraphics drivers, contact Technical Support.

HORIGIN

Sets the horizontal offset from the lower-left corner of the display area to the lower-left corner of the graph.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window
Restriction: not supported by Java or ActiveX
See also: VORIGIN

Syntax

HORIGIN=*horizontal-offset* <IN | CM | PT>

horizontal-offset **<IN | CM | PT>**
must be a nonnegative number and can be followed by a unit specification, either IN for inches (default), or 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 59 for details.

HOSTSPEC

Stores FILENAME statement options in the device entry.

Used in: GDEVICE procedure; GDEVICE Host File Options window

Syntax

HOSTSPEC='*text-string*'

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

Specifies the number of columns in the graphics output area.

Used in: GOPTIONS statement

Default: device-dependent: the value of the LCOLS or PCOLS device parameter

Restriction: not supported by Java or ActiveX

See also: PCOLS, LCOLS, VPOS

Syntax

HPOS=*columns*

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 59 for more information.

HSIZE

Sets the horizontal size of the graphics output area.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Restriction: partially supported by Java or ActiveX

See also: VSIZE, XMAX

Syntax

HSIZE=*horizontal-size* <IN | CM | PT>

***horizontal-size* <IN | CM | PT>**
specifies the width of the graphics output area; *horizontal-size* must be a positive number and can be followed by a unit specification, either IN for inches (default), or 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

Specifies the default height of the text in the graphics output.

Used in: GOPTIONS statement

Default: One cell

Restriction: partially supported by Java

Syntax

HTEXT=*text-height <units>*

text-height <units>
specifies the height of the text; by default *text-height* is 1. For a description of *units*, see "Specifying Units of Measurement" on page 328.

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 195 for more information. △

HTITLE

Selects the default height used for the first TITLE line.

Used in: GOPTIONS statement

Default: Two cells unless HTEXT= is used

Syntax

HTITLE=*title-height <units>*

title-height <units>
specifies the height of the text in the TITLE1 statement. By default, *title-height* is 2. For a description of *units*, see "Specifying Units of Measurement" on page 328.

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 195 for more information. △

IBACK

Specifies an image file to display in a graph's background area.

Restriction: partially supported by Java
See also: CBACK, IMAGESTYLE

Syntax

IBACK=*fileref* | '*external-file*' | '*URL*'| " "

fileref
specifies a fileref that points to the image file 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 filename of the image file 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.

Details

The image can be used with any procedures that produce a picture or support the CBACK= option. The IBACK option is supported by the Graph applet and the Map applet, but it is not supported by the Contour applet. See Chapter 16, "Introducing SAS/GRAPH Output for the Web," on page 439 for information about these applets.

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 on graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: User's Guide*.

For a list of the file types that you use, see "Image File Types Supported by SAS/GRAPH" on page 181.

ID

Specifies the description string used by the Metagraphics driver.

Used in: GDEVICE procedure; GDEVICE Metagraphics window

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

ID='*description*'

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

Enables or disables image output

Used in: GOPTIONS statement

Default: IMAGEPRINT

Restriction: not supported by Java or ActiveX

Syntax

IMAGEPRINT | NOIMAGEPRINT

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

Specifies the way to display the image file that is specified on the IBACK= option.

Default: TILE

Restriction: not supported by Java

Syntax

IMAGESTYLE= TILE | FIT

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 on graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: User's Guide*. △

INTERACTIVE

Sets level of interactivity for Metagraphics driver.

Used in: GDEVICE procedure; GDEVICE Metagraphics window

Default: USER

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

INTERACTIVE=USER | GRAPH | PROC

USER
specifies that the user-written part of the driver be executed outside of SAS/GRAPH.

PROC
specifies that the user-written part of the Metagraphics driver be invoked after the procedure is complete.

GRAPH
specifies that the user-written part be invoked for each graph.

Details

For information about Metagraphics drivers, contact Technical Support.

INTERLACED

Specifies whether images are to be displayed as they are received in the browser.

Used in: GOPTIONS statement
Default: NONINTERLACED
Restriction: driver-dependent, GIF series of drivers only

Syntax

INTERLACED | NONINTERLACED

Details

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 allow you to set an option that will determine how images are displayed.

INTERPOL

Sets the default interpolation value for the SYMBOL statement.

Used in: GOPTIONS statement
Restriction: not supported by Java or ActiveX

Syntax

INTERPOL=*interpolation-method*

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 252 for the complete syntax of all interpolation methods.

ITERATION

Specifies the number of times to repeat the animation loop.

Used in: GOPTIONS statement

Default: 0

Restriction: GIFANIM driver only

Syntax

ITERATION=*iteration-count*

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. *Iteration-count* can be a number from 0...65535. A value of 0 causes the animation to loop continuously.

Details

In Version 6, the GCOPIES graphics option controlled iteration for the GIFANIM driver.

KEYMAP

Selects the keymap to use.

Used in: GOPTIONS statement

Default: installation dependent

Restriction: not supported by Java or ActiveX

Syntax

KEYMAP=*key-map-name* | NONE

key-map-name
specifies the name of a keymap.

NONE
suppresses the keymap assigned by default to a non-U.S. keyboard. If you specify KEYMAP=NONE, text might display incorrectly or not at all.

Details

Non-default key maps usually are used only with non-U.S. Keyboards.

LCOLS

Sets the number of columns in the graphics output area for landscape orientation.

Used in: GDEVICE procedure; GDEVICE Detail window
Default: device-dependent
See also: HPOS, LROWS, PCOLS

Syntax

LCOLS=*landscape-columns*

landscape-columns
must be a nonnegative integer up to three digits long (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 59 for more information.

LFACTOR

Selects the default hardware line thickness.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Default: device-dependent
Restriction: Used only with devices that can draw hardware lines of varying thicknesses. Not supported by Java or ActiveX.

Syntax

LFACTOR=*line-thickness-factor*

line-thickness-factor
can range from 0 through 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

Sets the number of rows in the graphics output area for landscape orientation.

Used in: GDEVICE procedure; GDEVICE Detail window

Default: device-dependent

See also: LCOLS, PROWS, VPOS

Syntax

LROWS=*landscape-rows*

landscape-rows
is a nonnegative integer up to three digits long (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 59 for more information.

MAXCOLORS

Sets the total number of colors that can be displayed at once.

Used in: GDEVICE procedure; GDEVICE Parameters window

Default: device-dependent

See also: PENMOUNTS

Syntax

MAXCOLORS=*number-of-colors*

number-of-colors
must be an integer in the range 2 through 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

Sets the maximum number of vertices for hardware-drawn polygons.

Used in: GDEVICE procedure; GDEVICE Parameters window

Default: device-dependent

Syntax

MAXPOLY=*number-of-vertices*

number-of-vertices
is a nonnegative integer up to four digits long. 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

Specifies the model number of the output device.

Used in: GDEVICE procedure; GDEVICE Detail window
Default: device-dependent

Syntax

MODEL=*model-number*

model-number
is a nonnegative integer up to five digits long that is the SAS-designated model number for the corresponding device. It is not the same as a manufacturer's model number.

Details

Do not change this field in SAS-supplied drivers or in drivers that you copy from SAS-supplied drivers.

MODULE

Specifies the name of the corresponding executable driver module for the device.

Used in: GDEVICE procedure; GDEVICE Detail window
Default: device-dependent

Syntax

MODULE=*driver-module*

driver-module
is a literal string up to eight characters long. All standard driver modules begin with the characters SASGD.

Details

Do not change this field in SAS-supplied drivers or in drivers that you copy from SAS-supplied drivers.

NAK

Specifies the negative response for software handshaking for Metagraphics drivers.

Used in: GDEVICE procedure; GDEVICE Metagraphics window
Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

NAK='*negative-handshake-response*'X

negative-handshake-response
is a hexadecimal string up to 16 characters long.

Details

For information about Metagraphics drivers, contact Technical Support.

OFFSHADOW

Controls the width and depth of the drop shadow in legend frames.

Used in: GOPTIONS statement
Default: (0.0625, – 0.0625) IN
Restriction: not supported by Java or ActiveX

Syntax

OFFSHADOW=(*x* <*units*>, *y* <*units*>) | (*x,y*) <*units*>

x,y
specify the width (*x*) and depth (*y*) of the drop shadow generated by the LEGEND statement.

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 328.

Details

The values specified by OFFSHADOW= are used with the CSHADOW= and CBLOCK= options in a LEGEND statement. For details, see "LEGEND Statement" on page 225.

PAPERDEST

Specifies which output bin the printer should use if multiple bins are available on the device.

Used in: GOPTIONS statement; OPTIONS statement

Default: 1 (the upper output bin)

Restrictions: hardware-dependent, PostScript printers require a PPD file; not supported by Java or ActiveX

See also: PAPERSOURCE, PPDFILE

Syntax

PAPERDEST=*bin*

bin
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	the name or number of the output bin – for example, PAPERDEST=4, PAPERDEST=BIN2, PAPERDEST=SIDE
'*long bin name*'	a character string that is the name of the output bin – for example, PAPERDEST="Top Output Bin'. Names with blanks or special characters must be quoted.

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

Specifies the increment of paper that is ejected when a graph is completed.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Default: PAPERFEED=0.0 IN

Restriction: device-dependent; not supported by Java or ActiveX

Syntax

PAPERFEED=*feed-increment* <IN | CM>

***feed-increment* <IN | CM>**
must be a nonnegative number and can be followed by a unit specification, either 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

Sets the width of the paper used with plotters.

Used in: GOPTIONS statement

Default: maximum dimensions specified in the device driver

Restriction: ZETA plotters and KMW rasterizers

Syntax

PAPERLIMIT=*width* <IN | CM>

***width* <IN | CM>**
specifies the paper width in 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

Specifies the name of a paper size.

Used in: GOPTIONS statement; OPTIONS statement

Default: device-dependent

Restriction: hardware- dependent, PostScript printers require a PPD file; not supported by Java or ActiveX

See also: PAPERSOURCE, PPDFILE

Syntax

PAPERSIZE='*size-name*'

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 on 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

Specifies which paper tray the printer should use if multiple trays are available on the device.

Used in: GOPTIONS statement; OPTIONS statement

Default: device-dependent

Restriction: hardware– dependent, PostScript printers require a PPD file; not supported by Java or ActiveX

See also: PAPERDEST, PAPERSIZE, PPDFILE

Syntax

PAPERSOURCE=*tray*

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	the name or number of the paper tray, for example, PAPERSOURCE=3, PAPERSOURCE=TRAY3, PAPERSOURCE=Upper
'*long tray name*'	a character string that is the name of the paper tray, for example, PAPERSOURCE='Optional Output Tray'. Names with blanks or special characters must be quoted.

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

Specifies the name of a paper type.

Used in: GOPTIONS statement; OPTIONS statement

Default: PLAIN

Restriction: hardware– dependent, PostScript printers require a PPD file; not supported by Java or ActiveX

See also: PPDFILE

Syntax

PAPERTYPE='*type-name*'

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

Sets the increment of the angle for device–resident text rotation.

Used in: GDEVICE procedure; GDEVICE Metagraphics window

Default: PATH=0

Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

PATH=*angle-increment*

angle-increment
is an integer in the range 0 to 360 that specifies the angle at which to rotate the text baseline. A value of 0 means that the device uses its default orientation. Specify 0 if your device does not perform string angling in hardware.

Details

For information about Metagraphics drivers, contact Technical Support.

PCLIP

Specifies whether a clipped polygon is stored in its clipped or unclipped form.

Used in: GOPTIONS statement

Default: NOPCLIP

Restriction: not supported by Java or ActiveX

See also: POLYGONCLIP

Syntax

PCLIP |NOPCLIP

PCLIP
stores clipped polygons with the graph in the default catalog WORK.GSEG, or in the catalog you specify.

NOPCLIP
stores the unclipped form of the polygon and causes the polygon to be clipped when replayed.

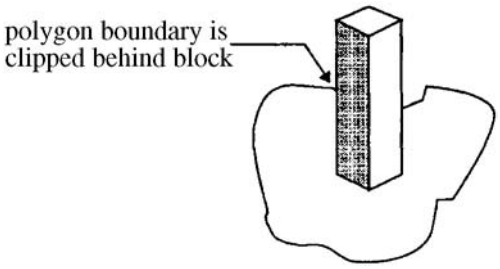
Details

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.

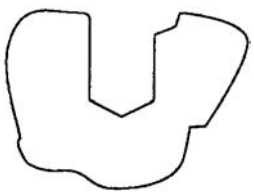
For example, suppose you create a block map like the one in Figure 15.3 on page 400.

Figure 15.3 Intersecting Polygons



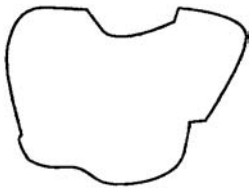
The block clips the boundary of the map area polygon. If you specify PCLIP, the map area polygon is stored in its clipped form, as shown in Figure 15.4 on page 400.

Figure 15.4 Clipped Polygon with PCLIP Option



NOPCLIP stores the map area in its unclipped form, as shown in Figure 15.5 on page 401.

Figure 15.5 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 polygons retain their original form.

PCOLS

Sets the number of columns in the graphics output area for portrait orientation.

Used in: GDEVICE procedure; GDEVICE Detail window
Default: device–dependent
See also: HPOS, LCOLS, PROWS

Syntax

PCOLS=*portrait-columns*

portrait-columns
must be a nonnegative integer up to three digits long (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 59 for more information.

PENMOUNTS

Specifies the number of active pens or colors.

Used in: GOPTIONS statement
Default: device-dependent
Restriction: not supported by Java or ActiveX

See also: MAXCOLORS

Syntax

PENMOUNTS=*active-pen-mounts*

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 only refers to foreground colors. 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

Specifies whether plotters draw graphics elements in order of color.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Restriction: not supported by Java or ActiveX
Default: device-dependent

Syntax

GOPTIONS: PENSORT | NOPENSORT

GDEVICE: PENSORT=Y | N

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 non-standard color names.

PIEFILL

Specifies whether to use the device's hardware pie-fill capability.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Default: device-dependent

Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: PIEFILL | NOPIEFILL

GDEVICE: PIEFILL=Y | N

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.

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

Specifies the type of clipping used when two polygons overlap.

Used in: GOPTIONS statement

Default: device-dependent
Restriction: not supported by Java or ActiveX
See also: PCLIP

Syntax

POLYGONCLIP | NOPOLYGONCLIP

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 15.3 on page 400 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

Specifies whether to use the hardware polygon-fill capability.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Default: device-dependent
Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: POLYGONFILL | NOPOLYGONFILL

GDEVICE: POLYFILL=Y | N

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.

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

Specifies data to send immediately after the data that is stored in the Gepilog field of the device entry is sent.

Used in: GOPTIONS statement

Default: Null string

Restriction: not supported by Java or ActiveX

See also: GEPILOG, PREGEPILOG

Syntax

POSTGEPILOG='*string*'

'string'
can be either of the following:

'*hex-string*'X

'*character-string*'

PROC GOPTIONS always reports the value as a hexadecimal string.

POSTGPROLOG

Specifies the data to send immediately after the data that is stored in the Gprolog field of the device entry is sent.

Used in: GOPTIONS statement

Default: Null string

Restriction: not supported by Java or ActiveX

See also: GPROLOG, PREGPROLOG

Syntax

POSTGPROLOG='*string*'

'string'
can be either of the following:
'*hex-string*'X
'*character-string*'
PROC GOPTIONS always reports the value as a hexadecimal string.

POSTGRAPH

Specifies host commands to be executed after the graph is produced.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host Commands window

Restriction: not supported by Java or ActiveX

See also: FILECLOSE

Syntax

POSTGRAPH1='*system-command(s)*'

POSTGRAPH2='*system-command(s)*'

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

Specifies the location of an external file containing PostScript Printer Description (PPD) information.

Used in: GOPTIONS statement

Restriction: PostScript printers only

See also: BINDING, COLLATE, DUPLEX, PAPERDEST, PAPERSIZE, PAPERSOURCE, PAPERTYPE, REVERSE

Syntax

PPDFILE=*fileref* | *'external-file'*

fileref
specifies a fileref that points to the PPD file 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 filename of the PPD file 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

Specifies data to send immediately before the data that is stored in the Gepilog field of the device entry is sent.

Used in: GOPTIONS statement
Default: Null string
Restriction: not supported by Java or ActiveX
See also: GEPILOG, POSTGEPILOG

Syntax

PREGEPILOG=*'string'*

'string'
can be either of the following:
*'hex-string'*X
'character-string'
PROC GOPTIONS always reports the value as a hexadecimal string.

PREGPROLOG

Specifies the data to send immediately before the data that is stored in the Gprolog field of the device entry is sent.

Used in: GOPTIONS statement
Default: Null string
Restriction: not supported by Java or ActiveX
See also: GPROLOG, POSTGPROLOG

Syntax

PREGPROLOG='*string*'

'string'
can be either of the following:
'*hex-string*'X
'*character-string*'
PROC GOPTIONS always reports the value as a hexadecimal string.

PREGRAPH

Specifies host commands to be executed before the graph is produced.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host Commands window
Restriction: not supported by Java or ActiveX
See also: FILECLOSE

Syntax

PREGRAPH1='*system-command(s)*'

PREGRAPH2='*system-command(s)*'

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

Specifies the command that translates the metafile into commands for the device.

Used in: GDEVICE procedure; GDEVICE Metagraphics window
Restriction: Used only with user-supplied Metagraphics drivers.
See also: INTERACTIVE

Syntax

PROCESS='*command*'

command
specifies 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. *Command* is a string up to 40 characters long.

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

Specifies the fileref for the file that contains input for the user-written part of the Metagraphics driver.

Used in: GDEVICE procedure; GDEVICE Metagraphics window
Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

PROCESSINPUT=*fileref*

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 36 *SAS/GRAPH: Reference* for additional information.

Details

For information about Metagraphics drivers, contact Technical Support.

PROCESSOUTPUT

Specifies the fileref for the file that receives output from the user-written part of the Metagraphics driver.

Used in: GDEVICE procedure; GDEVICE Metagraphics window
Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

PROCESSOUTPUT=*fileref*

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 36 for additional information.

Details

For information about Metagraphics drivers, contact Technical Support.

PROMPT

Specifies whether prompts are issued.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Restriction: not supported by Java or ActiveX
Default: device-dependent

Syntax

GOPTIONS: PROMPT | NOPROMPT

GDEVICE: PROMPT=0...7

PROMPT
causes all prompts to be displayed.

NOPROMPT
suppresses all prompts. NOPROMPT overrides the GWAIT= graphics option.

PROMPT=0...7
in the GDEVICE procedure, specifies the level of prompting:

0	provides no prompting
1	issues startup messages only. Startup 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 prom

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. △

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 hardcopy 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

Selects the prompt characters to be used by SAS/GRAPH device drivers.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Default: host dependent

Restriction: not supported by Java or ActiveX

See also: GSFLEN, HANDSHAKE

Syntax

PROMPTCHARS='*prompt-chars-hex-string*'X

prompt-chars-hex-string
is 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 list describes each byte in the string:

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 will be 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 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" on page 380), 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.

PROWS

Sets the number of rows in the graphics output area for portrait orientation.

Used in: GDEVICE procedure; GDEVICE Detail window

Default: device–dependent

See also: LROWS, PCOLS, VPOS

Syntax

PROWS=*portrait-rows*

portrait-rows
is a nonnegative integer up to three digits long (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 59 for more information.

QMSG

Specifies whether log messages are held until after the graphics output is displayed.

Used in: GDEVICE procedure; GDEVICE Detail window

Default: device–dependent

Syntax

GOPTIONS: QMSG | NOQMSG

GDEVICE: QMSG=Y | N

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

Specifies which rectangle fills should be performed by hardware.

Used in: GDEVICE procedure; GDEVICE Parameters window

Default: device-dependent

See also: FILL

Syntax

RECTFILL='*rectangle-fill-hex-string*'X

rectangle-fill-hex-string
is 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.

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 240 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

Controls the creation and disposition of rendered Bitstream fonts.

Used in: GOPTIONS statement

Default: MEMORY

Restriction: not supported by Java or ActiveX

See also: RENDERLIB

Syntax

RENDER=APPEND | DISK | MEMORY | NONE | READ

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

Specifies the SAS library in which rendered font files are stored.

Used in: GOPTIONS statement

Default: WORK

Restriction: not supported by Java or ActiveX

See also: RENDER

Syntax

RENDERLIB=*libref*

libref
specifies a previously defined libref that identifies the SAS library. The default library is WORK. See "LIBNAME Statement" on page 36 for more information on assigning a libref.

REPAINT

Specifies how many times to redraw the graph.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window

Default: device–dependent

Restriction: not supported by Java or ActiveX

Syntax

REPAINT=*redraw-factor*

redraw-factor
is a nonnegative integer up to three digits long (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

Resets graphics options to their defaults and/or cancels global statements.

Used in: GOPTIONS statement

Syntax

RESET=ALL | GLOBAL | *statement-name* | (*statement-name(s)*)

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).

Note: RESET=GOPTIONS sets all graphics options to defaults but does not cancel any global statements. △

Featured in: "Example 10. Creating a Bar Chart with Drill-Down Functionality for the Web" on page 321

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

Specifies whether to print the output in reverse order, if reverse printing is supported by the device.

Used in: GOPTIONS statement

Default: NOREVERSE

Restrictions: hardware-dependent, PostScript printers require a PPD file; not supported by Java or ActiveX

See also: PPDFILE

Syntax

REVERSE | NOREVERSE

Details

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/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

Specifies whether and how to rotate the graph.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window
Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: ROTATE=LANDSCAPE | PORTRAIT

GOPTIONS: ROTATE | NOROTATE

GDEVICE: ROTATE=LANDSCAPE | PORTRAIT

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).

If you do not specify a rotation, a default is searched for in this order:
1. the ORIENTATION setting on an OPTIONS statement
2. device-dependent default.

ROTATION

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 in: GDEVICE procedure; GDEVICE Metagraphics window
Default: ROTATION=0
Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

ROTATION=*angle-increment*

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 to 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

Specifies the number of rows the device–resident font uses in graphics output.

Used in: GDEVICE Chartype window; GDEVICE procedure; CHARREC= option
Default: 0
See also: CHARREC

Syntax

See “CHARREC” on page 337 for syntax.

Details

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 will be 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

Specifies whether a font is scalable.

Used in: GDEVICE Chartype window; GDEVICE procedure; CHARREC= option

Default: device– dependent

See also: CHARTYPE

Syntax

See “CHARREC” on page 337 for syntax.

Details

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

Specifies a SAS/GRAPH font to use if the default device-resident font cannot be used.

Used in: GOPTIONS statement

Default: SIMULATE

Restriction: not supported by Java or ActiveX

Syntax

SIMFONT=*SAS/GRAPH-font*

SAS/GRAPH-font
specifies a SAS/GRAPH font to use instead of the default device-resident font. By default, this is the 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 11, “Specifying Fonts in SAS/GRAPH Programs,” on page 155 for details.

SPEED

Selects pen speed for plotters with variable speed selection.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Default: device– dependent
Restriction: not supported by Java or ActiveX

Syntax

SPEED=*pen-speed*

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

Specifies whether to reverse BLACK and WHITE in the graphics output.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Defaults: GOPTIONS: NOSWAP; GDEVICE: SWAP=N
Restriction: not supported by Java or ActiveX

Syntax

GOPTIONS: SWAP | NOSWAP

GDEVICE: SWAP=Y | N

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 only affects 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

Specifies the method used to render system fonts.

Used in: GOPTIONS statement
Default: device– dependent
Restriction: not supported by Java or ActiveX

Syntax

SWFONTRENDER = POLYGON | SCANLINE

SWFONTRENDER = POLYGON
uses polygon rendering

SWFONTRENDER = 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

Specifies whether to use the device's symbol-drawing capability.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Default: device– dependent
Restriction: not supported by Java or ActiveX
See also: SYMBOLS

Syntax

GOPTIONS: SYMBOL | NOSYMBOL

GDEVICE: SYMBOL=Y | N

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.

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

Specifies which symbols can be generated by hardware.

Used in: GDEVICE procedure; GDEVICE Parameters window

Default: device–dependent

See also: "SYMBOL Statement" on page 252

Syntax

SYMBOLS='*hardware-symbols-hex-string*'X

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.

Bit to turn on	Symbol Description	Symbol
1	PLUS	
2	X	
3	STAR	
4	SQUARE	
5	DIAMOND	
6	TRIANGLE	
7	HASH	
8	Y	
9	Z	
10	PAW	
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

Displays the output as it would appear on a different device. Also, specifies the device driver for the PRINT command.

Alias: TARGET

Used in: GOPTIONS statement

Restriction: not supported by Java or ActiveX

Syntax

TARGETDEVICE=*target-device-entry*

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:

- preview graphics output on your monitor as it would appear on a different output device. For details, see "Previewing Output" on page 109.
- print output from the Graph window or the Graphics Editor window with the PRINT command. For details, see "Printing Your Graph" on page 110.
- specify a device driver for graphics output created by the ODS HTML statement.

TRAILER

Specifies the command that creates TRAILER records for the Metagraphics driver.

Used in: GDEVICE procedure; GDEVICE Metagraphics window
Restriction: Used only with user-supplied Metagraphics drivers
See also: TRAILERFILE

Syntax

TRAILER='*command*'

command
specifies a command that runs a user-written program that creates the TRAILER file. *Command* is a string up to 40 characters long.

Details

For information about Metagraphics drivers, contact Technical Support.

TRAILERFILE

Specifies the fileref of the file from which the Metagraphics driver reads TRAILER records.

Used in: GDEVICE procedure GDEVICE Metagraphics window
Restriction: Used only with user-supplied Metagraphics drivers
See also: TRAILER

Syntax

TRAILERFILE=*fileref*

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 36 for additional information on the FILENAME statement.

Details

For information about Metagraphics drivers, contact Technical Support.

TRANSPARENCY

Specifies whether the background of the image should appear to be transparent when the image is displayed in the browser.

Used in: GOPTIONS statement

Default: NOTRANSPARENCY

Restriction: This option is supported by the ACTIVEX and ACTXIMG drivers when the output is used in a PowerPoint presentation and by the GIF series of drivers only.

Syntax

TRANSPARENCY | NOTRANSPARENCY

Details

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. As an alternative, consider using the UPNGT device. △

TRANTAB

Selects a translate table for your system that performs ASCII-to-EBCDIC translation.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Host File Options window

Default: host dependent

Restriction: not supported by Java or ActiveX

Syntax

TRANTAB=*table* | *user-defined-table*

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

Specifies the type of output device to which graphics commands are sent.

Used in: GDEVICE procedure; GDEVICE Detail window

Default: device–dependent

Syntax

TYPE=CAMERA | CRT | EXPORT | PLOTTER | PRINTER

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 SAS-supplied device drivers.

UCC

Sets the user-defined control characters for the device.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Parameters window
Restriction: device–dependent; not supported by Java or ActiveX

Syntax

UCC='*control-characters-hex-string*'X

control-characters-hex-string
is a hexadecimal string that can be up 32 bytes (64 characters) long. You only need to specify up to the last non-zero 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

Determines whether user input is enabled for the device.

Used in: GOPTIONS statement
Default: NOUSERINPUT
Restrictions: GIFANIM driver only; not supported by all browsers

Syntax

USERINPUT | NOUSERINPUT

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

Sets the vertical offset from the lower-left corner of the display area to the lower-left corner of the graph.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Restriction: not supported by Java or ActiveX

See also: HORIGIN

Syntax

VORIGIN=*vertical-offset* <IN | CM | PT>

***vertical-offset* <IN | CM | PT>**
must be a nonnegative number and can be followed by a unit specification, either IN for inches (default), or 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 59 for details.

VPOS

Sets the number of rows in the graphics output area.

Used in: GOPTIONS statement

Default: device–dependent: the value of the LROWS or PROWS device parameter

Restriction: not supported by Java or ActiveX

See also: HPOS, LROWS, PROWS

Syntax

VPOS=*rows*

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 59 for more information.

VSIZE

Sets the vertical size of the graphics output area.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Restriction: partially supported by Java or ActiveX

See also: HSIZE, YMAX

Syntax

VSIZE=*vertical-size* <IN | CM | PT>

***vertical-size* <IN | CM | PT>**
specifies the height of the graphics output area; *vertical-size* must be a positive number and can be followed by a unit specification, either IN for inches (default), or 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

Allows programs that are run in the current version of SAS to run with selected Version 6 defaults.

Used in: GOPTIONS statement

Default: NOV6COMP

Restriction:
Partially supported by Java or ActiveX

Ignored unless OPTIONS NOGSTYLE is also specified

Syntax

V6COMP | NOV6COMP

V6COMP
causes SAS/GRAPH programs to use these Version 6 behaviors:

- 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

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 in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Restriction: Ignored by default display drivers, universal printing drivers, Java, and ActiveX

See also: HSIZE, PAPERSIZE, XPIXELS

Syntax

XMAX=*width* <IN | CM | PT>

width
is a positive number that can be followed by a unit specification, either IN for inches (default), or 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 hardcopy 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 59 for more information.

XPIXELS

Specifies the width of the addressable display area in pixels and in conjunction with XMAX determines the horizontal resolution for the device.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Default: device–dependent

See also: XMAX

Restriction: Partially supported by Java and ActiveX

Syntax

XPIXELS=*width-in-pixels*

width-in-pixels
is a positive integer up to eight digits long (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 on graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: User's 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, using this formula:

  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* is 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

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 in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Restriction: ignored by default display drivers and universal printing drivers; not supported by Java or ActiveX

See also: PAPERSIZE, VSIZE, YPIXELS

Syntax

YMAX=*height* <IN | CM | PT>

height

is a positive number that can be followed by a unit specification, either IN for inches (default), or 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.

Details

See "XMAX" on page 432.

YPIXELS

Specifies the height of the addressable display area in pixels and in conjunction with YMAX determines the horizontal resolution for the device.

Used in: GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window

Default: device–dependent

See also: YMAX

Restriction: Partially supported by Java and ActiveX

Syntax

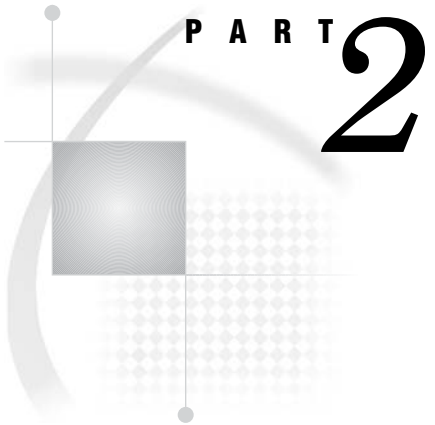
YPIXELS=*height-in-pixels*

height-in-pixels
is a positive integer up to eight digits long (0...99999999).

Details

See "XPIXELS" on page 433.

Note: This option overrides the OutputHeight style attribute in the graph styles. For more information on graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: User's Guide*. △



PART 2

Bringing SAS/GRAPH Output to the Web

CHAPTER

16

Introducing SAS/GRAPH Output for the Web

Which Device Driver or Macro Do I Use?

Generating a web presentation that includes graphics requires that you use a device driver or macro that generates web output. Determining which device driver or macro to use requires that you consider issues such as

- What type of graph do I need?
- What procedure, if any, generates the graph that I need?
- In which operating environments do I need to generate the presentation?
- In which operating environments do I need to deliver the presentation?
- Will my audience need to install additional software to view the presentation?
- What interactive features do I want in my presentation?

The following topics describe the types of web presentations that are available, help you decide which type you need, and tell you how to generate the presentation and deliver it to your audience. The primary purpose of these topics is to help you determine which device driver or macro you need to use.

- "Types of Web Presentations Available" on page 440 describes each type of web presentation, their features, and which device driver or macro you need to use to create that type of presentation.
- "Selecting a Type of Web Presentation" on page 447 guides you through the process of determining which device driver or macro to use. If the type of presentation you need to generate can be generated with multiple device drivers, then additional factors determine which driver to use.
- "Generating Web Presentations" on page 451 summarizes the methods by which each type of web presentation is created.

Types of Web Presentations Available

Delivering information via the web frequently requires a web presentation that includes not only tables but graphics as well. SAS/GRAPH provides three basic ways to display presentations that include graphics. Presentations can be displayed

by an ActiveX control
: The ActiveX control displays the output of SAS/GRAPH procedures. It enables such features as pop-up data tips, drill-down links, and interactive menus. For more information, see "Presentations That Use The ActiveX Control" on page 440.

by a Java applet
: Java applets display the output of SAS/GRAPH procedures and macros. Depending on the applet, it may enable such features as data tips, drill-down links, or interactive features available through a pop-up menu. For more information, see "Presentations That Use Java Applets" on page 441.

as a static graph
: You can also generate graphs that do not have any interactive features but do have interactive capabilities such as data tips or drill-down links. Static graphs can be generated as GIF, JPEG, or PNG files. For more information, see "Presentations that Use Static Images" on page 445.

For additional information about SAS/GRAPH output for the Web, including samples, refer to

```
http://support.sas.com/rnd/datavisualization
```

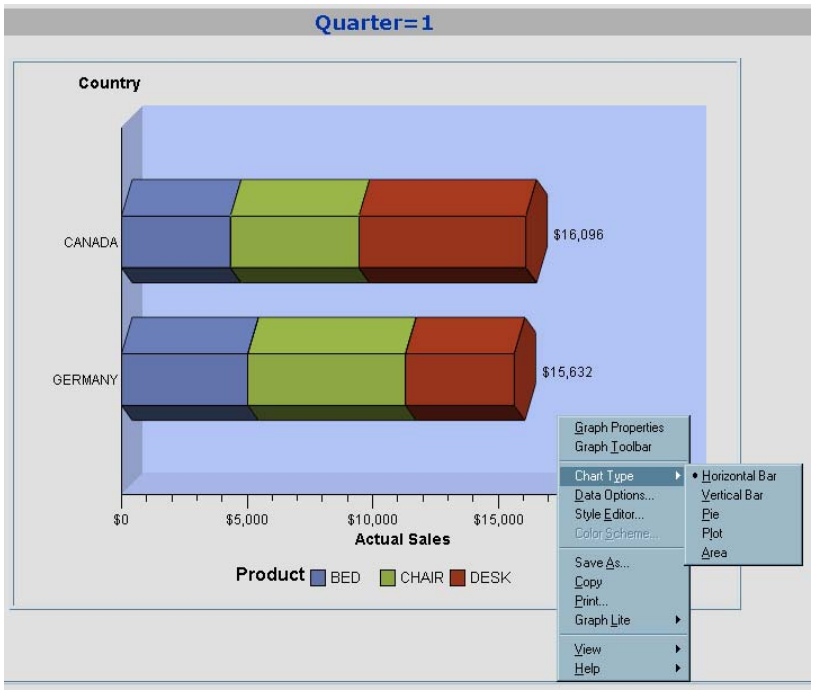
Presentations That Use The ActiveX Control

The SAS/GRAPH ActiveX control displays the output of SAS/GRAPH procedures and enables extensive interactive features via a pop-up menu. The pop-up menus enable you to rotate, and zoom, and to control the properties of graphs such as its colors, legends, and axes.

You can enable pop-up data tips and drill-down links with presentations created for the ActiveX control.

Display 16.1 on page 441 shows output from the GCHART procedure as displayed by the ActiveX control. (You can open the pop-up menu for the ActiveX control by positioning your cursor over the graph and pressing the right mouse button.)

Display 16.1 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 ActiveX control displays output from the G3D, GAREABAR, GBARLINE, GCHART, GCONTOUR, GMAP, GPLOT, GRADAR, and GTILE procedures.

To create a graph to be displayed by ActiveX, specify DEVICE=ACTIVEX on your GOPTIONS statement. See "Using ODS HTML with a SAS/GRAPH Procedure" on page 451 and Chapter 17, "Creating Interactive Output for ActiveX," on page 453 for more information.

Presentations That Use Java Applets

If you want to deliver your presentation to more operating environments than just Windows, you can use one of the following Java applets:

Graph, Map, Tilechart, and Contour applets
: These applets display the output of SAS/GRAPH procedures and offer many interactive features. The Graph and Map applets.

Treeview and Constellation applets
: These applets generate hierarchical treeview diagrams and constellation diagrams, respectively, and are generated with the DS2TREE and DS2CONST macros.

Metaview applet
: The Metaview applet displays the output of SAS/GRAPH procedures, and it enables pop-up data tips, drill-down links, and zooming.

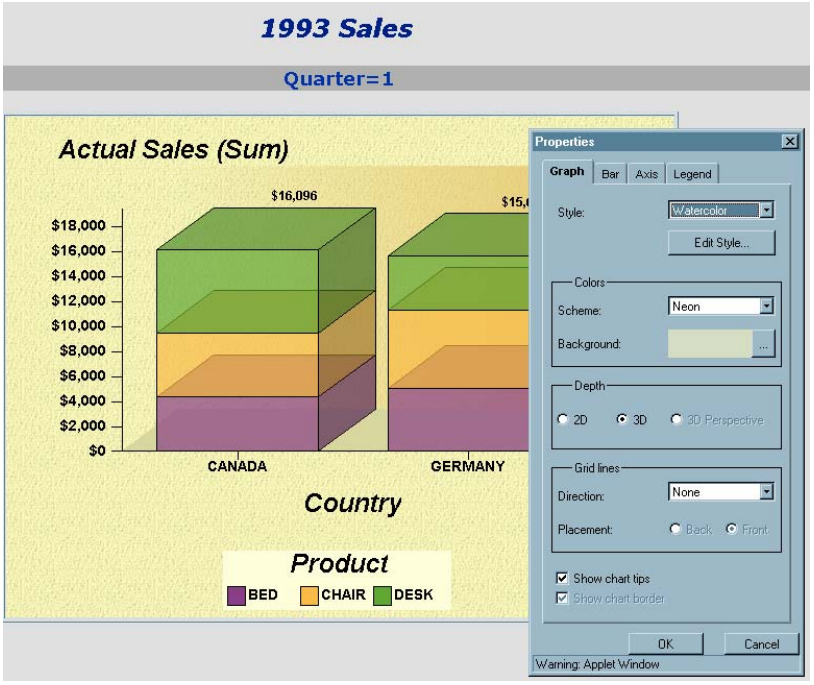
Graph, Map, Tilechart, and Contour Applets

Like the ActiveX control, the Graph, Map, Tilechart, and Contour applets display the output of SAS/GRAPH procedures and enable extensive interactive features. The

Graph, Map, Tilechart, and Contour applets enable interactive features such as data tips and drill-down links, and they provide pop-up menus which enable the user to change properties such as the graph's colors, legends, and axes.

Display 16.2 on page 442 shows PROC GCHART output displayed by the Java Graph applet with a Properties dialog box. You can open the pop-up menu for these applets by positioning your cursor over the graph and pressing the right mouse button.

Display 16.2 Sample Java Presentation



These applets display the output of the following SAS/GRAPH procedures:

Graph Applet	G3D Scatter Plots, GCHART, GPLOT
Contour Applet	G3D Surface Plots, GCONTOUR
Map Applet	GMAP
Tilechart Applet	GTILE

To create a graph to be displayed by one of these applets, specify DEVICE=JAVA on your GOPTIONS statement. For more information, see "Using ODS HTML with a SAS/GRAPH Procedure" on page 451 and Chapter 18, "Creating Interactive Output for Java," on page 469.

Treeview Applet

This applet displays a treeview diagram, which shows the parent-child relationships in a tree structure. In a treeview diagram, each child node has exactly one parent, and each parent node has zero or more children. In other words, the relationships in a treeview diagram are one-to-many. A treeview diagram is ideal for displaying such data as organizational charts or the hierarchical relationships of the pages of a Web site.

By default, the Treeview applet zooms in on the portion of the tree that is in the center of the display, as if you were looking through a fish-eye lens. Nodes in the center of the display are spread apart and shown with more detail, including node labels. Nodes near the periphery of the display are compressed and shown with less detail.

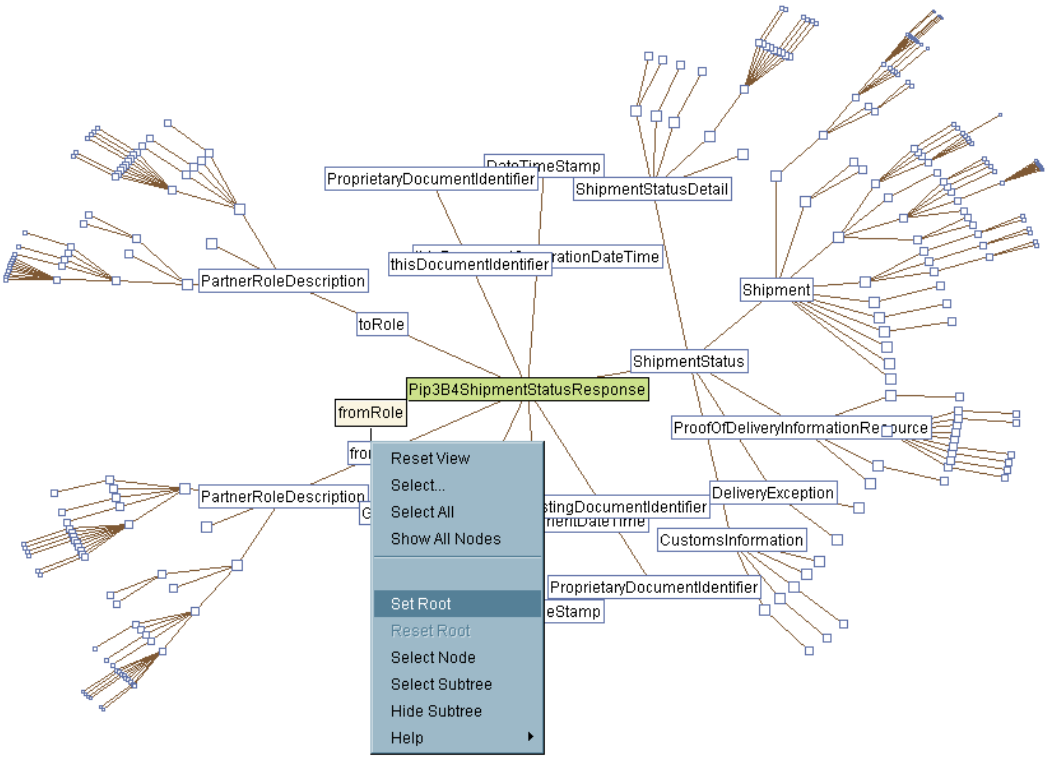
Initially, the Treeview applet places the root node in the center of the display. You can click and drag the diagram to change the portion of the diagram that is in the center of the display.

The Treeview applet supports a pop-up menu that enables you to search for nodes, select or hide subtrees, and so on. You can add hotspots that link to Web pages when the user clicks on a node.

For example, Display 16.3 on page 443 shows a treeview diagram (with the pop-up menu opened) displaying the structure of an XML Document Type Definition.

To generate a treeview diagram, use the DS2TREE macro. For more information, see Chapter 24, "Creating Interactive Treeview Diagrams," on page 543.

Display 16.3 Sample Treeview Diagram



Constellation Applet

The Constellation applet displays a general node-link diagram. Each node can be linked to one or more other nodes. Unlike the Treeview applet, the Constellation applet does not require a hierarchical relationship between the nodes. (Although it can be used to display hierarchical relationships, the Constellation applet does not automatically place the root node at the center of the display.)

The Constellation applet supports node and link properties, which determine the color and size of the nodes and the color and thickness of the link joining the nodes. These properties indicate the relative strength of the relationship between the nodes.

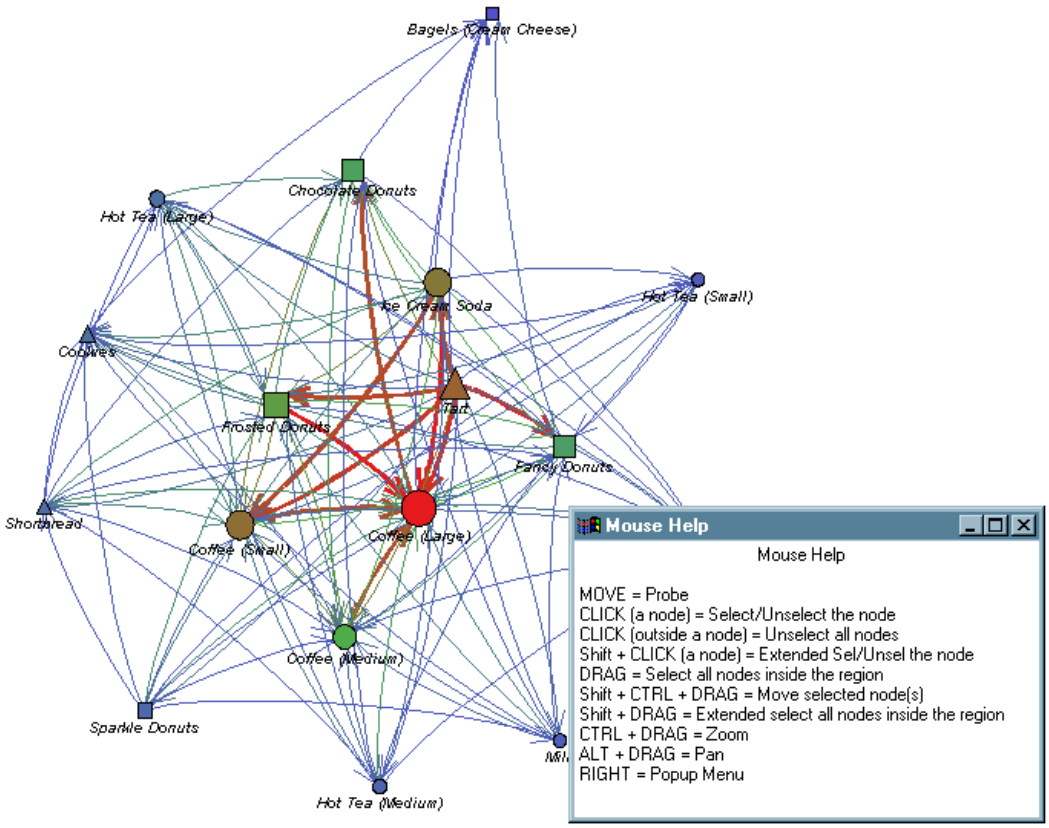
Like the Treeview applet, by default, the Constellation applet zooms in on the portion of the diagram that is in the center of the display, as if you were looking through a fish-eye lens. Nodes in the center of the display are spread apart and shown with more detail, including node labels. Nodes near the periphery of the display are compressed and shown with less detail. You can click and drag the diagram to change the portion of the diagram that is in the center of the display.

The Constellation applet has a pop-up menu that supports several functions such as highlighting specific links and searching for specific nodes. You can add hotspots that link to Web pages when the user clicks on a node.

Display 16.4 on page 444 shows a constellation diagram (with the Mouse Help menu displayed).

To generate the Constellation applet, use the DS2CONST macro. For more information, see Chapter 25, "Creating Interactive Constellation Diagrams," on page 553.

Display 16.4 Sample Constellation Diagram

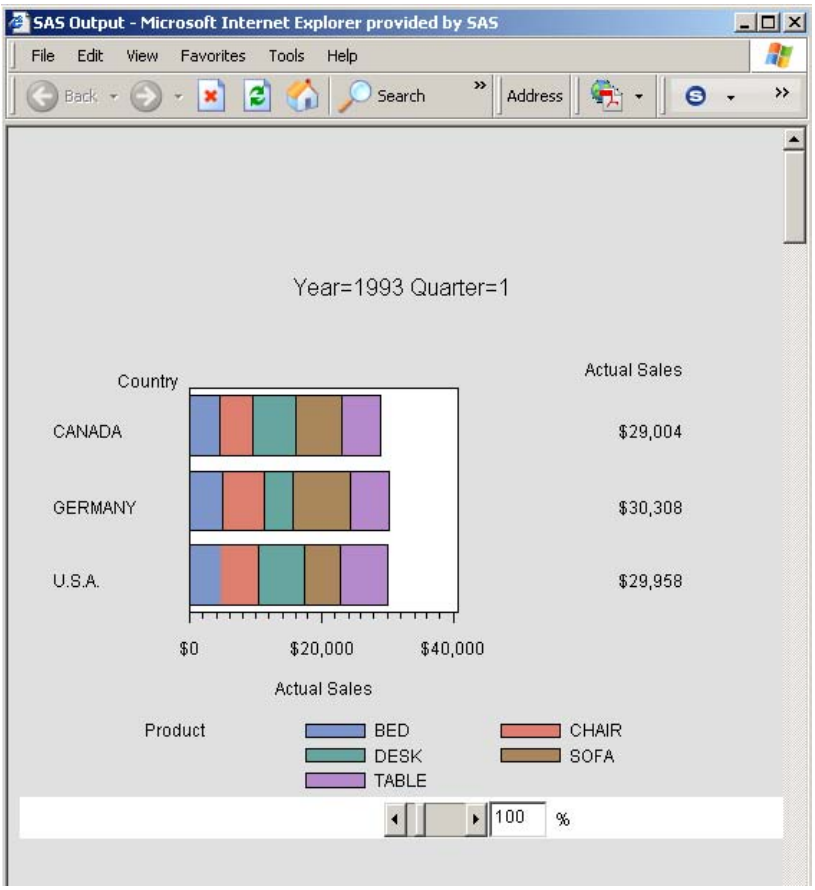


Metaview Applet

The Metaview applet displays the output of SAS/GRAPH procedures and enables interactive features that are not available with static images such as GIFs or JPEGs. It enables zooming and scrolling and supports pop-up menus with customized user-selectable links. When you generate a graph with the Metaview applet, you can specify background colors and text fonts, and enable drill-down links to HTML files, metagraphics files, and sets of metacodes.

Display 16.5 on page 445 shows the zoom control that the Metaview applet provides.

Display 16.5 Sample Metaview Applet



The Metaview applet displays output from the G3D, GANNO, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GREPLAY, and GSLIDE procedures. To create a graph to be displayed by the Metaview applet, specify DEVICE=JAVAMETA on your GOPTIONS statement.

For additional information, see Chapter 22, “Generating Interactive Metagraphics Output,” on page 531.

Presentations that Use Static Images

If you do not need any interactive features in your presentations, then you can specify one of the following device drivers to generate a presentation that uses a GIF, JPEG, or PNG file.

ACTXIMG or JAVAIMG
: create a web presentation that uses a static PNG image instead of an interactive applet. The images are identical to the images generated with the ACTIVEX and JAVA device drivers.

GIF, JPEG, or PNG
: create web presentations that use static GIF, JPEG, or PNG images.

GIFANIM
: generates a series of images that are displayed in sequence from a single GIF file.

To generate a web presentation that uses one of these drivers, specify the driver name with the DEVICE= option in your GOPTIONS statement. All of these device drivers generate output from SAS/GRAPH procedures.

For more information, refer to the following topics:

ACTXIMG Presentations

You can use the ACTXIMG device driver to create a presentation that uses a PNG file that is identical in appearance to the image produced with the ACTIVEX device driver.

A presentation generated with the ACTXIMG driver supports data tips and drill-down links for GCHART, GBARLINE, and GPLOT (except for high-low plots) output.

To render your output (create the PNG file), the ActiveX control must be installed on the PC where your SAS session is running. Because of this requirement, ACTXIMG presentations can be generated only on PCs. When you specify the ACTXIMG device driver, the output is rendered when your web presentation is generated, and the user does not need to have the ActiveX control installed to view it.

Note: The ACTXIMG device cannot be used with the ODS PDF, PCL, PS, or PRINTER destinations on 64-bit machines. SAS uses the JAVAIMG device instead. △

You can use the ACTXIMG device driver to generate presentations with the same procedures that are supported by the ACTIVEX driver: G3D, GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, and GTILE.

JAVAIMG Presentations

You can use the JAVAIMG device driver to create a presentation that uses a PNG file that is identical in appearance to the image produced with the JAVA device driver.

The appropriate Java applet (Graph, Map, Tilechart, or Contour applet) is required to render your output (create the PNG file). The appropriate Java applet must be installed on the machine where your SAS session is running. When you specify the JAVAIMG device driver, the output is rendered when your web presentation is generated, and the user does not need to have any Java applet files installed to view it.

You can use the JAVAIMG device driver to generate presentations with the same procedures that are supported by the JAVA driver: G3D, GCHART, GCONTOUR, GPLOT, and GTILE.

GIF, JPEG, and PNG Presentations

For Web presentations generated with the GIF, JPEG, or PNG device drivers, you can add pop-up data tips that are displayed when the cursor is over a portion of the image and links to other Web pages.

You can use the GIF, JPEG, or PNG device drivers to generate presentations to display output from the G3D, GANNO, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GREPLAY, and GSLIDE procedures.

To create a web presentation with one of these devices, specify DEVICE=GIF, JPEG, or PNG in your GOPTIONS statement.

Animated GIF Presentations

An animated presentation is a series of static images that are displayed automatically one after the other. Specify DEVICE=GIFANIM in your GOPTIONS statement to generate a web presentation that displays a series of images from a single GIF file. You can control the rate at which the successive images are presented.

You can generate animated GIF presentations from the G3D, GANNO, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GREPLAY, and GSLIDE procedures.

For more information, see Chapter 21, "Generating Web Animation with GIFANIM," on page 519.

Selecting a Type of Web Presentation

The type of web presentation that you choose to generate depends on several factors such as the type of graphs you need, the operating environment in which you want to generate your presentation, and the operating environments in which you plan to deliver your web presentation.

To determine which type of web presentation you need, consider the following questions:

How is your graphical output produced?
: The structure of your data and the information that you need to generate from this data determine the type of graph that you need to produce. The type of graph that you need determines which procedure or macro you need to use to produce your graph. Which procedure or macro, if any, you need to use may determine which device drivers you can use.

What features are supported for each type of presentation?
: Each type of web presentation enables different features such as data tips, drill-down links, and pop-up menus. Whether you need extensive interactive capabilities or just data tips can determine which device driver you need to use.

What does your audience need to view the presentation?
: Which device or macro you use to generate your web presentation determines whether the presentation can be viewed on multiple platforms and whether it requires any software except a supported browser.

How is the graphical output produced?

Which type of graph you need to produce is determined by the structure of your data and the information that you need to convey to your audience. For example, treeview diagrams and bar charts convey very different types of information. If you need to create a web presentation that includes graphics that are produced by one of the SAS/GRAPH procedures, then you need to use one of the device drivers that supports that procedure. Assuming that you know which type of graph you need, then you can determine which device drivers or macros you can use.

Table 16.1 on page 448 lists the procedures that are supported by each device driver and the diagrams that are produced by each macro.

Note: To generate a web presentation using the ACTXIMG device driver, the ActiveX control must be installed on the PC on which your SAS session is running. △

Table 16.1 How is the graphical output produced?

Driver or Macro	How Output is Produced
ACTXIMG	G3D, GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GTILE
JAVAIMG	G3D, GCHART, GCONTOUR, GKPI, GPLOT, GMAP, GTILE
ACTIVEX	G3D, GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GTILE
JAVA	G3D, GCHART, GCONTOUR, GMAP, GPLOT, GTILE
SVG	G3D, GANNO, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GREPLAY, GSLIDE
GIF, JPEG, PNG	G3D, GANNO, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GREPLAY, GSLIDE
GIFANIM	G3D, GANNO, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GREPLAY, GSLIDE
JAVAMETA	G3D, GANNO, GBARLINE, GCHART, GCONTOUR, GPLOT, GMAP, GRADAR, GREPLAY, GSLIDE
DS2TREE, DS2CONST	These macros create treeview diagrams or constellation diagrams, respectively, without involving a SAS/GRAPH procedure.

For example, if you need a radar chart, you can use the ACTXIMG, ACTIVEX, or JAVAMETA driver (as well as other drivers). Which device driver you choose depends on what additional features (such as interactive capabilities) you need and on how you plan to deliver your web presentation.

If you need to graph hierarchical relationships, consider using the DS2TREE macro to generate a treeview diagram. If you need to show relationships that are not hierarchical or if you need to show the relative affinity of the relationships, then consider using the DS2CONST macro to generate a constellation diagram.

What features are supported for each type of presentation?

The following table shows, for each type of Web presentation, what features are available to a viewer when viewing the presentation in a browser. You can see from the table that presentations that involve a Web executable, such as Java applets or the ActiveX control, enable interactive manipulation via pop-up menus. Presentations that use GIF, JPEG, and PNG files provide static images with no interactivity besides pop-up data tips and drill-down links.

After you have determined which device drivers or macros you can use, you then need to determine which extra features you need in your web presentation. For example, you may not want or need to give your audience the ability to subset the graph's data or change the graph from a bar chart to a pie chart.

The following table shows which features are supported for each device driver or macro.

Table 16.2 What features are supported for each type of presentation?

Driver or Macro	Features Supported
ACTXIMG	pop-up data tips and drill-down links (for selected output), static graphics with no interactivity
JAVAIMG	static graphics with no interactivity
ACTIVEX	pop-up data tips, drill-down links, interactivity via pop-up menus
JAVA	pop-up data tips, drill-down links, interactivity via pop-up menus
SVG	drill-down links, zooming supported in SVG-enabled browsers
GIF, JPEG, PNG	drill-down links, static graphics with no interactivity
GIFANIM	Slide show of static images with no interactivity
JAVAMETA	Pop-up data tips, drill-down links, some interactivity such as zooming and slide shows
DS2TREE, DS2CONST	Pop-up data tips, drill-down links, interactivity via pop-up menus

Data tips and drill-down links for ACTXIMG are supported for output from GCHART, GPLOT (except for high-low plots), GBARLINE, and GRADAR.

The pop-up menus available with the JAVA and ACTIVEX device drivers typically enable your audience to change many aspects of the graph such as changing chart types, subsetting data, changing the variable used as the response variable, turning data tips on or off, or changing the colors used the graph. Static graphs do not offer any of these interactive features. Web presentations that use the JAVAMETA driver may enable a zoom control, and page selection and slide show controls for presentations that include multiple images.

What does your audience need to view the presentation?

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 at

```
http://support.sas.com/documentation/installcenter
```

Select the *System Requirements* link for the appropriate operating system environment and search for the section on viewing HTML pages created for Java and ActiveX.

It is recommended that graphs be displayed on a device that has at least 16-bit color (that is, more than 8-bit, 256 colors).

Depending on how the presentation is generated, there may be additional requirements. The following table shows, for each type of Web presentation, what is required on a viewer's machine besides a supported browser.

Table 16.3 What does your audience need to view the presentation besides the browser?

Driver or Macro	Additional Requirements
ACTXIMG	None
JAVAIMG	None
ACTIVEX	The presentation must be viewed with Internet Explorer on a Windows system with the SAS ActiveX control installed locally.
JAVA	The Java applet files must be installed locally or on a server accessible by the client machine, and Java 1.4 plug-in must be installed on each client machine. On Windows systems, the user is prompted to install the plug-in if it is not already installed. On other systems, the plug-in can be installed from the Sun Microsystems site (http://www.sun.com) or from one of the SAS Third Party Software Components CDs.
SVG	The presentation must be viewed in an SVG-enabled browser.
GIF, JPEG, PNG	None
GIFANIM	None
JAVAMETA	The Java applet files must be installed locally or on a server accessible by the client machine. The Java plug-in is not required on the client machine; the Metaview applet works with the Java Virtual Machine that is built into the supported browsers.
DS2TREE, DS2CONST	The Java applet files must be installed locally or on a server accessible by the client machine, and Java 1.4 plug-in must be installed on each client machine. On Windows systems, the user is prompted to install the plug-in if it is not already installed. On other systems, the plug-in can be installed from the Sun Microsystems site (http://www.sun.com) or from one of the SAS Third Party Software Components CDs.

Presentations generated with the ACTIVEX driver can be viewed only with Internet Explorer on Windows PCs, and the ActiveX control must be installed locally on each PC.

Presentations generated with the JAVAMETA driver can be viewed in any supported browser and offer limited interactivity, but do not require that a Java plug-in be installed.

Recommendations

If you will be delivering your presentation on Windows only and you want your audience to be able to interact with the graph, then you can use the ACTIVEX device driver. If you will be delivering your presentation to other operating environments, but you still want to use interactive features, then you can use the JAVA device driver. However, the ACTIVEX and JAVA device drivers require that your audience install the ActiveX control and Java plug-in, respectively.

If you want the look of the ACTIVEX or JAVA driver, but do not need the interactive capability or do not want to require that your audience install the ActiveX control or the Java plug-in, then use the ACTXIMG or JAVAIMG device drivers.

If you need data tips, drill-down capability, or limited interactivity such as zoom, but you do not want to require that your audience install the Java plug-in or the ActiveX control, then you can use the JAVAMETA device driver.

If you need only data tips and drill-down capability, then you can use the GIF, JPEG, or PNG device driver.

Generating Web Presentations

There are two basic methods in which you can generate a web presentation:

- using the ODS HTML destination with a SAS/GRAPH procedure.
- using the DS2TREE or DS2CONST macro.

Using ODS HTML with a SAS/GRAPH Procedure

The recommended method for getting procedure output on the Web is with ODS. By using the ODS HTML statement in a program with one or more SAS/GRAPH procedures, you can create an HTML file and its associated SAS/GRAPH (or tabular) output.

At a minimum, to use ODS with SAS/GRAPH you must do the following:

1. Use the an ODS HTML statement to open the HTML destination. For device drivers that generate image output files, use the PATH= option to ensure that all output files are stored in the same location.
2. Run a graphics procedure. If your procedure supports run-group processing, be sure include a QUIT statement.
3. Close the HTML destination to write your HTML file.

If the LISTING destination is open, then SAS/GRAPH creates an additional copy of your output. To improve performance, you might want to close the LISTING destination while you are generating Web output.

Using DS2TREE and DS2CONST Macros

The following macros generate a Web presentation from a SAS data set:

- DS2TREE generates treeview diagrams
- DS2CONST generates constellation diagrams.

To use these macros, simply define your data, then call one of these macros using the appropriate options. For these macros, you do not use ODS or call a SAS/GRAPH procedure. For additional information, refer to Chapter 24, “Creating Interactive Treeview Diagrams,” on page 543 and Chapter 25, “Creating Interactive Constellation Diagrams,” on page 553.

CHAPTER

17

Creating Interactive Output for ActiveX

Overview

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 SAS installed locally, the control is run automatically when the HTML output file is displayed in Internet Explorer. For your Web users who do not have the SAS system installed locally, 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 455.

You can enhance your ActiveX presentations by adding drill-down links (see “Configuring Drill-Down Links with ACTIVEX” on page 460) and configuring interactive features (see “Specifying Parameters and Attributes for Java and ActiveX” on page 485).

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 8, “Exporting Your Graphs to Microsoft Office Products,” on page 113.

The following table lists the procedures and statements that generate output that can be displayed in the SAS/GRAPH ActiveX Control.

Table 17.1 Procedures and Statements that Generate Output for the SAS/GRAPH ActiveX Control

Procedure	Statements
GAREABAR	HBAR, VBAR
GBARLINE	BAR, PLOT
GCHART	BLOCK, HBAR, HBAR3D, VBAR, VBAR3D, PIE, PIE3D, DONUT
GCONTOUR	PLOT
GMAP	CHORO, BLOCK, PRISM (see note)
GPLOT	BUBBLE, BUBBLE2, PLOT, PLOT2
GRADAR	CHART
G3D	PLOT, SCATTER
GTILE	FLOW, TILE, TOGGLE

Note: 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.

When to Use the ACTIVEX Device

If your Web users are using the Windows operating environment and the Internet Explorer Web browser, the SAS/GRAPH ActiveX Control might be preferable over a Java applet from a performance standpoint. In general, the interactive features of the SAS/GRAPH ActiveX Control are comparable to those that are provided in Java through the Java applets. Some features differ, as you can see in the comparison table that is presented in the “Parameter Reference for Java and ActiveX” on page 488. Also, the JAVA device does not display output that is generated with the GAREABAR, GBARLINE, or GRADAR procedures.

Unlike the JAVA device, you can use the ACTIVEX device to embed interactive graphics in Microsoft Word documents by using the ODS RTF statement, as shown in “Generating an ActiveX Graph for a Microsoft Word Document” on page 461 and in “Importing Your Graphs into Microsoft Office” on page 120. You can also copy the ActiveX window out of Internet Explorer and paste it into a Microsoft Word, Excel, or PowerPoint document.

If you have created a graph with the ACTIVEX device but you do not need the interactivity that it provides, then use the ACTXIMG device, as described in “Developing Web Presentations with the JAVAIMG and ACTXIMG Devices” on page 510. The ACTXIMG device creates a static snapshot of the graph in a PNG file. The graph has the same look as the graph that is produced with the ACTIVEX device, but the graph does not support interactivity. You can use the ACTXIMG device only on

Windows systems. Although you do not need the SAS/GRAPH ActiveX Control when you are viewing the ACTXIMG output, to produce the output file, you must install the SAS/GRAPH ActiveX Control on your computer. See "Installing the SAS/GRAPH ActiveX Control" on page 455.

You can generate output for the SAS/GRAPH ActiveX Control even if you are not working in the Windows operating environment. For example, you can generate HTML output for ActiveX in the UNIX operating environment, even though you cannot run Internet Explorer in that environment. Displaying the HTML in Internet Explorer on Windows will display the output as if it was generated in that operating environment. You can also run your SAS jobs in a stored process on UNIX and display the output in the Internet Exporer browser on Windows.

When you use the ACTIVEX device with an ODS destination that does not support the ACTIVEX device, SAS/GRAPH switches to the ACTXIMG device, which generates a PNG image. For example, the ODS PDF statement generates output for the Adobe Reader in a Portable Document Format (PDF) file. This format does not support embedded ActiveX applications. Specifying the ACTIVEX device with the ODS PDF statement generates a PDF output file that contains a static image of the graphics output that is embedded in the PDF file. The ODS RTF destination creates an RTF file that contains a PNG image. The ODS PRINTER destinations use their native format.

The ACTXIMG device can produce an image map in the HTML output file to enable data tips and drill-down functionality from the image. See Chapter 27, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 595.

Installing the SAS/GRAPH ActiveX Control

The SAS/GRAPH ActiveX Control is installed silently when you install SAS/GRAPH. The SAS/GRAPH ActiveX Control can be installed manually, as described in "Manually Installing the SAS/GRAPH ActiveX Control" on page 455. You can configure your presentation to prompt your Web users to go through the installation process, as described in "Configuring Your Program to Prompt Users to Install the SAS/GRAPH ActiveX Control" on page 456 and "Configuring an Existing ActiveX Presentation to Prompt Users to Install the SAS/GRAPH ActiveX Control" on page 456.

Manually Installing the SAS/GRAPH ActiveX Control

Follow these steps to manually install the SAS/GRAPH ActiveX Control.

1. Open the SAS Downloads page in your Web browser:

   `http://www.sas.com/apps/demosdownloads/setupintro.jsp`

2. If you are not already logged in, type in your user name and password, and then click the **`LOG IN`** button.

   Note: You must log in to download files. If you do not have an account, click the **`Sign up now`** link to create an account. △

3. Click the **`Request Download`** button for your Windows platform. This opens the License Agreement page.

4. On the License Agreement page, read the license agreement, and the click the **`I Accept`** button. This opens the Downloads page.

   Note: If you do not want to accept the license agreement, click the **`Do Not Accept`** button to cancel the download. △

5 On the Downloads page, click the **Download** button, and then select a location on your computer for the file. This downloads a ZIP file to your computer.

6 Extract the ZIP file that you downloaded. This extracts file sasgraph.exe.

7 Run the installation program (sasgraph.exe) and follow the installation prompts. The installation program installs the SAS/GRAPH ActiveX Control files in the following folder:

```
C:\Program Files\SAS\SharedFiles\Graph\Vx
```

Where *x* is the version number. Installation requires eight megabytes of disk space.

Note: For 64-bit enabled Windows, the SAS/GRAPH ActiveX Control works only in the 32-bit version of Internet Explorer. △

Configuring Your Program to Prompt Users to Install the SAS/GRAPH ActiveX Control

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/v92/sasgraph.exe#version=9,2"
```

No files are installed without the user's permission. 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.

To be able to access the installation program, Web users must be able to access its storage location. You might need to copy the installation program to another location to ensure availability.

You can 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/sasgraph.exe#version=9,2";
```

If the installation program is not stored 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="/grsrc/sasgraph.exe#version=9,2";
```

Configuring an Existing ActiveX Presentation to Prompt Users to Install the SAS/GRAPH ActiveX Control

You can edit an existing presentation that was generated with the ACTIVEX device so that the presentation prompts your users to install the SAS/GRAPH ActiveX Control if it is not already installed.

Follow these steps to add the installation capability to your ACTIVEX presentation:

1 In a text editor, open the initial HTML file of your Web presentation.

2 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="file://grsrc/sasgraph.exe"
```

If the installation program is stored on a Web server, use an HTTP reference. For example:

```
CODEBASE="http://www.ourco.com/sasweb/graph/sasgraph.exe#version=9,2"
```

3 Save the HTML file and close the editor.

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.

Note: If you want to install the control in a non-default location, you must install the control manually, as described in "Manually Installing the SAS/GRAPH ActiveX Control" on page 455. △

Uninstalling the SAS/GRAPH ActiveX Control

If the SAS/GRAPH ActiveX Control was installed with the SAS/ GRAPH software, you cannot manually uninstall the SAS/GRAPH ActiveX Control separately from the SAS/GRAPH software. In this case, to uninstall the SAS/GRAPH ActiveX Control, you must uninstall the SAS/GRAPH software. If you manually installed the SAS/GRAPH ActiveX Control, you can manually uninstall it.

To manually uninstall the SAS/GRAPH ActiveX Control on Windows XP:

1 Open the Control Panel window by selecting **Start ▸ Settings ▸ Control Panel**.
2 Double-click **`Add or Remove Programs`**.
3 Select **`SAS Graph ActiveX Control`**.
4 Click **`Remove`**.

To manually uninstall the SAS/GRAPH ActiveX Control on Windows Vista:

1 Open the Control Panel window by selecting **Start ▸ Control Panel**.
2 In the Control Panel, under Programs, click **`Uninstall a program`**. A list of the installed programs is displayed.
3 In the program list, select **`SAS Graph ActiveX Control`**, and then click **`Uninstall`**.

Generating Output for ActiveX

The SAS/GRAPH ActiveX Control displays interactive charts, maps, and plots. The following table lists the various ways that you can deliver ActiveX output to your audience.

Table 17.2 Primary Delivery Choices for SAS/GRAPH ActiveX Control Output

Application	ODS Statement	Output File
Internet Explorer	ODS HTML	HTML
Microsoft Word	ODS RTF	Rich text format
Adobe Acrobat Reader	ODS PDF	Portable document format
Ghostview, and so on	ODS PS	PostScript format

Note: These choices also apply to the JAVA device. △

Table 17.1 on page 454 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 To conserve resources, close the ODS LISTING destination:

```
ods listing close;
```

3 Open an ODS destination that is listed in Table 17.2 on page 457. Use the STYLE= option to specify an ODS style (see Chapter 10, "Controlling The Appearance of Your Graphs," on page 133), and use the PATH= and BODY= options to specify an output filename other than the default. For example:

```
ods html path="C:\" body="your_file.htm"
    style="banker";
```

4 Run a procedure or procedures that are supported by the ACTIVEX device (see Table 17.1 on page 454):

```
proc gchart data=sashelp.class;
   vbar height / group=age;
run;
quit;
```

5 Close the ODS destination that you opened in step 3, and then reopen the ODS LISTING destination. For example:

```
ods html close;
ods listing;
```

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 the SAS/GRAPH ActiveX Control. For information on prompting new users to start the SAS/GRAPH ActiveX Control installation process, see "Configuring Your Program to Prompt Users to Install the SAS/GRAPH ActiveX Control" on page 456. For further troubleshooting information, see "Troubleshooting Web Output" on page 633. For information on enhancing the default Web presentation, see "Configuring Drill-Down Links with ACTIVEX" on page 460.

About Languages in ACTIVEX

For international audiences, the SAS/GRAPH ActiveX Control has a graphical user interface that can appear in the following languages: Chinese (simplified), Danish, English, French, German, Hebrew, Hungarian, Italian, Japanese, Korean, Polish, Russian, and Spanish. To display a translated graphical user interface, in general, Web-based ActiveX devices must use a language-specific operating environment and Web browser. For further 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. In the LABEL and GPLOT SYMBOL statement, the ACTIVEX device supports the following SAS markers:

Marker

Square

Star

Circle

Plus

Flag

X

Prism

Spade

Heart

Diamond

Club

Hexagon

Cylinder

See "SYMBOL Statement" on page 252 for more information.

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 Language Reference: Dictionary*.

Table 17.3 Character Formats Supported by ActiveX

$	$ASCII	$BINARY	$BYVAL	$CHAR
$EBCDIC	$HEX	$OCTAL	$QUOTE	$REVERJ
$REVERS	$UPCASE	$XPORTCH		

Table 17.4 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				

Table 17.5 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. If you create a custom format to use with these devices, do not nest existing formats in your new format. △

Configuring Drill-Down Links with ACTIVEX

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 “Parameter Reference for Java and ActiveX” on page 488.

In the ODS HTML statement, ActiveX parameters are specified with the PARAMETERS= option, as described in “Specifying Parameters and Attributes for Java and ActiveX” on page 485.

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. These drill-down modes are implemented in ActiveX in the same way that they are implemented in Java. For information on implementing these drill-down modes, see Chapter 27, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 595.

Note: You can convert the Java examples to ActiveX by changing the DEVICE=JAVA graphics option in the GOPTIONS statement to DEVICE=ACTIVEX. △

The following table lists the procedures and statements that generate output that can be used in ActiveX presentations with drill-down functionality.

Table 17.6 Statements Enabled for Drill-Down Functionality in ActiveX

Procedure	Statements
GBARLINE	BAR, PLOT
GCHART	HBAR, HBAR3D, VBAR, VBAR3D, PIE, PIE3D, DONUT
GPLOT	PLOT, BUBBLE, BUBBLE2, PLOT2
GMAP	CHORO, BLOCK, PRISM
G3D	PLOT, SCATTER

ActiveX Examples

The following sections provide examples of how to create interactive graphs using the ACTIVEX device:

"Generating an ActiveX Graph for a Microsoft Word Document" on page 461

"Generating an Interactive Contour Plot in ActiveX" on page 463

"Providing JavaScript Drill-Down with ActiveX" on page 464

"Providing More JavaScript Drill-Down with ActiveX" on page 466

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

Generating an ActiveX Graph for a Microsoft Word Document

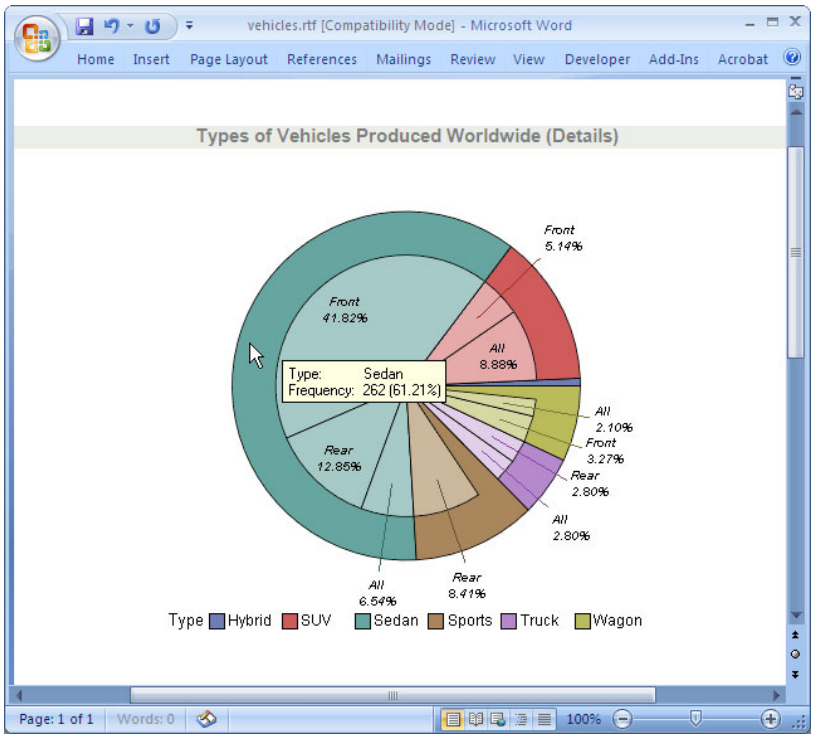
Here is an example that demonstrates how the ODS RTF statement can be combined with the ACTIVEX device to generate interactive graphs inside Microsoft Word files.

The sample program is as follows:

```
goptions reset=all device=activex;
ods listing close;
ods rtf path="C:\" file="vehicles.rtf" style=statistical;
title "Types of Vehicles Produced Worldwide (Details)";

proc gchart data=sashelp.cars;
```

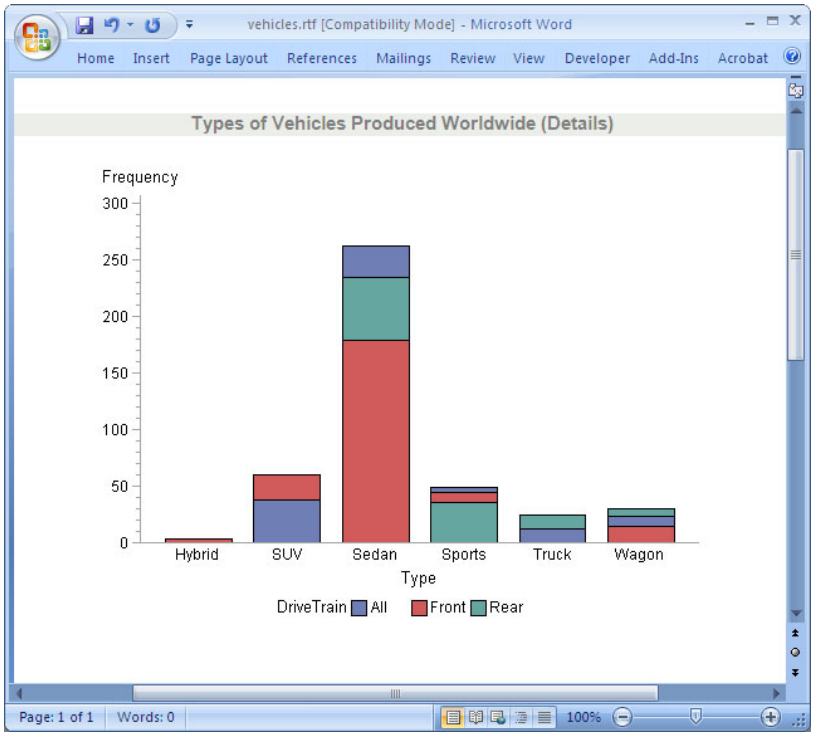
```
pie type / detail=drivetrain
    detail_percent=best
    detail_value=none
    detail_slice=best
    detail_threshold=2
    legend;
run;
quit;
ods rtf close;
ods listing;
```

The following display shows the resulting file opened in Microsoft Word.



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 **ChartType ▸ VerticalBar** in the pop-up menu. The chart changes from a pie chart to a vertical bar chart.

Note: The SAS/GRAPH ActiveX pop-up menu does not display if the SAS/GRAPH ActiveX Control is in the design mode in Microsoft Word. If the ActiveX object is in the design mode, in Microsoft Word, click the Exit Design Mode icon in the Control Toolbox. △



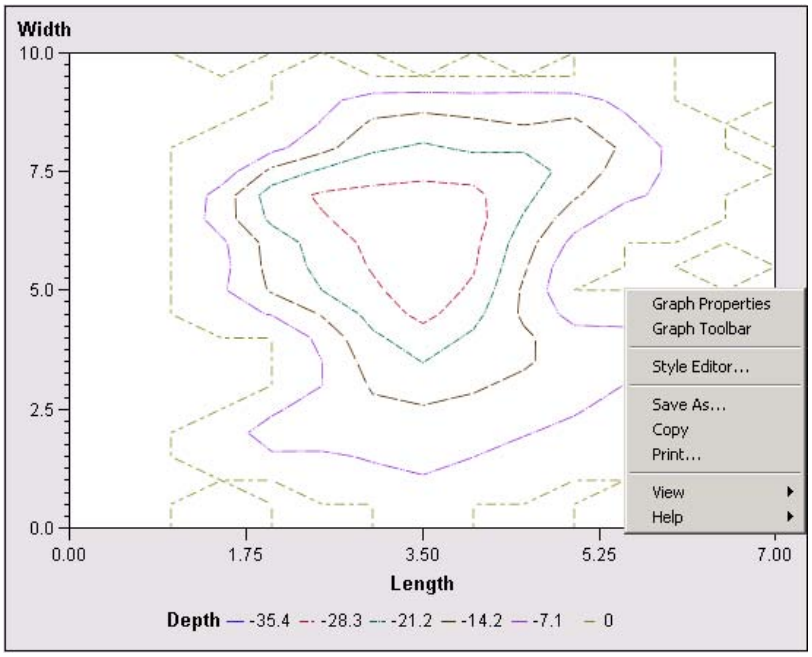
Generating an Interactive Contour Plot in ActiveX

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.

The sample program is as follows:

```
goptions reset=all border device=activex;
ods listing close;
ods html style=default;
proc gcontour data=sashelp.lake;
    plot width * length = depth;
run;
quit;
ods html close;
ods listing;
```

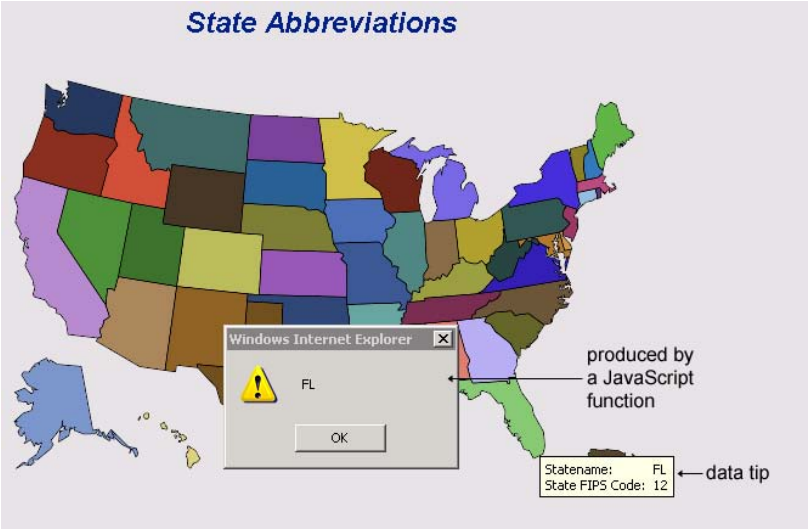
The following display shows the result.



Providing JavaScript Drill-Down with ActiveX

Here is an example that shows you how to implement the Script drill-down mode using the MAP procedure and the ACTIVEX device. By default, SAS/GRAPH provides data tips for graphs that are generated with the ACTIVEX device. These data tips are displayed when the cursor is over a portion of the map. To implement JavaScript drill-down functionality, PUT statements are used to insert JavaScript code into the HTML file. The JavaScript, in the example, opens an alert window that displays the state abbreviation.

This example is available in the Sample Library under the name GWBDRACT.



The sample program is as follows:

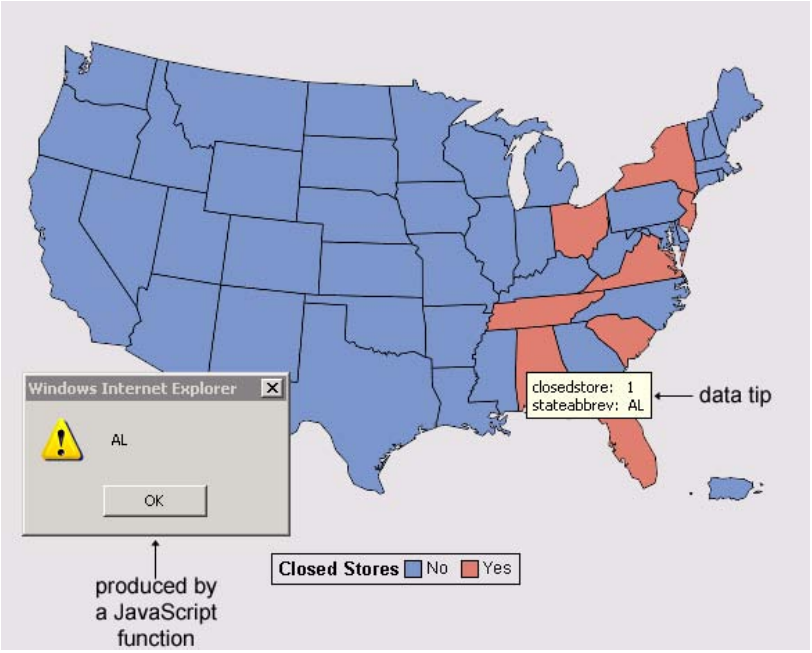
```
 /* Change the following line to specify your output file. */
filename odsout "states.htm" ;

/* If your site has already installed the map data sets and      */
/* defined the MAPS libref, then you can delete the LIBNAME      */
/* statement below and the sample code should work.                */
/* If not, contact your on-site SAS support personnel                  */
/* to determine how to define the MAPS libref.                   */
*libname maps 'SAS-MAPS-library';
/* Create a data set that contains the US states. */
proc sql;
create table work.mydata as
select unique state from maps.us;
quit;

/* Add state abbreviations to the new data set. */
data work.mydata;
 length Statename $2;
 set work.mydata;
 Statename=trim(left(upcase(fipstate(state))));
run;

/* Specify the ACTIVEX device. */
goptions reset=all device=activex;

/* Specify the HTML output file, the Script   */
/* drill-down mode, and the callback method.  */
/* Close ODS LISTING to conserve resources. */

ods listing close;
ods html file=odsout
     style=default
     parameters=("DRILLDOWNMODE"="Script"
                 "EXTERNALNAME"="GIDX"
                 "DRILLTARGET"="_self"
                 "DRILLFUNC"="MapDrill")
     attributes=("NAME"="GIDX");

/* Specify a map title and generate the map. */
title "State Abbreviations";
proc gmap map=maps.us data=work.mydata all;
   id state;
   choro statename / nolegend;
run;
quit;

/* Close the HTML destination and  */
/* open the listing destination.   */
ods html close;
ods listing;

/* Create the MapDrill script that is specified on */
/* the ODS HTML statement's DRILLFUNC parameter.   */
/* Write the script to the same file that contains */
```

```
/* the HTML output from the GMAP procedure.        */
data _null_ ;
file odsout mod; /* modify rather than replace file */
  put " " ;
  put "<SCRIPT LANGUAGE='JavaScript'>" ;
  put "function MapDrill( appletref )" ;
  put "{" ;
  put " " ;
  put "/* Open an alert box to show the abbreviated state name. */" ;
  put "for(i = 2; i < MapDrill.arguments.length; i += 2 )" ;
  put "  {" ;
  put "    if (MapDrill.arguments[i] == 'G_DEPV,f' ) " ;
  put "        alert(MapDrill.arguments[i+1]);" ;
  put "  }" ;
  put " " ;
  put "}" ;
  put "</SCRIPT>";
run ;
```

Providing More JavaScript Drill-Down with ActiveX

Here is an example that is similar to the example shown in "Providing JavaScript Drill-Down with ActiveX" on page 464 but involves slightly more JavaScript coding. The program generates a map of the United States showing the states in which stores have been closed. If you click on a state in which no stores have been closed, then no action is performed. If you click on a state in which stores have been closed, a JavaScript alert window is displayed that shows the state abbreviation. In a real application, the JavaScript could be modified to display a list of the closed stores. This example is available in the Sample Library under the name GWBDRAC2.

For additional information on the script drill-down mode, see "Controlling Drill-Down Behavior For ActiveX and Java Using Parameters" on page 608.



The sample program is as follows:

```
 /* Change the following line to specify your output file. */
filename odsout "stores.htm" ;

goptions reset=all device=activex;

data stores;
length stateabbrev $ 2;
input state closedstore stateabbrev $ @@;
datalines;
 1  1  AL  2  0  AK  3  0  --  4  0  AZ  5  0  AR  6  0  CA
 7  0  --  8  0  CO  9  0  CT 10  0  DE 11  0  DC 12  1  FL
13  0  GA 14  0  -- 15  0  HI 16  0  ID 17  0  IL 18  0  IN
19  0  IA 20  0  KS 21  0  KY 22  0  LA 23  0  ME 24  0  MD
25  0  MA 26  0  MI 27  0  MN 28  0  MS 29  0  MO 30  0  MT
31  0  NE 32  0  NV 33  0  NH 34  1  NJ 35  0  NM 36  1  NY
37  0  NC 38  0  ND 39  1  OH 39  1  OH 40  0  OK 41  0  OR
42  0  PA 43  0  -- 44  0  RI 45  1  SC 46  0  SD 47  1  TN
48  0  TX 49  0  UT 50  0  VT 51  1  VA 52  0  -- 53  0  WA
54  0  WV 55  0  WI 56  0  WY 57  0  -- 58  0  -- 59  0  --
60  0  AS 61  0  PQ 62  0  EQ 63  0  -- 64  0  FM 65  0  --
66  0  GU 67  0  JQ 68  0  MH 69  0  MP 70  0  PW 71  0  MQ
72  0  PR
;
run;

/* create own custom maps data set where id is 2--letter state
abbreviation(statecode) not state fips number(state) */
data cus_map;
   length stateabbrev $2;
   set maps.us;
   stateabbrev=fipstate(state);
run;

ods listing close;
ods html body=odsout nogtitle
      style=default
      parameters=("DRILLDOWNMODE"="Script"
                        "EXTERNALNAME"="GIDX"
                        "DRILLTARGET"="_self"
                        "DRILLFUNC"="MapDrill")
      attributes=("NAME"="GIDX");

legend1 label=("Closed Stores")
             value=(t=1 j=l "No" t=2 j=l "Yes") frame;

proc gmap map=cus_map(where=(state ^in(2, 15))) data=stores;
     id stateabbrev;
     choro closedstore/discrete missing
     legend=legend1;
     run;
quit;

ods html close;
```

```
ods listing;

data _null_ ;

  file odsout mod;

  put "<SCRIPT LANGUAGE='JavaScript'>" ;
  put " var isclosed = null; ";
  put " var newWin; ";
  put " var stateabbrev; ";
  put " stateabbrev = ''; ";

  put "function MapDrill( appletref )" ;
  put "{" ;
  put " " ;
  put "/* Open an alert box to show the abbreviated state name. */";
  put "  for(i = 2; i < MapDrill.arguments.length; i += 2 )";
  put "  { ";
  put "    if (MapDrill.arguments[i] == 'G_DEPV,f' )
             {isclosed=MapDrill.arguments[i+1]; }";
  put "    if (MapDrill.arguments[i] == 'G_LABELV,f')
             {stateabbrev =MapDrill.arguments[i+1] + ' '; }";
  put "  } ";
  put "  if (isclosed == 1.000000){ alert(stateabbrev) }";
  put "}";

  put "</SCRIPT>";
run;
```

CHAPTER

18

Creating Interactive Output for Java

Overview

The JAVA device generates interactive presentations that run in the Graph, Map, Tile, and Contour applets. These applets can display the output of certain SAS/GRAPH procedures as follows:

Graph applet	G3D scatter plots, GCHART, GPLOT
Map applet	GMAP
Tile applet	GTILE
Contour applet	G3D surface plots, GCONTOUR

The Java applets enable Web users to display data tips, to change the graph type, to pan, rotate, and zoom, and to change colors, fonts, axes, legends, and variable roles.

Note: The Java applets do not support the GAREABAR, GBARLINE, or GRADAR procedures. To provide interactivity with the output of these procedures, use the ACTIVEX device instead, as described in Chapter 17, "Creating Interactive Output for ActiveX," on page 453. ActiveX output can also appear in Microsoft Word documents or other OLE applications. △

You can enhance JAVA-device-generated graphs by setting applet parameters and specifying Output Delivery System (ODS) options. Applet parameters let you configure drill-down links and override default values in the user interface. Information on parameters is provided in Chapter 19, "Attributes and Parameters for Java and ActiveX," on page 485.

You can use ODS styles to enhance the appearance of JAVA-device-generated charts, as described in Chapter 10, "Controlling The Appearance of Your Graphs," on page 133.

To generate a Web presentation that runs the Graph, Map, or Contour applet, you generally specify the JAVA device in a GOPTIONS statement, open the HTML destination, generate one or more graphs, and then close the HTML destination, as described in "Generating Output for Java" on page 470.

You can generate the same graphs as static images using the DEVICE=JAVAIMG graphics option. Static images can be displayed without requiring that the Web user install the applets or Java Runtime Environment (JRE). For details, see "ACTXIMG and JAVAIMG Devices" on page 506.

You can also use the JAVAMETA device to create interactive metagraphics output. See Chapter 22, "Generating Interactive Metagraphics Output," on page 531.

When to Use the JAVA Device

The JAVA device generates output for the Graph, Map, Tile, and Contour applets. These applets provide user interactivity in all of the supported Web browsers. If you do not need interactivity, then use the JAVAIMG device, as described in "Developing Web Presentations with the JAVAIMG and ACTXIMG Devices" on page 510, or use the PNG device, as described in "Developing Web Presentations with the GIF, JPEG, SVG, and PNG Devices" on page 508.

Generating Output for Java

To develop a SAS/GRAPH program that generates output for the Graph applet or Map applet, follow these steps:

1 Reset graphics options and specify the JAVA device:

```
goptions reset=all device=java;
```

2 To conserve resources, close the ODS LISTING destination:

```
ods listing close;
```

3 Open the ODS HTML destination. You can use the BODY= option to specify an HTML filename, and the STYLE= option to specify an ODS style (see Chapter 10, "Controlling The Appearance of Your Graphs," on page 133). Use the PARAMETERS= option to configure the applet (see "Specifying Parameters and Attributes for Java and ActiveX" on page 485). For example:

```
ods html
   file="your_file.htm"
   style=gears
   parameters=("tips"="none");
```

Note: To run an applet, your users must be able access the appropriate Java archive files. Two archives are referenced by default: one is the Java plug-in from Sun Microsystems, and the other is the SAS Java archive. △

In the HTML output file, the location of the Java plug-in from Sun Microsystems is specified in the CODEBASE attribute of the OBJECT tag. If you need to change this default value, then use the ATTRIBUTES= option of the ODS statement, as described in "Specifying Parameters and Attributes for Java and ActiveX" on page 485. On Windows systems, the user is prompted to install the plug-in if it is not already installed. On other systems, the plug-in can be installed

from the Sun Microsystems site (**http://www.sun.com**) or from the SAS Third-Party Software References Web page:

http://support.sas.com/resources/thirdpartysupport/index.html

The location of the SAS Java archive is specified in the JAVA_CODEBASE and the ARCHIVE parameters in the body of the APPLET tag. The default JAVA_CODEBASE is specified by the APPLETLOC= system option. If the default value of this system option specifies a widely accessible URL, then you do not need to change this value. If you need to specify a different location, then you can change the value of the system option. Another alternative is to override the APPLETLOC= system option by specifying a value for the ODS statement option CODEBASE=, as described in "Specifying Parameters and Attributes for Java and ActiveX" on page 485.

Note: When specifying a location for the SAS Java archive, you can use an HTTP address, or you can use a UNC path, such as **//sasjava**, with forward slashes instead of backward slashes. △

4 Run a procedure or procedures that are used by the JAVA device (see Table 17.1 on page 454):

```
proc gchart data=sashelp.class;
   vbar height / group=age;
run;
quit;
```

5 Close the ODS HTML destination, and then reopen the ODS LISTING destination:

```
ods html close;
ods listing;
```

Running your program starts the applet and displays the initial graph. If the browser display differs from what you see in SAS, then ensure that your SAS/GRAPH procedure is fully enabled in the applet. Refer to Appendix 1, "Summary of ActiveX and Java Support," on page 1601 for details.

Note: Using the GMAP procedure to generate a highly detailed map might create a large HTML output file, which might cause problems on certain browsers. If this is the case, you can run the GREDUCE procedure to remove some of the complexity and produce a more usable map. △

For further information on troubleshooting Web output, see "Resolving Differences Between Graphs Generated with Different Technologies" on page 638.

About the Java HTML Output and the Java Runtime Environment Plug-In

The Java Runtime Environment (JRE) plug-in is required to open HTML output that is generated by the JAVA device. If you open an HTML file that you generated using the JAVA device and you do not have the JRE plug-in installed for your Web browser, the browser prompts you to install the JRE plug-in. You must install the JRE plug-in for your browser in this case. When your users open your HTML file, they will also have to install the JRE plug-in for their Web browser if the plug-in is not already installed on their computer.

The 9.2 SAS/GRAPH Java applets will work with JRE 1.5.0_12. They have also been tested with 1.5.0_13 and 1.5.0_15. We recommended that you use one of these versions. If future JREs are backward compatible, then the applets should work without any issues.

About Languages in JAVA

For international audiences, the Java applets have graphical user interfaces that can appear in the following languages: Chinese (simplified), Czech, Danish, English, French, German, Hebrew, Hungarian, Italian, Japanese, Korean, Norwegian, Polish, Russian, Spanish, and Swedish. Generally, to display a translated graphical user interface, Web-based JAVA devices must use a language-specific operating environment and Web browser. This requires the all-languages version of the JRE. For further information, contact your on-site SAS support personnel.

About Special Fonts and Symbols in JAVA

The JAVA device supports only system fonts. In the LABEL and GPLOT SYMBOL statement, the ACTIVEX device supports the following SAS markers:

Marker

Square

Star

Circle

Plus

Flag

X

Prism

Spade

Heart

Diamond

Club

Hexagon

Cylinder

See “SYMBOL Statement” on page 252 for more information.

SAS Formats Supported for Java

The JAVA devices support 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 Language Reference: Dictionary*.

Table 18.1 Character Formats Supported By Java

$	$ASCII	$BINARY	$CHAR
$F	$HEX	$OCTAL	

Table 18.2 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 18.3 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	NLDATMW
NLDATMWN	NLDATMYM	NLDATMYQ	NLDATMYR	NLDATMYW
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 JAVA and JAVAIMG devices do not support nested formats. If you create a custom format to use with these devices, do not nest existing formats in your new format. △

Configuring Drill-Down Links for Java

You can configure your Java applet to add drill-down links to your graph in one of the following modes:

- Local mode
- Script mode
- URL mode

See Chapter 27, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 595. See also "Examples of Interactive Java Output" on page 475.

Examples of Interactive Java Output

The following sections provide examples of creating interactive graphs using the JAVA device:

Additional samples are available in the Sample Library:

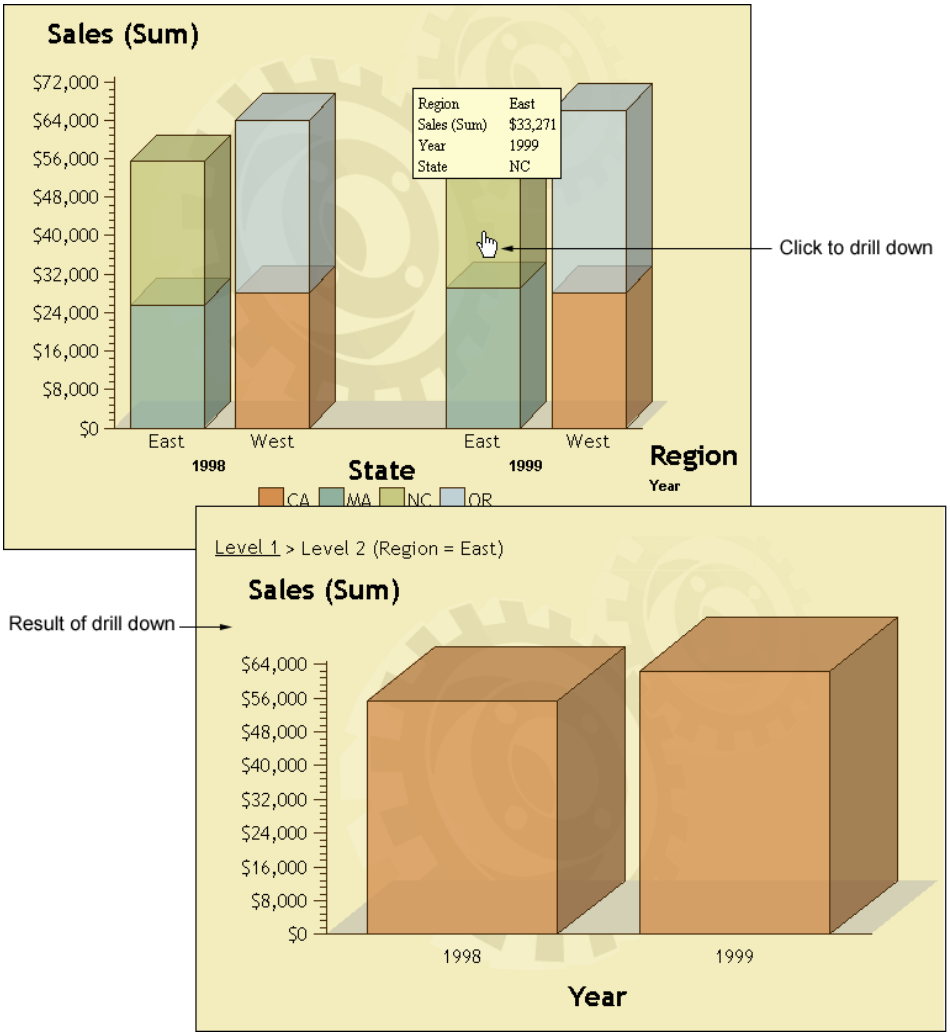
- GWBJABAR—Generating a Bar Chart for the Web
- GWBJACON—Generating a Contour Plot for the Web

Local Drill-Down Mode with Java

Here is an example that generates an HTML output file that runs the Graph applet. If the graph contains a group or subgroup, then by default the applet automatically provides drill-down functionality. When a user clicks on an element in the graph, the applet generates and displays a new graphic based on the selected elements. In the example, note how variable roles are assigned in the VBAR3D statement.

This example is available in the Sample Library under the name GWBJALOC. For further information, see "Links in ACTIVEX Presentations" on page 605.

The following picture shows output produced by the sample program. The top of the picture shows the initial graph. The bottom of the picture shows the graph that results from a user clicking on a portion of the initial graph.



Here is the sample program:

```
filename odsout "sales.htm";

/* Close the listing destination. */
ods listing close;

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
```

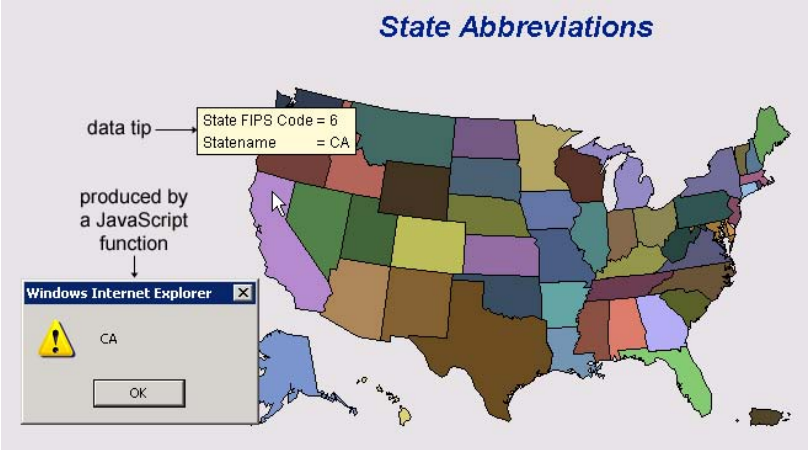
```
East NC 13611 1999 1
East MA 11084 1999 2
East NC 19660 1999 2
West CA 12536 1998 1
West OR 17888 1998 1
West CA 15623 1998 2
West OR 17963 1998 2
East NC 17638 1998 1
East MA 12811 1998 1
East NC 12184 1998 2
East MA 12760 1998 2
;
goptions reset=all device=java;

ods html file=odsout style=gears;

title "Company Sales, Mid Year";

proc gchart data=sales;
   vbar3d region / sumvar=sales
   group=year subgroup=state;
run; quit;

ods html close;
ods listing;
```

You can also use the HTML= procedure option to implement local drill-down links in your graphs. See “Adding Links with the HTML= and HTML_LEGEND= Options” on page 601.

Script Drill-Down Mode with Java

Here is an example that shows how to implement the script drill-down mode in the Graph applet or Map applet. SAS/GRAPH provides data tips by default. These data tips are displayed when the cursor is over a portion of the map. To implement JavaScript drill-down functionality, PUT statements are used to insert JavaScript code into the HTML file. The JavaScript, in the example, opens an alert window that displays the state abbreviation.

This example is available in the Sample Library under the name GWBSCDRL. For further information, see “Links in ACTIVEX Presentations” on page 605.



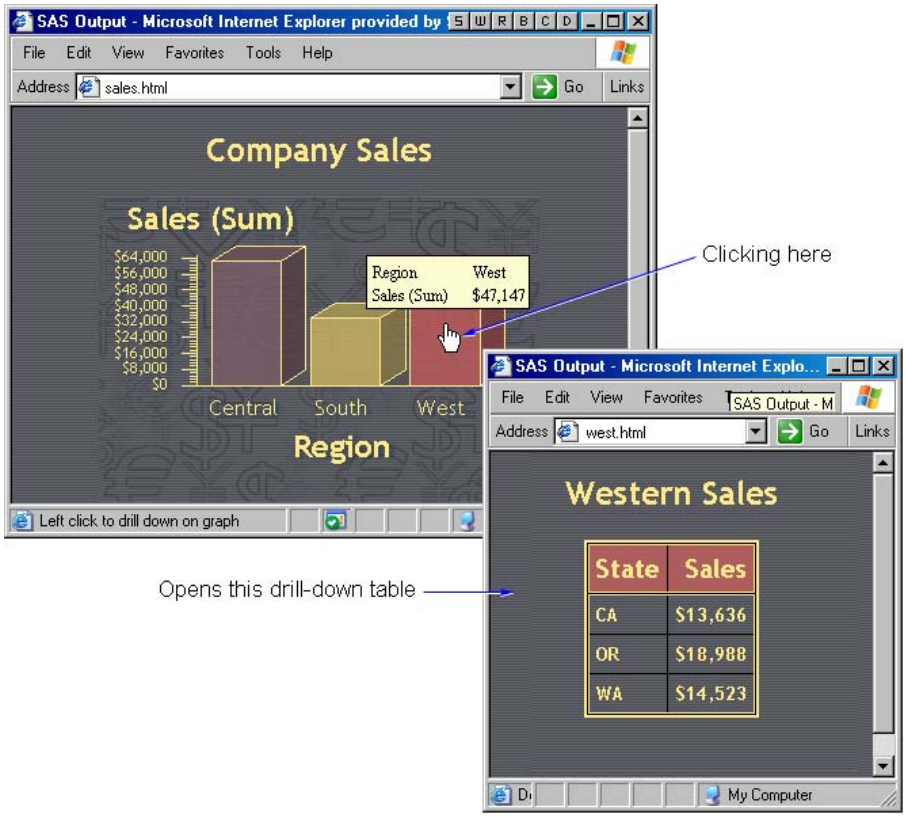

```
/* Change the next two lines to run this program. */
filename odsout "states.htm" ;

/* If your site has already installed the map data sets and           */
/* defined the MAPS libref, then you can delete the LIBNAME statement */
/* below and the sample code will work.        */
/* If not, contact your on-site SAS support personnel */
/* to determine how to define the MAPS libref. */
*libname maps 'SAS-MAPS-library';

/* Create a data set that contains the US states. */
proc sql;
create table work.mydata as
select unique state from maps.us;
quit;

/* Add state abbreviations to the new data set. */
data work.mydata;
 length Statename $2;
 set work.mydata;
 Statename=trim(left(upcase(fipstate(state))));
run;

/* Specify the JAVA device. */
goptions reset=all device=java;

/* Close the LISTING destination to save       */
/* system resources.                                          */
/* Specify the HTML output file, the script  */
/* drill-down mode, and the callback method. */
ods listing close;
ods html file=odsout style=default
         parameters=("DRILLDOWNMODE"="Script"
                     "DRILLFUNC"="MapDrill");

/* Specify a map title and generate the map. */
title1 "State Abbreviations";
```

```
proc gmap map=maps.us data=work.mydata all;
   id state;
   choro statename / nolegend;
run;
quit;

/* Close the HTML destination and */
/* open the listing destination.  */
ods html close;
ods listing;

/* Create the MapDrill script that is specified on */
/* the ODS HTML statement's DRILLFUNC parameter.   */
/* Write the script to the same file that contains */
/* the HTML output from the GMAP procedure.        */
data _null_ ;
file odsout mod; /* Modify the file rather than replacing it. */
put " " ;
put "<SCRIPT LANGUAGE='JavaScript'>" ;
put "function MapDrill( appletref )" ;
put "{" ;
put " " ;
put "/* Open an alert box to show the abbreviated state name. */ " ;
put "for(i = 2; i < MapDrill.arguments.length; i += 2)" ;
put "  {" ;
put "    if (MapDrill.arguments[i] == 'G_DEPV,f' )";
put "        alert(MapDrill.arguments[i+1]);" ;
put "  }" ;
put " " ;
put "}" ;
put "</SCRIPT>";
run ;
```

URL Drill-Down Mode with Java

Here is an example that demonstrates the URL drill-down mode. This example is available in the SAS sample library under the name GWBURLDR. For further information, see "Links in ACTIVEX Presentations" on page 605.

The following display shows the output of the sample code. The resulting sales.html file displays a bar chart. Clicking on any one of the bars opens the corresponding HTML file that displays a table further breaking down the data.




```
/* Change web-output-path in the following statements */
filename urldrill "web-output-path";
filename sales "web-output-path/sales.html";
filename central "web-output-path/central.html";
filename south "web-output-path/south.html";
filename west "web-output-path/west.html";

/* Close the ODS listing destination to conserve resources. */
ods listing close;

/* Specify the device. */
goptions reset=all device=java;

/* 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 link variable. */
if Region="Central" then
     rpt="href='central.html'";
   else if Region="South" then
```

```
     rpt="href='south.html'";
   else if Region="West" then
     rpt="href='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
;

/* Open the HTML output file and specify the URL drill-down mode. */
ods html body=sales
    path=urldrill
    style=money
    parameters=("drilldownmode"="url");

/* Create a chart that uses the link variable. */
title "Company Sales";
proc gchart data=regsales;
   vbar3d region / sumvar=sales
   patternid=midpoint
   html=rpt;
run;
quit;

/* Create an HTML file for Central sales. */
ods html body=central path=urldrill style=money;
title "Central Sales";
proc print data=regsales noobs;
   var state sales;
   where region="Central";
run;

/* Create an HTML file for Southern sales */
ods html body=south path=urldrill style=money;
title "Southern Sales";
proc print data=regsales noobs;
   var state sales;
   where region="South";
run;

/* Create an HTML file for Western sales. */
ods html body=west  path=urldrill style=money;
title1 "Western Sales";
proc print data=regsales noobs;
   var state sales;
   where region="West";
run;
```

```
quit;

/* Close the HTML destination and open the listing destination. */
ods html close;
ods listing;
```

HTML Drill-Down Mode

Here is an example that generates an HTML output file that displays the Map applet. The applet is configured for the HTML drill-down mode, where URLs are dynamically generated based on the data in the graph element that was selected in the drill-down action. In this example, the value of the STATENAME variable is used to complete the URLs. For additional information, see “Links in ACTIVEX Presentations” on page 605.

In the resulting HTML page, clicking on a state in the U.S. map activates a URL. This sample is available in the SAS Sample Library under the name GWBJAMAP.

```
/* Close the listing destination to conserve resources.     */

ods listing close;

/* Specify a path and name for the HTML output file.  */

 ods html
   file="your_HTML_file.htm"
    style=default
    parameters=("DRILLDOWNMODE"="HTML")
    parameters=("DRILLPATTERN"="http://www.state.{&statename}.us")
    parameters=("BACKCOLOR"="FFFFFF");

/* Specify the JAVA device and set up customizations.       */

goptions reset=all device=java;

/* Create data for the graph. */

proc sql;
   create table work.mydata as
   select unique state from maps.us;
   quit;
   run;
data work.mydata;
   length statename $1020;
   set work.mydata;

/* Place the state name in the data set. */

   statename=trim(left(lowcase(fipstate(state))));
   run;

title1
   "Click on a state to go to that state's home page";

/* Generate the graph. */
```

```
proc gmap map=maps.us
   data=work.mydata all;
   id state;
   choro statename / levels=1 discrete
         coutline=black
         nolegend
         des="US Government Web Sites"
         name="usgov";
    run; quit;

/* Close the HTML output file and open the listing  */
/* destination.                                     */

ods html close;
ods listing;
```

CHAPTER

19

Attributes and Parameters for Java and ActiveX

Specifying Parameters and Attributes for Java and ActiveX

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 in most cases.

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 a Java applet or 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 applet or 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.

Note: Using **,n** is not the same as using the applet parameter PATTERNSTRIP. The PATTERNSTRIP parameter removes blank spaces from data values before those values are applied to substitution strings. △

Most of the examples in the following topics specify parameters:

- "Examples of Interactive Java Output" on page 475
- "ActiveX Examples" on page 461

For information on other ODS statement options, see the *SAS Output Delivery System: User's Guide*.

In HTML output that runs an applet or a 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 JAVA or ACTIVEX device drivers that run 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 tooltips 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, Graph applet, Map applet, and Contour applet are defined in "Parameter Reference for Java and ActiveX" on page 488. Parameters for other applets, such as the Metaview applet, are provided in the sections that apply to those applets, as in "Metaview Applet Parameters" on page 534.

Specifying the Location of Control and Applet Files (CODEBASE= and ARCHIVE= Options)

When you generate Web presentations with the JAVA and ACTIVEX device drivers, the SAS/GRAPH software generates HTML pages that automatically look for the Java archive files or the ActiveX control file in the default installation location. If you install the ActiveX control **.exe** file or the Java archive **.jar** files in a location other than the default or if you want to publish Output Delivery System (ODS) output containing the SAS/GRAPH control or the applets in a Web server, then you might need to specify the location of the **.exe** file or the **.jar** files when you generate your Web presentation.

You can use the CODEBASE= option to specify the location of the ActiveX control or the Java applets. You can use the ARCHIVE= option to specify the name of the Java archive file.

Note: The ActiveX control must be installed locally on each PC where the Web presentation will be 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 filename of the **.exe** file. (The default filename is **sasgraph.exe.**) The CODEBASE location can be specified as a pathname or as a URL. (See "Specifying the CODEBASE= URL" on page 487 for more information.) If you have installed previous versions of the ActiveX control, then you also need to specify the version that you want to use. For example, if your **.exe** file is in **/sasweb/graph**) you would specify

```
ods html file="/path/to/mygraph.html"
      codebase="/sasweb/graph/sasgraph.exe#version=9,2";
```

Specifying the Location of the Java Applets

By default, the location of the SAS Java archive files is specified by the APPLETLOC= system option. This value is the default value of the CODEBASE= parameter. If the default location is accessible by users who will be viewing your Web presentation, and the SAS Java archive is installed at that location, then you do not need to change the value of the CODEBASE= parameter.

If you use the JAVA device driver to generate output containing a SAS/GRAPH applet, then specify the path to the **.jar** file with the CODEBASE= option in the ODS statement. Specify only the directory of the **.jar** file. The CODEBASE location can be specified as a pathname or as a URL. (See "Specifying the CODEBASE= URL" on page 487 for more information.) For example, if your **.jar** file is in **/sasweb/graph**), you would specify

```
ods html body="/path/to/mygraph.html"
             codebase="/sasweb/graph";
```

The ARCHIVE= option specifies the filename of the **.jar** file(s). You do not need to specify the ARCHIVE= option in the ODS statement unless you have renamed the **.jar** files.

For applets generated with macros, specify the CODEBASE= argument for the macro. For example:

```
%ds2const(codebase=http://your_path_to_archive, htmlfile=your_path_and_filename.htm
     ...
);
```

For the DS2TREE and DS2CONST macros, you do not need to specify the ARCHIVE= argument unless you have renamed the **.jar** files.

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 control or the applets are installed in a common location, 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 the control or the applets are installed on that server). If you are creating HTML that will be viewed from an e-mail or copied to a Web server on which the applets are not installed, then you should use a full URL to point to the applet **.jar** files at a known location.

Specifying the Location of the Java Plug-In (CODEBASE= Attribute)

The CODEBASE= attribute in the ODS statement specifies the location of the Java plug-in from Sun Microsystems. By default, SAS points to the Web site of the Java plug-in from Sun Microsystems. If necessary, you can change the location of the Java plug-in by specifying the CODEBASE= attribute in the ODS statement. For example:

```
ods html file="c:\myfile.htm"
   attributes=("codebase"="http://ourco.com/Plugins/j2re--1_4_1--windows-i586.exe");
```

On Windows systems, the user is prompted to install the plug-in if it is not already installed. On other systems, the plug-in can be installed from the Sun Microsystems site (**http://www.sun.com**) or from the SAS Third-Party Software References Web page: **http://support.sas.com/resources/thirdpartysupport/index.html**.

Parameter Reference for Java and ActiveX

The following table lists the parameters that you can specify in programs that use the JAVA and ACTIVEX device drivers. Output from the JAVA device driver runs in the Graph applet, Map applet, or Contour applet. Output from the ACTIVEX device driver runs in theSAS/GRAPH Control for ActiveX.

For information on parameters for other applets, see the sections that apply to those applets, such as “Metaview Applet Parameters” on page 534.

Parameter definitions appear after the following table.

Note: These parameters are not supported by the JAVAIMG and ACTXIMG device drivers. △

Table 19.1 Parameters Enabled for Java and ActiveX

Parameter	ActiveX	Graph Applet	Map Applet	Contour Applet
AMBIENT on page 491	x			x
BACKDROPCOLOR on page 491		x		x
BACKIMAGE on page 491	x*	x	x	x
CLIPTIPS on page 491				x
COLORNAMELIST on page 491				x
COLORNAMES on page 491		x		x
COLORSCHEME on page 491	x	x		
DDLEVELn on page 492	x	x		
DIRECT on page 492	x			x
DRAWIMAGE on page 492		x	x	x
DRAWMISSING on page 492				x

Parameter	ActiveX	Graph Applet	Map Applet	Contour Applet
DRAWSIDES on page 492				x
DRILLDOWNFUNCTION on page 492	x	x	x	
DRILLDOWNMODE on page 492	x	x	x	
DRILLPATTERN on page 493	x	x	x	
DRILLTARGET on page 493	x	x	x	
DUPLICATEVALUES on page 493				x
FILLPOLYGONEDGES on page 494				x
FREQNAME on page 494		x		
G_COLOR on page 494	x	x	x	
G_COLORV on page 494	x	x	x	
G_DEP on page 494	x	x	x	
G_DEPTH on page 494	x	x	x	
G_DEPTHV on page 494	x	x	x	
G_DEPV on page 494	x	x	x	
G_GROUP on page 495	x	x	x	
G_GROUPV on page 495	x	x	x	
G_INDEP on page 495	x	x	x	
G_INDEPV on page 495	x	x	x	
G_LABEL on page 495			x	
G_LABELV on page 495			x	
G_SUBGR on page 495	x	x	x	
G_SUBGRV on page 495	x	x	x	
GRADIENTBACKGROUND on page 495		x	x	x
GRADIENTENDCOLOR on page 496		x	x	x
GRADIENTSTARTCOLOR on page 496		x	x	x
HONORASPECT on page 496				x
IMAGEPOSX on page 496		x	x	x
IMAGEPOSY on page 496		x	x	x
LEGENDFIT on page 496				x
LEGENDFONT on page 496				x
LEGENDFONTSIZE on page 496				x
LEGENDHEIGHTPERCENT on page 496				x
LEGENDPERCENT on page 496				x
LEVELOFDETAIL on page 497				x

Parameter	ActiveX	Graph Applet	Map Applet	Contour Applet
LEGENDWIDTHPERCENT on page 496				x
LIGHTING on page 497				x
LOADFUNC		x		
LOCALE on page 497		x	x	x
LODCOUNT on page 497				x
MENUREMOVE on page 497		x	x	
MINLEGENDFONTSIZE on page 497				x
MISSINGCOLOR on page 498				x
NAME on page 498		x	x	x
NAVIGATERENDERMODE on page 498				x
NOJSOOBJECT on page 498		x		
OUTLINES on page 498				x
OVERFLOWCOLOR on page 498				x
PATTERNSTRIP on page 498	x	x	x	
PROJECTION on page 498				x
PROJECTIONRATIO on page 498				x
RENDERMODE on page 498				x
RENDEROPTIMIZE on page 499				x
RENDERQUALITY on page 499			x	x
SHOWBACKDROP on page 499		x		x
SIMPLEDEPTHSORT on page 499				
SIMPLETHRESHOLD on page 500		x		
STACKED on page 500				x
STACKPERCENT on page 500				x
SURFACESIDECOLOR on page 500				x
TIPBACKCOLOR on page 500				x
TIPBORDERCOLOR on page 500				x
TIPS on page 500	x	x	x	x
TIPMODE on page 500	x			
TIPSTEMSIZE on page 501				x
TIPTEXTCOLOR on page 501				x
UNDERFLOWCOLOR on page 501				x
USERFMTn on page 501				x
VIEW2D on page 501	x	x		x

Parameter	ActiveX	Graph Applet	Map Applet	Contour Applet
VIEWPOINT on page 501				x
XBINS on page 501				x
YBINS on page 501				x

* This option works only if STYLE=MINIMAL is specified in ODS destination statement.

Parameter Definitions

AMBIENT=*light-level*
specifies the intensity of non-directional ambient light in relation to direct light. Valid values range from 0.0 to 1.0. The default value is 0.4. The sum of direct light (see the DIRECT parameter) and ambient light can never exceed 1.0. Direct light is given priority. If you specify a sum of these two values that is greater than one, the ambient value will be reduced so that the sum of the two values equals one. This parameter is valid in the ActiveX control and for the Contour applet.

BACKDROPCOLOR=*color*
specifies the color of all walls in the applet, including the floor. The default value is white. This parameter is valid only in the Contour applet.

BACKIMAGE=*image-URL*
specifies the URL of the image that is applied to the background of the applet image area. By default, no image is used and the background is drawn in a single solid color. The way that the image will be applied to the background is specified with the DRAWIMAGE parameter. For the ActiveX control, the background image must be in GIF, JPEG, or BMP format. For the Graph, Map, or Contour applet, the URL must be absolute and not relative.

CLIPTIPS=TRUE | FALSE
indicates whether data tips should be clipped. The default value of TRUE does not display data tips when the cursor is outside of the plot area. A value of FALSE displays data tips when the cursor is outside of the plot area. The data tips window hugs the boundary and displays the value of the element that is closest to the cursor along that edge of the plot. This parameter is valid only in the Contour applet.

COLORNAMELIST=*string*
specifies which of two named color lists has priority when searching for named colors. The default is to search the list of HTML 3.2 colors first, followed by the SAS name list. Specifying SAS as the string reverses this priority, giving SAS names higher priority. This parameter is valid only in the Contour applet.

COLORNAMES=*name1=value1,name2=value2, ... nameN=valueN*
specifies the color names and associated 6-digit hexadecimal RGB values that will be displayed in the Standard Colors list box in the Color Edit dialog box. In the parameter value, no white space is allowed. The color name can be any valid string, and is displayed as specified in the list box. This parameter is valid in the Graph, Map, and Contour applets.

COLORSCHEME=*scheme-name*
specifies the name of the color scheme that is applied to the graph. By default, no color scheme is applied to the graph. This parameter is valid in the ActiveX control and the Graph applet.

DDLEVELn*configuration-string*
configures the drill-down graph that is generated at the drill-down level that is specified by the letter *n*. The drill-down graph is configured using drill-down tags such as G_INDEPV. For details, see "Configuring Drill-Down Links with ACTIVEX" on page 460. This parameter is valid in the ActiveX control and in the Graph and Map applets.

DIRECT=*light-level*
specifies the intensity of direct light (from a light source) in relation to the ambient (non-directional) light. Valid values range from 0.0 to 1.0. The default value is 0.6. The sum of direct light and ambient light (see the AMBIENT parameter) cannot exceed 1.0. Direct light is given priority. If you specify a sum of these two values that is greater than one, the level of ambient light will be reduced so that the sum of the two values equals one. This parameter is valid in the ActiveX control and the Contour applet.

DRAWIMAGE=*background-image-application*
specifies how the image specified in the BACKIMAGE parameter is applied to the background of the applet window. This parameter is valid for the Graph, Map, and Contour applets. Here are the valid values:

CENTER
centers a single instance of the image in the background, without resizing the image.

POSITION
places a single instance of the image at the location supplied by the IMAGEPOSX and IMAGEPOSY parameters, without resizing. If these parameters are not specified, then the image is centered in the applet window.

SCALE
fills the entire background of the applet window with a single instance of the specified image, which is resized as necessary.

TILE
fills the entire background of the applet window using multiple instances of the specified image, without resizing that image. The images are arranged in rows and columns.

DRAWMISSING=TRUE | FALSE
specifies whether missing values should be drawn. By default, missing values are not drawn. Missing values are drawn only when this parameter is set to TRUE and the **Styles** menu option is set to Block, Smooth, or Surface. This parameter is valid only in the Contour applet.

DRAWSIDES=TRUE | FALSE
specifies that sides should be drawn when the value of the STACKED parameter is TRUE and when the **Styles** menu option is set to Surface, Areas, or LinesAndAreas. The default value is FALSE. To override this parameter, you can specify an ODS style definition. This parameter is valid in the Contour applet.

DRILLDOWNFUNCTION=*function-name*
DRILLFUNC=*function-name*
specifies the name of the JavaScript function that is called in Script drill-down mode. This parameter is valid in the ActiveX control and in the Graph and Map applets.

DRILLDOWNMODE=HTML | LOCAL | SCRIPT | URL
specifies the drill-down mode. This parameter is valid in the ActiveX control and in the Graph and Map applets. Here are the valid 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.

Local mode (Graph applet only)
constructs and displays a new graph based on the data in the previous level of a drill-down graph.

Script mode
invokes the JavaScript function specified in the DRILLDOWNFUNCTION parameter, and passes into the function data from the selected graph element.

URL mode
provides static drill-down, using an image map in the HTML file. The image map is generated using the IMAGEMAP= and HTML= options in SAS/GRAPH.

The default drill-down mode is Local for the Graph applet. The Map applet and the ActiveX control do not enable user-selectable drill-down modes.

DRILLPATTERN=*substitution-string*
specifies how to construct the drill-down URL when the drill-down mode is HTML. 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 460. This parameter is valid in the ActiveX control and in the Graph and Map applets.

DRILLTARGET=*target*
specifies where the drill-down destination is displayed in the browser. The default target is _BLANK, which is an HTML reserved word that displays the drill-down destination in a new browser window. The target can be specified as another reserved target name or as the name of a window or frame in your Web presentation. This parameter is valid in the ActiveX control and in the Graph and Map applets.

DUPLICATEVALUES=*string*
determines how the applet will handle data values for grid positions that already have a data value. This parameter is valid in the Contour applet. Specify one of the following values:

COUNT
stores at each grid location the number of values found for that location.

FIRST
stores the first value found.

LAST
stores the last value found.

MAX
stores the maximum value found.

MEAN
stores the mean (average) of all values found. This is the default value.

MIN
stores the maximum value found.

NMISS
stores the number of missing values found.

RANGE
stores the range of values found. The range is computed as the maximum value minus the minimum value.

SUM
stores the sum of all values found.

FILLPOLYGONEDGES=ALWAYS | NEVER | OS/2
specifies whether to adjust rendering to fix a temporary vendor rendering defect. This parameter is valid only in the Contour applet. When you set the value to ALWAYS, the adjusted rendering is always performed, regardless of the operating system on which the applet is running. Similarly, if you set the value to NEVER, the adjusted rendering is never performed on any operating system. If the value of this parameter equals the os.name Java system property, then the Contour applet sets the default value of this parameter to OS/2, which lets draw Polygon correctly fill in (render) the polygon edges, yet this extra drawing effort slows performance. If you set this parameter to the value of the parameter of the name of the operating system returned in os.name, then the adjusted rendering is performed when the applet runs on that operating system because the applet notifies the Java console.

FREQNAME=*variable-name*
specifies a name for a new variable that contains the frequency count when a frequency chart is produced. By default, the name assigned to this variable is "Frequency". This parameter might be overridden if you specify an ODS style definition. This parameter is valid in the Graph applet.

G_COLOR=*variable-name*
specifies a new color variable for the current drill-down level. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_COLORV=*variable-name*
specifies that the current color variable is the same variable that was used to configure the previous drill-down level. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_DEP=*variable-name*
specifies a new dependent variable for the current drill-down level. This parameter is valid in the ActiveX control and in the Graph and Map applets.

Note: The value of the G_DEP tag cannot be set to *None* because it is always represented in the graph. △

G_DEPV=*variable-name*
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. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_DEPTH=*variable-name*
specifies a new depth variable for the current drill-down level. Drill-down graphs that use this variable can be vertical bar charts or scatter plots. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_DEPTHV=*variable-name*
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. Drill-down graphs that use this variable can be vertical bar charts or scatter plots. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_GROUP=*variable-name*
specifies a new group variable for the current drill-down level. Drill-down graphs that use this variable can be bar charts. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_GROUPV=*variable-name*
specifies that this group variable should be the same group variable that was used at the previous drill-down level. Drill-down graphs that use this variable can be bar charts. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_INDEP=*variable-name*
specifies a new independent variable for the current drill-down level. Drill-down graphs that use this variable can be charts and maps. This parameter is valid in the ActiveX control and in the Graph and Map applets.

Note: The values of the G_INDEP tags cannot be set to *None* because it is always represented in the graph. △

G_INDEPV=*variable-name*
specifies that an independent variable at the current drill-down level is the same variable that was used at the previous drill-down level. Drill-down graphs that use this variable can be charts and maps. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_LABEL=*variable-name*
specifies a new label variable for the current drill-down level. Drill-down graphs that use this variable can be maps. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_LABELV=*variable-name*
specifies that this label variable should be the same label variable that was used at the previous drill-down level. Drill-down graphs that use this variable can be maps. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_SUBGR=*variable-name*
specifies a new subgroup variable for the current drill-down level. Drill-down graphs that use this variable can be bar charts and scatter plots. This parameter is valid in the ActiveX control and in the Graph and Map applets.

G_SUBGRV=*variable-name*
specifies that a subgroup variable at this drill-down level is the same subgroup variable that was used at the previous drill-down level. Drill-down graphs that use this variable can be bar charts and scatter plots. This parameter is valid in the ActiveX control and in the Graph and Map applets.

GRADIENTBACKGROUND=TRUE | FALSE | VERTICAL | HORIZONTAL
specifies that the background of the window is or is not using a color gradient. To override this parameter, you can specify an ODS style definition. This parameter is valid in the ActiveX control and in the Graph, Map, and Contour applets. TRUE and FALSE are valid only for the Graph and Map applets. VERTICAL and HORIZONTAL specify the orientation of the color gradient and are valid only for the Contour applet. This parameter is ignored in the Contour applet if you specify the BACKIMAGE parameter. Use GRADIENTSTARTCOLOR and GRADIENTENDCOLOR to define the colors used to draw the background.

GRADIENTENDCOLOR=*color*
GRADIENTSTARTCOLOR=*color*
specify the start color and the end color when two colors are blended in a gradient across a wall, background, or graph element. The color can be an HTML 3.2 color name or a 6-digit hexadecimal RGB value. This parameter might be overridden if you specify an ODS style definition. This parameter is valid in the ActiveX control and in the Graph, Map, and Contour applets.

HONORASPECT=TRUE | FALSE
specifies whether the aspect of the data being displayed is or is not honored. The default value FALSE scales the shortest axis (x or y). This parameter is valid in the Contour applet. Note that certain annotations, such as pies, might display differently in the applet than in SAS when the value is FALSE.

IMAGEPOSX=*horizontal-pixels*
IMAGEPOSY=*vertical-pixels*
specify the location of the upper-left corner of the background image that is named in the BACKIMAGE parameter. These parameters are ignored unless the value of the DRAWIMAGE parameter is POSITION. Positive pixel values are measured from the top-left corner of the applet window. Negative pixel values are measured from the bottom-right corner of the applet window. These parameters are valid in the ActiveX control and in the Graph, Map, and Contour applets.

LEGENDFIT=TRUE | FALSE
specifies whether the legend should fit within the height of the contour plot area. By default the legend occupies as much of the applet height as is feasible. If TRUE, the height of the legend is restricted to the height of the contour plot within the legend. When you set this parameter, any value specified for LEGENDHEIGHTPERCENT is ignored. This parameter is valid only in the Contour applet.

LEGENDFONT=*font*
specifies which font to use in the legend. Except for the case, the font name must match the name of a Java font available in the browser. This parameter is valid only in the Contour applet.

LEGENDFONTSIZE=*font-size*
specifies the default size of the font to be used in the legend. Only positive values are valid. This parameter is valid only in the Contour applet.

LEGENDHEIGHTPERCENT=*percentage*
restricts the height of the legend to a specified percentage of the height of the Contour applet. A vertical margin is always maintained. Valid values are greater than 0 and less than 100 percent, with the default value being 20. This parameter is valid only in the Contour applet.

LEGENDPERCENT=*percentage*
specifies how much of the Contour applet space (width) to use as the legend area. Valid values are 0 to 80 percent. The default value is 20. This parameter is valid only in the Contour applet.

LEGENDWIDTHPERCENT=*percentage*
restricts the width of the legend to a specified percentage of the width of the Contour applet. A horizontal margin is always maintained. Valid values are greater than 0 and up to 80 percent, which the default value being 20. This parameter is valid only in the Contour applet.

LEVELOFDETAIL=TRUE | FALSE
specifies whether the level-of-detail processing should be used when drawing plots. The default value is TRUE, which allows level-of-detail processing. See also the LODCOUNT parameter. This parameter is valid only in the Contour applet.

LIGHTING=HEADLIGHT | OVERHEAD | NORTHEAST | SOUTHEAST
specifies the position of the light source relative to the position of the graph. The default value is HEADLIGHT, which directs two light sources at the graph from the front-center of the screen. This parameter is valid in the Contour applet.

LOADFUNC=*Java-method*
specifies the name of a JavaScript method in the HTML output file that loads values and specifications. This parameter is valid in the Graph applet. This parameter should not be specified if you are using ODS.

LOCALE=*xx_yy<_variant>*
specifies the language and country to use when displaying locale-sensitive text. This parameter is valid in the Graph, Map and Contour applets. Here are the values for this parameter, which are java.util locale specifiers:

xx
represents the required two-digit ISO-639 language code, as defined at **http://www.loc.gov/standards/iso639-2/**.

yy
represents the required two-digit ISO-3166 country code, as defined at **http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm**.

<_variant>
represents the optional variant code, which depends on the browser and operating environment. If a variant is specified, the initial underscore character is required.

LODCOUNT=*number-of-cell(s)*
specifies the number of cells to use as the level-of-detail threshold. The default value is 2000. When the number of cells involved in drawing a plot in the applet exceeds this value and level-of-detail processing is on, then some cells are ignored when rendering the plot representation. See also the LEVELOFDETAIL parameter. This parameter is valid only in the Contour applet.

MENUREMOVE=*menu-item(s)*
disables items in the Graph applet menu and in the Map applet menu. Here is the syntax of *menu-item(s)*:

menu1-item<.menu2-item... .menuN-item, *menu–item2, ...menu-itemN>*

In the *menu-item(s)* value, periods (".") separate menu levels in menu paths. In menu paths, the menu item that is disabled is the last item in the path. Commas separate menu items and menu paths in a series. Menu items are specified using the text that is displayed by the applet, with blank spaces removed. For example, the menu item Graph Properties would be specified as GRAPHPROPERTIES. To apply the MENUREMOVE parameter, first generate the graph without the MENUREMOVE parameter. Then note the menu paths of the items that you want to disable. This parameter is valid in the Graph applet and the Map applet.

MINLEGENDFONTSIZE=*font*
specifies the minimum font to be used when attempting to fit the legend in the available applet area. Only positive integers are valid values. This parameter is valid only in the Contour applet.

MISSINGCOLOR=*color*
specifies an HTML 3.2 color name or 6-digit hexadecimal RGB value that is to be used to draw missing values. The default color is black. This parameter is valid in the Contour applet.

NAME=*applet-name*
specifies the name for this instance of the applet. Use this parameter only if you have more than one instance of the APPLET tag in your HTML file, and if you have included your own scripts or DHTML that communicates with or acts on a particular instance of the applet. This parameter might be overridden if you specify an ODS style definition. This parameter is valid in the Graph, Map, and Contour applets.

NAVIGATERENDERMODE=NONE | POINT | SOLID | WIREFRAME
specifies how to render the graph during pan, rotate, and zoom. The default value is WIREFRAME. This parameter is valid when the RENDERQUALITY parameter is set to CUSTOM. This parameter might be overridden if you specify an ODS style definition. This parameter is valid in the Contour applet.

NOJSOOBJECT
specifies that no JavaScript callback options can be created or used within the applet. This parameter might be overridden if you specify an ODS style definition. This parameter is valid in the Graph applet.

OUTLINES=TRUE | FALSE
specifies whether outlines should be drawn for the current contour style. Outlines are drawn when this parameter is TRUE and the **Styles** menu option is set to Area, Block, or Surface. This parameter is valid only in the Contour applet.

OVERFLOWCOLOR=*color*
specifies an HTML 3.2 color name or a 6-digit hexadecimal RGB color for colors that are assigned to data values that exceed the maximum range of colors that have been defined in the style or color list. The default value is CYAN. This parameter is valid in the ActiveX control and in the Contour applet.

PATTERNSTRIP=TRUE | FALSE
removes preceding and trailing white space from drill-down substitution patterns before the substituted text is added into a dynamically generated drill-down URL. The default value is FALSE. This parameter is valid in the ActiveX control and in the Graph and Map applets.

PROJECTION=ORTHOGRAPHIC | PERSPECTIVE
specifies the type of projection that is used to draw contours. The default value is ORTHOGRAPHIC. This parameter is valid in the Contour applet.

PROJECTIONRATIO=*plot-size-ratio*
specifies the ratio of the plot area (applet size minus legend reserve) to the longest dimension of the plot. For example, specifying a value of 2.0 means that the area that contains the contour plot is twice the size of the longest plot dimension. This guarantees that the plot will be surrounded by a space that measures half the length of the longest projection (not including axes). The default value is 1.5. Values must be greater than or equal to 1.0. This parameter is valid in the Contour applet.

RENDERMODE=*string*
specifies how to render the contours when you are not navigating (panning, rotating, or zooming) the Contour applet. This parameter is valid only in the Contour applet. In some cases, changing the representation can provide additional information about the image, such as more clearly displaying cell boundaries.

Here are the valid values for the polygon representations that determine how the Contour applet image can be drawn:

POINT
draws polygons using only single-pixel points at the polygon vertices.

SOLID
draws filled polygons. This is the default value and the normal representation.

WIREFRAME
draws polygons using only lines to represent their edges.

RENDEROPTIMIZE=ALWAYS| NAVIGATION | NEVER | ONNAVIGATION
sets the default for rendering optimization for the Contour applet. This parameter is valid only in the Contour applet. To correctly render images, the applet must first sort the polygons that comprise the image. Some polygons require additional sorting steps to ensure that they are correctly drawn. In many cases, these additional steps are unnecessary because they only slow applet performance and do not add to image quality. This parameter lets you specify if and when the applet should attempt to optimize or reduce the number of sorting operations to be performed. The RENDEROPTIMIZE parameter is ignored unless you set the RENDERQUALITY parameter to CUSTOM. The default value depends on the value of the RENDERQUALITY parameter.

When the RENDERQUALITY parameter is set to BESTQUALITY, the default value for the RENDEROPTIMIZE parameter is NEVER.

When the RENDERQUALITY parameter is set to FASTERNAVIGATION, the default value for the RENDEROPTIMIZE parameter is ONNAVIGATION.

When the RENDERQUALITY parameter is set to BESTPERFORMANCE, the default value for the RENDEROPTIMIZE parameter is ALWAYS.

RENDERQUALITY=*value*
specifies how two available rendering algorithms, one slower and one faster, are applied to the graph. This parameter might be overridden if you specify an ODS style definition. This parameter is valid for the Map and Contour applets. Here are the valid values:

BESTPERFORMANCE | PERFORMANCE
always uses the faster, less complex rendering algorithm.

BESTQUALITY | QUALITY
always uses the slower, more complex rendering algorithm.

FASTERNAVIGATION | NAVIGATION
uses the faster, less complex rendering algorithm during pan, rotate, and zoom, and uses the more complex algorithm otherwise. This is the default value.

CUSTOM (Contour applet only)
lets the user select individual elements that control speed and quality directly, rather than as a group when rendering an image.

SHOWBACKDROP=TRUE | FALSE
specifies whether all walls (including the floor) should be displayed. This parameter overrides any ODS settings and is valid only in the Contour applet.

SIMPLEDEPTHSORT=TRUE | FALSE
the default value TRUE indicates that the simpler polygon sorting algorithm is used when rendering the plot. This parameter is valid in the Contour applet.

SIMPLETHRESHOLD=*number-of-elements* | NEVER
specifies an integer for the threshold that is used to determine whether the graph should be rendered using simple geometry. For bar charts, simple geometry means that graphical elements are represented as lines. For plots, simple geometry means that graphical elements are represented as plus signs (+).

If the graph contains a number of elements that is greater than the SIMPLETHRESHOLD value, simple geometry is used and the Shape menu is made unavailable. The default value is 500. You can also specify the value NEVER. In that case, simple geometry is never used and the Shape menu is always available.

Note that if you select and display a subset of the graph, and if the number of elements in the resulting graph drops below the value of the SIMPLETHRESHOLD parameter, regular markers are drawn and the Shape menu is made available.

This parameter is valid in the Graph applet.

STACKED=TRUE | FALSE
specifies whether the contours should be displayed in stacked form, where height is added to the contour plot based on the contour level. This parameter takes effect only when the Style menu option is set to Areas or LinesAndAreas. The default value of this parameter is FALSE. See also the DRAWSIDES parameter. This parameter is valid in the Contour applet.

STACKPERCENT=*height-percentage*
specifies the maximum stacking height as a percentage of the longest axis. The default value is 30. This parameter is valid in the Contour applet.

SURFACESIDECOLOR=*color*
specifies the color of the sides of a contour plot when that plot uses multiple colors. The value of the parameter is ignored when drawing a surface plot in a single color. The default color is the color of the minimum data value. The value must be an HTML 3.2 color name or a 6-digit hexadecimal RGB value. This parameter is valid in the Contour applet.

TIPBACKCOLOR=*color*
specifies an HTML 3.2 color name or a 6-digit hexadecimal RGB value for the background of the data tips. The default value is YELLOW. This parameter is valid in the Contour applet.

TIPBORDERCOLOR=*color*
specifies an HTML 3.2 color name or a 6-digit hexadecimal RGB value for the border of the data tips. The default value is BLACK. This parameter is valid in the Contour applet.

TIPS=NONE | STATIONARY | TRUE | FALSE
specifies whether to display data tips. NONE and STATIONARY are valid values only for the Graph and Map applets, and TRUE and FALSE are valid only for the Contour applet. Specifying the default value of STATIONARY or TRUE enables displays data tips, and NONE and FALSE disables this. This parameter is valid in the ActiveX control and in the Graph, Map, and Contour applets.

TIPMODE=STANDARD | HTML | TABULAR | ALL
specifies which of two types of data tips are to be displayed. One set of data tips is specified with the TIPS parameter. The other set of data tips is specified with the HTML= statement option. Specify TIPMODE=HTML to display only the data tips that are indicated by the HTML= statement option. Specify TIPMODE=TABULAR to display only the data tips that are indicated by the value of the TIPS parameter. Specify TIPMODE=STANDARD to display both sets of data tips. The default value is STANDARD.

To display data tips with the HTML= statement option. You can specify the HTML= option. The syntax of that option is HTML="ALT='*text*' | *variable-name*". For further information on data tips, see "Data Tips for Web Presentations" on page 598.

TIPSTEMSIZE=*line-length*
specifies the length in pixels of the line that connects the data tips to the graph element that makes use of that data. The default value is 20. This parameter is valid in the Contour applet.

TIPTEXTCOLOR=*color*
specifies an HTML 3.2 color name or a 6-digit hexadecimal RGB value for the text in the data tips. The default value is BLACK. This parameter is valid in the Contour applet.

UNDERFLOWCOLOR=*color*
specifies an HTML 3.2 color name or a 6-digit hexadecimal RGB value for the color that is assigned to data values that are smaller than the minimum range of colors that have been defined in the style or color list. The default value is WHITE. This parameter is valid in the Contour applet.

USERFMT*n*=*string(s)*
defines the user format specification. The syntax is the same as that of the VALUE and PICTURE statements for PROC FORMAT. You can specify multiple USERFMT*n* parameters by replacing *n* with the appropriate number from 1 to *n*, where *n* is the number of format parameters to be defined. For example, to define a simple YESNO format, specify the parameter <PARAM NAME="USERFMT1" VALUE="VALUE YESNO 1='Yes' 2='No' ">. This parameter is valid only in the Contour applet.

VIEW2D=TRUE | FALSE
indicates whether the view point should be locked to two dimensions. The default value is TRUE for the Contour applet and FALSE for the Graph applet and ActiveX control. This parameter might be overridden if you specify an ODS style definition.

XBINS=*bin-number-or–values*
YBINS=*bin-number-or-values*
configures the bins uses to generate a contour plot. Specifying a single integer uses that number of bins. The single integer must be greater than 2. Specifying multiple values uses multiple bins with those values. Multiple values are real numbers that are separated by semicolons, as follows:

```
ods html file=filename.html
   parameters=("XBINS"="-1;0;2.5;3.5;4"
               "YBINS"="1;2;3;4;5;6");
```

These parameters are enabled in the Contour applet.

VIEWPOINT=2D| SE | SOUTHEAST
defines the initial viewpoint for the Contour applet. The value SE or SOUTHEAST set the initial viewpoint to Southeast, a three-dimensional viewpoint. The value 2D sets the value to be two-dimensional. The default value is 2D for PROC GCONTOUR output and SOUTHEAST for PROC G3D. Setting this parameter unlocks the 2D view. (See VIEW2D.) This parameter is valid only in the Contour applet.

CHAPTER

20

Generating Static Graphics

What is a Static Graphic?

A static graphic is a graphic that is permanently fixed after it is displayed. You can view a static graphic but you cannot manipulate it as you view it in a browser. Examples of static graphics include GIF and PNG images. To generate a static graphic, in your SAS program, run a SAS graphics procedure and specify with the DEVICE= graphics option on one of the following devices:

PNG

GIF

SVG

JPEG

ACTXIMG

JAVAIMG

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. To enable you to

create GIF images with different default dimensions, the following GIF device variants are provided:

GIF160	160 x 120
GIF260	260 x 195
GIF373	373 x 280
GIF570	570 x 430
GIF733	733 x 550

However, we recommend that you use the XPIXELS= and YPIXELS= graphics options with the GIF device 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 95.

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
Z*device*	devices provided for compatibility with previous releases of the SAS/GRAPH software

See Chapter 6, "Using Graphics Devices," on page 67 for more information on these devices.

When you send your graph output to the ODS HTML destination, you can add data tips and drill-down links to your static graphic. See Chapter 27, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 595.

Creating a Static Graphic

You can use a GOPTIONS statement with a device type of GIF, JPEG, SVG, or PNG to create a static 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 static 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, filename, 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 102.

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 filename 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 102.

3 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`**.

To create an HTML file that embeds the image, use the ODS HTML destination with the following options:

BODY= The filename of the output HTML file (FILE= is a synonym for BODY=).

PATH= The location (URL or file reference) of the HTML file and static graphic 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 filename, and you do not specify the PATH= option.

If you specify FILE= as just a filename (and extension), and you 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 "Sample Programs for Static Images" on page 512.

For complete information on these options, see *SAS Output Delivery System: User's Guide*.

ACTXIMG and JAVAIMG Devices Compared to GIF, JPEG, SVG, and PNG Devices

GIF, JPEG, SVG, and PNG Devices

When you use the graphics option DEVICE=GIF, JPEG, SVG, or PNG with a SAS/GRAPH procedure, an ODS style is applied to your graph by default—that is, 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 on the GSTYLE system option, see *SAS Language Reference: Dictionary*.

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 GIF, JPEG, PNG, and SVG Presentations" on page 604.

ACTXIMG and JAVAIMG Devices

Like the GIF, JPEG, SVG, and PNG devices, when you use the graphics option DEVICE=ACTXIMG or JAVAIMG with a SAS/GRAPH procedure, an ODS style is applied to your graph by default. You can apply any of the ODS styles to your graph. Unlike the GIF, JPEG, SVG, and PNG 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 506.) An ODS style is always applied to a graph that is generated by the ACTXIMG or JAVAIMG device.

You can also add data tips to your graph (see "Data Tips in ACTIVEX, ACTXIMG, JAVA, and JAVAIMG Presentations" on page 600) and drill-down links to other URLs.

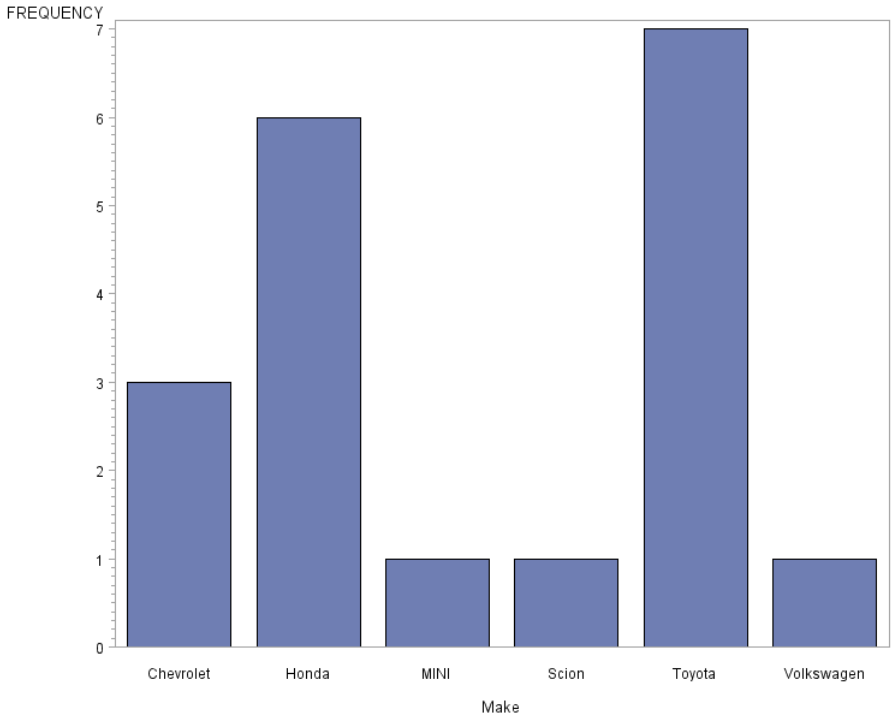
Output From Different Devices and the GSTYLE/NOGSTYLE System Options

The static devices all support ODS styles. However, the ACTXIMG and JAVAIMG devices are not affected by the GSTYLE|NOGSTYLE system option, so an ODS style is applied to an ACTXIMG or JAVAIMG image regardless of the setting of this system option. To demonstrate the impact of the GSTYLE and NOGSTYLE system options on the device output, here is an example that applies the Statistical style to a GIF image with the GSTYLE system option on:

```
options gstyle;
ods listing close;
ods html style=statistical;
goptions reset=all device=gif;

proc gchart data=sashelp.cars;
   vbar Make;
 where MPG_Highway >= 37;
   run;
  quit;
```

```
ods html close;
ods listing;
```

Display 20.1 A Bar Chart Using the GIF Device with the Statistical Style and GSTYLE System Option

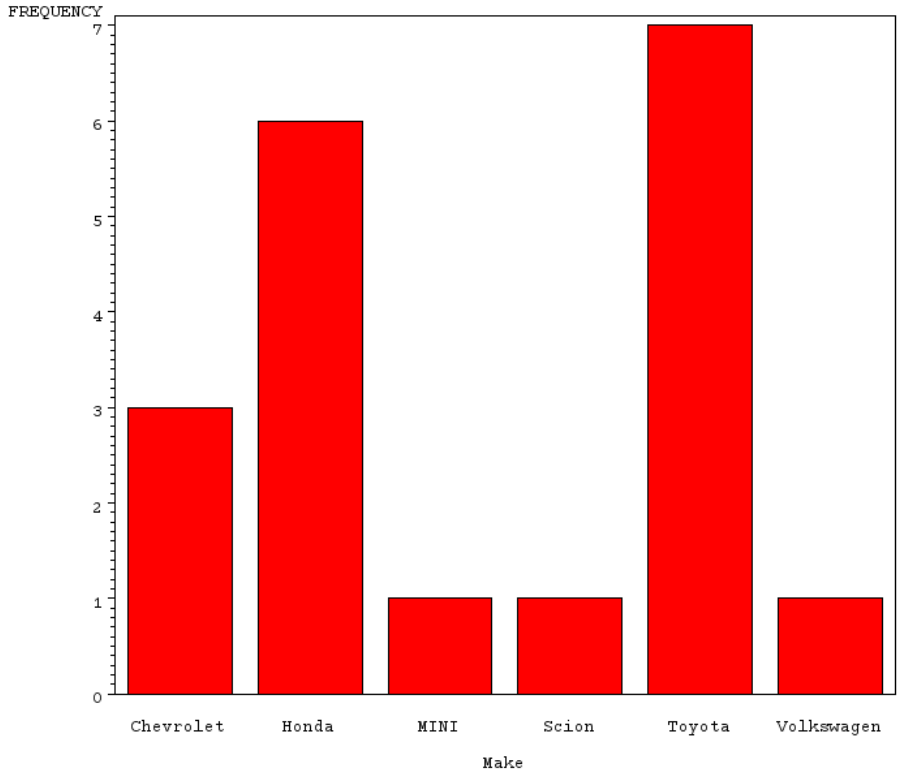


The output is similar for the other devices.

Here is an example that applies the Statistical style to a GIF image with the NOGSTYLE system option on:

```
options nogstyle;
ods listing close;
ods html style=statistical;
goptions reset=all device=gif;

proc gchart data=sashelp.cars;
     vbar make;
     where MPG_Highway >= 37;
run;
quit;

ods html close;
ods listing;
```

Display 20.2 A Bar Chart Using the GIF Device with the Statistical Style and the NOGSTYLE System Option



Notice that STYLE=STATISTICAL 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 GIF, JPEG, SVG, and PNG 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 Statistical style is applied even though the NOGSTYLE system option is on.

Developing Web Presentations with the GIF, JPEG, SVG, and PNG Devices

You can use the GIF, JPEG, SVG, and PNG devices to create Web presentations that consist of static graphics. You can also add data tips and drill-down links to your graphs.

About the GIF, JPEG, SVG, and PNG Devices

The GIF, JPEG, SVG, and PNG devices enable you to generate static 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 “Generating an HTML Output File Using the GIF, PNG, SVG, or JPEG Device” on page 509.

Enhancements that are available to GIF, PNG, SVG, and JPEG Web presentations include adding drill-down links or tool-tip 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 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 graphics 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 511.

When to Use the GIF, JPEG, SVG, and PNG Devices

The GIF, JPEG, SVG, and PNG devices are best suited to Web presentations that consist of static graphs and graphs with simple drill-down capabilities. 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 or JAVA device. The GIF device provides only 256 colors, which might be suitable for many presentations. The PNG, SVG, and JPEG devices provide TruColor support and are better suited for Web presentations that contain color-intensive graphics.

Generating an HTML Output File Using the GIF, PNG, SVG, or JPEG Device

Follow these steps to generate a complete Web presentation that consists of an HTML output file and one or more images:

1. To conserve resources, close the ODS LISTING destination (the Output window, which is open by default):

```
ods listing close;
```

2. Enter your DATA step, if necessary.
3. Specify your ODS HTML statement, with the following options:

```
ods html
   path="C:/Public/graph" (url=none)/* HTML output directory */
   body="webgif1.htm"       /* HTML filename              */
   gpath="C:/Public/graph/images"; /* graphics output file  location */
```

   Specifying URL=NONE tells ODS to reference the graphics output file simply by name without prefixing the full path (assuming that the graphics output file is in the same directory as the HTML file).

   Note: With the GIF, JPEG, SVG, or PNG 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 with Java and ActiveX Devices in HTML Output" on page 194. △
4. Specify your device:

```
goptions reset=all device=png;
```

5. Run procedures to generate graphs. For example

```
proc gchart data=sashelp.class;
        hbar3d sex / sumvar=height type=mean;
 run;
```

```
quit;
```

You can use BY statements to create multiple graphs.

6 Close the HTML output file and reopen the ODS listing destination:

```
ods html close;
ods listing;
```

Reopening the listing destination establishes standard operating conditions for later programs that you run in the same SAS session.

7 Open the HTML output file in your Web browser. For example, open file **`C:\Public\graph\webgif1.htm`**.

Note: Using this technique, however, you cannot create drill-down links or data tips for your graphs. △

For an example, see "Generating PNG Output" on page 514.

Developing Web Presentations with the JAVAIMG and ACTXIMG Devices

You can use the JAVAIMG and ACTXIMG devices to create Web presentations that include snapshots of graphs that are generated with the JAVA and ACTIVEX devices, but without the interactivity that JAVA and ACTIVEX graphs provide. You can also add data tips and drill-down links to your graphs.

About the JAVAIMG and the ACTXIMG Devices

The JAVAIMG and ACTXIMG devices enable you to generate Web presentations that display a snapshot of one or more graphs in the PNG format. The ACTXIMG device works on PC hosts only. On all other hosts, the ACTXIMG device 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 on installing the ActiveX control, see "Installing the SAS/GRAPH ActiveX Control" on page 455. 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 17.1 on page 454. The procedures that can be used with the JAVAIMG device are listed in "Graph, Map, Tilechart, and Contour Applets" on page 441.

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 are identical in appearance to the graphs created with the DEVICE=JAVA or DEVICE=ACTIVEX graphics option as they are initially displayed in a browser. However, unlike these latter graphs, which are interactive and can be manipulated by a user viewing them in a browser, the PNG files are static 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 and not in the graphics output files regardless of whether the GTITLE and GFOOTNOTE options are set. See "Controlling Titles and Footnotes with Java and ActiveX Devices in HTML Output" on page 194. △

When to Use the JAVAIMG or ACTXIMG Device

If you do not need interactivity such as changing the chart type or style, the JAVAIMG and ACTXIMG devices provide several advantages over the interactive presentations that are generated with the JAVA and ACTIVEX devices. Because PNG image files are generated, the Web clients are not required to access the Java run-time environment or 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 applet or an ActiveX control.

Note: The ACTXIMG device cannot be used with the ODS PDF, PCL, PS, or PRINTER destinations on 64-bit computers. The SAS/GRAPH software uses the JAVAIMG device instead. △

Some of the SAS/GRAPH procedures, such as the GKPI and GEAREABAR procedures, support only the JAVA, JAVAIMG, ACTIVEX, and ACTXIMG devices. For these procedures, you must use the JAVAIMG or the ACTXIMG device if you need a static image.

Finally, in some cases such as plots generated with the G3D procedure, the ACTXIMG and JAVAIMG devices provide a better static image than the other devices.

Note: When SAS is installed on a server, the ACTXIMG and JAVAIMG devices are limited by the display capabilities of the server on which they run—for example, the number of colors that the server is capable of. Consequently, the ACTXIMG or JAVAIMG PNG snapshot might not look as good as what you get from the JAVA and ACTIVEX devices. Therefore, it is better to use the JAVA or ACTIVEX device if the server's display settings are less than optimal. △

Using JAVAIMG in the z/OS Environment

If you are running SAS in the z/OS operating environment with the DEVICE=JAVAIMG graphics option, then you must specify FILESYSTEM=HFS 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 an HTML output file for the GIF, PNG, SVG, or JPEG devices. See "Generating an HTML Output File Using the GIF, PNG, SVG, or JPEG Device" on page 509.

For an example, see "Using the ACTXIMG Device" on page 512.

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 ACTXIMG, JAVAIMG, GIF, JPEG, SVG, or PNG device. For information on the default

configurations of these Web presentations, see “GIF Output with Drill-Down Links” on page 515.

You can add drill-down links to the following elements:

- graph elements or legend elements or both. See “GIF Output with Drill-Down Links” on page 515.
- graph elements specified in an Annotate data set. See “Generating Web Links with the Annotate Facility” on page 540.
- titles and footnotes using the LINK= option in the TITLE or FOOTNOTE statement.

Sample Programs for Static Images

The following sections describe how to create a Web presentation using static images:

Using the ACTXIMG Device

Here is an example that uses the ODS HTML destination to create an HTML file that references four PNG files that are created by the GCHART procedure with the DEVICE=ACTXIMG graphics option. Because the ACTXIMG device invokes an 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.

Display 20.3 Using ODS with the ACTXIMG Device

The following is the complete SAS code for this example. In this example, the output files are sent to the default location. If you want to send the output files to a different location, add the BODY= option to the ODS HTML statement to specify the new location of the output files. You can specify the complete path and filename with the BODY= option (or the FILE= option, which is the same), or you can specify the path separately using the PATH= option, and just the filename with the FILE= or BODY= 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 the section "ODS HTML Statement" in the *SAS Output Delivery System: User's Guide*.

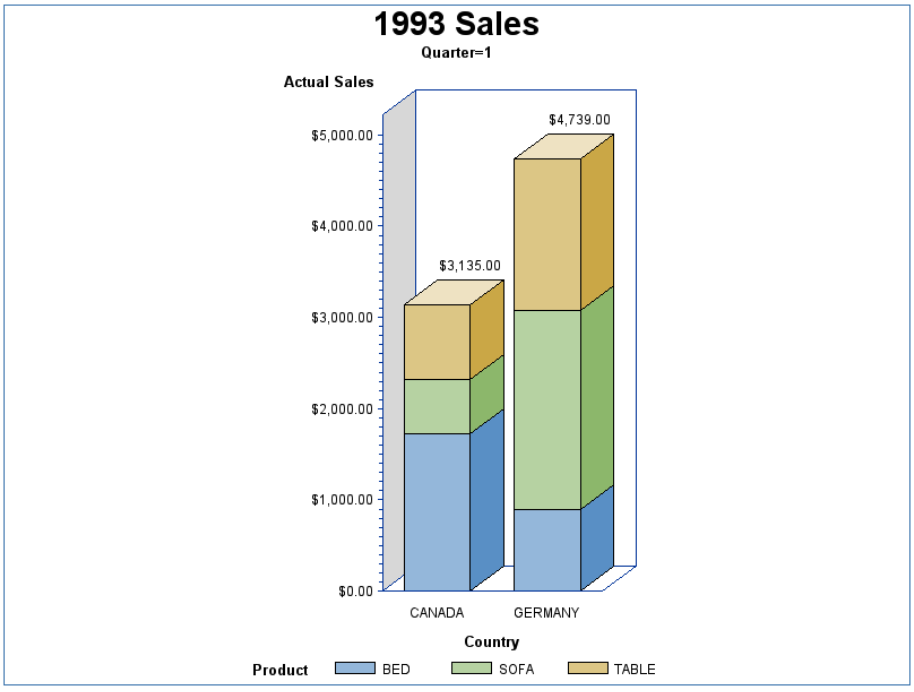
```
 /* Create data set from sashelp.prdsale */
data prdsummary;
  set sashelp.prdsale;
  where year=1993 and (country = "GERMANY" or country = "CANADA")
        and region="EAST" and division="CONSUMER" and
        (product="SOFA" or product="TABLE" or product="BED");
run;

/* Sort the data set by quarter */
proc sort data=work.prdsummary;
 by quarter;
run;

/* Since the LISTING destination is not used, close it to save system resources */
ods listing close;
```

```
/* Send output to an HTML file */
ods html 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 HTML file */
ods html close;

/* Reopen the LISTING destination */
ods listing;
```

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.

Display 20.4 Generating PNG Output Using ODS



The following is the complete SAS code for this example. In this example, the output files are sent to the default location. If you want to send the output files to a different location, add the BODY= option to the ODS HTML statement to specify the new location of the output files. You can specify the complete path and filename with the BODY= option (or the FILE= option, which is the same), or you can specify the path separately using the PATH= option, and just the filename with the FILE= or BODY= option. See the section "ODS HTML Statement" in the *SAS Output Delivery System: User's Guide*.

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.

```
/* 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;
ods listing close;
ods html 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 listing;
```

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.

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 27, "Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality," on page 595.)

In this example, the DEVICE=GIF graphics option generates image output files and the ODS HTML statement generates an HTML output file. The HTML= 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.

When you display the HTML output file in a Web browser, the following chart is displayed.

Display 20.5 Three-Dimensional 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 block, 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, which is available in the SAS Sample Library under the name GWBDRILL:

```
 /* Close the LISTING destination. */
ods listing close;

 /* Set graphic options. */
goptions reset=all border device=gif;

 /* 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 link variable. */
if Region="Central" then
     rpt="href='central.html'";
   else if Region="South" then
     rpt="href='south.html'";
   else if Region="West" then
     rpt="href='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
;

 /* Open the HTML destination for ODS output. Specify the */
 /* filename in BODY=.       */

ods html body="company.html" style=statistical;

 /* Create a chart that uses the link variable.  */
title1 "Company Sales";
proc gchart data=regsales;
   vbar3d region / sumvar=sales
   patternid=midpoint
   html=rpt;
run;
quit;

 /* Create the Central sales page */
ods html 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 the HTML output file and   */
 /* open the listing destination.    */

ods html close;
ods listing;
```

CHAPTER

21

Generating Web Animation with GIFANIM

Developing Web Presentations with the GIFANIM Device

The GIFANIM device enables you to create sequences of images that are displayed 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 it completes the number of iterations that it is configured to run.

You can use graphics options to customize your GIFANIM presentations, as described in “ GOPTIONS for Controlling GIFANIM Presentations” on page 521.

When to Use the GIFANIM Device

The GIFANIM device is useful for slide shows or animations 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 GIFANIM device does not support data tips and drill-down links. If you need to add data tips and drill-down links to your images, use the JAVAMETA device instead. This device generates Web presentations that run in the Metaview

applet, as described in “Developing Web Presentations for the Metaview Applet” on page 531.

Creating an Animated Sequence

To create an animated sequence with the GIFANIM device, you need to ensure that the resulting data stream is constructed properly. The GIFANIM data stream has three parts: header, body, and trailer.

To see an example of a program that uses the GIFANIM device, see “Sample Programs: GIFANIM” on page 522.

Preparing the Header

When creating a new animated GIF data stream, you must issue a GOPTIONS GSFMODE=REPLACE statement before you invoke the first SAS/GRAPH procedure. The device then constructs a new data stream by writing a valid GIF header and inserting graphical data from the first procedure.

Preparing the Body

After the first procedure has been executed, you must construct the body of the GIF animation. You can think of the body as all of the graphic images between the first and the last images in the sequence. Specify GSFMODE=APPEND in your GOPTIONS statement to suppress the header information and to begin appending graphic data to the current data stream. The GOPTIONS GSFMODE=APPEND statement must appear between the first and second SAS/GRAPH procedures.

Note: If you use BY-group processing on the first graphics procedure to generate multiple graphs, then the output is automatically appended to the same GIF file. Thus, you do not need to specify GSFMODE=APPEND for that first procedure. If you do not use a second graphics procedure to append additional graphs to the GIF file, you do not need to set the GSFMODE= graphics option in the body section of your program. △

Preparing the Trailer

The final step in the GIF animation process is to mark the end of the animation by appending a GIF trailer (“3B”x) to the data stream. The way you do this depends on whether you use BY-group processing in the last procedure:

- If you do not use BY-group processing in the last procedure, set GOPTIONS GEPILOG=“3B”X before the last SAS/GRAPH procedure.
- If you use BY-group processing in the last procedure, do not assign a value to the GEPILOG= option. If you assign a value to GEPILOG=, because the GEPILOG= value is written after each graph in a BY-group, the GIF decoder interprets the first “3B”x as the end of the animation. Instead, use a DATA step to add the trailer to the data stream as follows:

```
data _null_;
   file out recfm=n mod;
   put "3B"x;
run;
```

In the preceding example, OUT is the file reference of the GIF output file.

After the animation is complete, issue a GOPTIONS RESET=ALL statement to prepare for subsequent SAS jobs.

GOPTIONS for Controlling GIFANIM Presentations

You can specify the following options in the GOPTIONS statement to configure Web presentations that are generated with the GIFANIM device.

ITERATION=*iteration-count*
specifies the number of times to repeat the animation loop. The default value of 0 continues the animation indefinitely (until the Web user selects Stop or displays another Web page in the Web browser). Specifying a number greater than 0 repeats the animated sequence for the specified number of iterations, and then continuously displays the last image in the sequence, unless the DISPOSAL= graphics option specifies otherwise.

GSFMODE=REPLACE | APPEND
specifies whether the graphics output should replace the contents of an existing file or be appended to it. In this case, the value of REPLACE specifies that the device is to write a GIF header. Use the GSFMODE= option to specify when to write the GIF header. Specify REPLACE before you generate the first GIF image, and then specify APPEND in a second statement before you generate the rest of the images.

DELAY=*delay-time*
specifies the amount of time that each image is displayed, in hundredths of a second. For example, a value of 1 specifies a delay of 0.01 seconds. The default value is 0.

DISPOSAL=NONE | BACKGROUND | PREVIOUS | UNSPECIFIED
specifies how the image sequence is to be displayed.

NONE
superimposes the images in the sequence, without removing any of them from the screen. This is the default value.

BACKGROUND
restores the background color before displaying the next image.

PREVIOUS
replaces the current image with the previous image before displaying the next image.

UNSPECIFIED
takes no further action before displaying the next image.

TRANSPARENCY | NOTRANSPARENCY
specifies whether the background of the image should be replaced by the background color of the Web browser.

INTERLACED | NONINTERLACED
specifies whether interlacing is to be used as the images are displayed.

Sample Programs: GIFANIM

The following sections provide examples of how to generate animated GIFs:

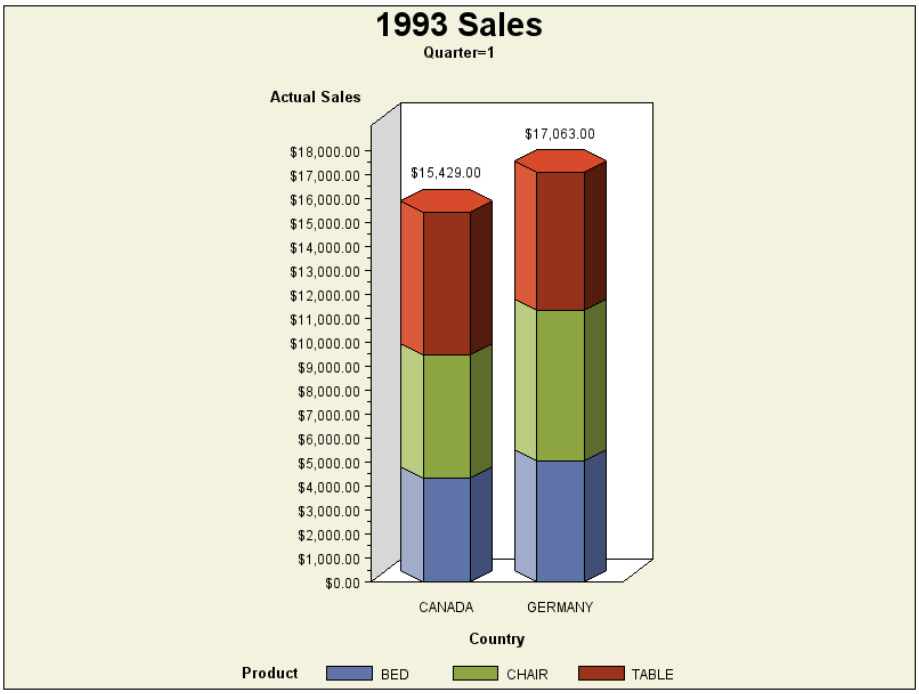
See also example GWBANIMA in the Sample Library.

Creating an Animated GIF with BY-Group Processing

Here is an example that generates an animated GIF from a SAS data set and two invocations of the GCHART procedure, each of which uses BY-group processing. It also generates an HTML file that enables you to view the animation.

Results Shown in a Browser

The following picture shows the first image of the animated GIF only. After a specified time lapse, the chart for each quarter of each of the two years is displayed in turn.



SAS Code

The following is the complete SAS code for this example. Notice the following features:

- The GSFNAME= graphics option specifies the file reference that defines the name of the GIF file that is to be created. In this example, the value of GSFNAME= is specified as gifout, which is defined as the file gifanim1.gif in a FILENAME statement.

- The statement **goptions gsfmode=append;** is included before the second invocation of PROC GCHART so that the output is appended to the same GIF file.
- FILENAME statements specify the filename of the GIF file and the HTML file to be created by the PUT statements.

```
/* Specify output files for the images and the HTML code */
filename gifout  "gifanim1.gif";  /* Image output */
filename htmlout "gifanim1.htm";   /* HTML output */

/* Set the graph style */
ods listing style=harvest;

 /* Delete the previously created graphs before creating new ones */
proc greplay igout=work.gseg nofs;
    delete _all_;
run; quit;

 /* Use gifout to specify the name of the GIF file */
goptions reset=all device=gifanim gsfname=gifout
    gsfmode=replace   /* not necessary when using "BY" */
    delay=150         /* set delay between images      */
    border;

/* Create our data set by extracting information on 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 our data by quarter */
proc sort data=work.qsales;
    by quarter;
run;

/* Generate the first set of graphs */
title1 "1993 Sales";
proc gchart data=work.qsales(where=(year=1993));
    vbar3d country / sumvar=actual subgroup=product sum
    shape=hexagon;
    where product in ("BED" "TABLE" "CHAIR");
    by quarter;
run;
quit;

/* Set the GSFMODE= graphics option to append the subsequent graphs to the file */
goptions gsfmode=append;

/* Generate the second set of graphs */
title1 "1994 Sales";
proc gchart data=work.qsales(where=(year=1994));
    vbar3d country / sumvar=actual subgroup=product sum
    shape=hexagon;
    where product in ("BED" "TABLE" "CHAIR");
    by quarter;
```

```
run;
quit;

/* Write the trailer to the GIF file. Since we */
/* used BY-group processing, use a DATA step */
data _null_;
    file gifout recfm=n mod;
    put "3B"x;
run;

/* Create the HTML file to view the animated GIF */
data _null_ ;
    file htmlout ;
    put "<HTML>";
    put "<HEAD>";
    put "<TITLE> GIFANIM </TITLE>";
    put "</HEAD>";
    put "<BODY>";
    put "<IMG src='gifanim1.gif'>";
    put "</BODY>";
    put "</HTML>";
run;
quit;
```

About the HTML File

The following is the code in the HTML file that is generated by the PUT statements. Instead of embedding PUT statements in a SAS program, you can manually create your own HTML file using an editor of your choice.

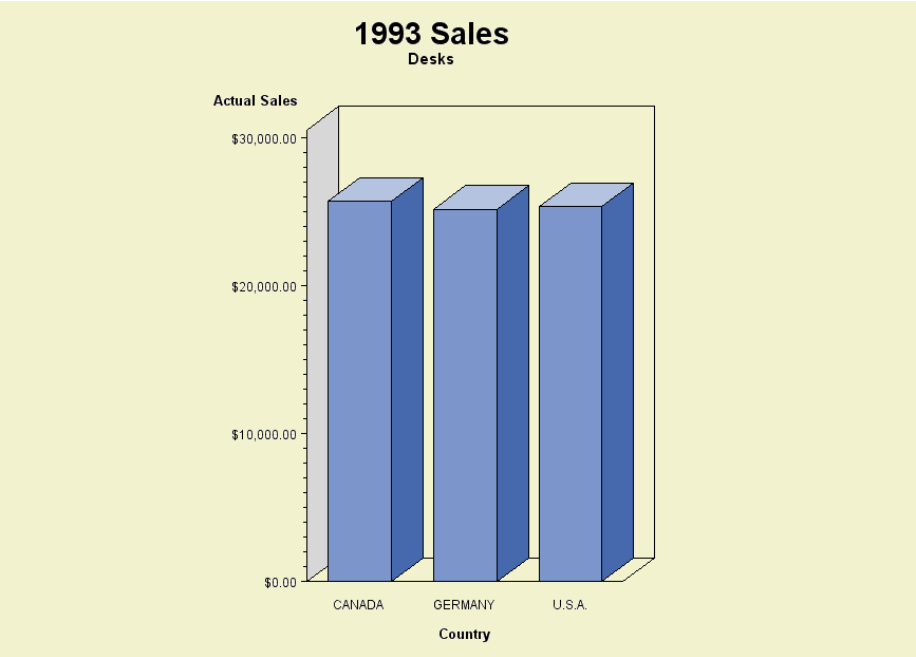
```
<HTML>
<HEAD>
<TITLE> GIFANIM </TITLE>
</HEAD>
<BODY>
<IMG src='gifanim.gif'>
</BODY>
</HTML>
```

Creating an Animated GIF with RUN-Group Processing

This section describes an example that generates an animated GIF using RUN-group processing. RUN-group processing is used to show the 1993 sales data in a specific product order: desks, tables, chairs, sofas, and beds.

Results Shown in a Browser

The following display shows the first image of the animated GIF only.



The animation iterates through the sales data for Canada, Germany, and the U.S.A. The animation waits two seconds between each image and iterates through the animation four times. The animation stops after the fourth iteration and displays the first graph (desks).

SAS Code

The images are generated using the GCHART procedure with RUN-group processing and WHERE clauses to select individual products. Transparency is enabled for each image, so that the Web browser background shows through the unoccupied areas of each image. PUT statements are then used to generate an HTML file that enables you to view the animation with a Web browser. The <BODY> tag in the HTML code specifies a Web browser background color of #F2F2CF, which shows through the image.

You can change the delay between each image by changing the DELAY= graphics option. You can change the number of iterations by changing or removing the ITERATIONS= graphics option. You can also remove the TRANSPARENCY graphics option or change it to NOTRANSPARENCY to see the affect that transparency has on the image.

The SAS code for this example follows.

```
/* Create file references for the output */
filename gifout "gifanim2.gif";   /* Image output */
filename htmout "gifanim2.html";  /* HTML output */

/* Set the graph style */
ods listing style=highcontrast;

 /* Delete the previously created graphs before creating new ones */
proc greplay igout=Work.Gseg nofs;
    delete _all_;
run; quit;

 /* Set graphics options */
```

```
goptions reset=all device=gifanim gsfmode=replace gsfname=gifout noborder
    transparency        /* Let the browser background show through */
    disposal=background /* Restore the background between images */
    delay=200           /* Wait 2 seconds between each image (200 x 0.01s) */
    iterations=4        /* Run the animation four times */
    gsfname=gifout gsfmode=replace;

/* Generate the graphs using RUN-group processing */
title1 "1993 Sales";
proc gchart data=sashelp.prdsale(where=(year=1993));
    title2 "Desks";
    vbar3d country / sumvar=actual;
    where product="DESK";
run;

/* Set the GSFMODE= graphics option to append the remaining graphs */
goptions gsfmode=append;

    title2 "Tables";
    vbar3d country / sumvar=actual;
    where product="TABLE";
run;
    title2 "Chairs";
    vbar3d country / sumvar=actual;
    where product="CHAIR";
run;
    title2 "Sofas";
    vbar3d country / sumvar=actual;
    where product="SOFA";
run;

/* For the last graph, set the GEPILOG= graphics option to */
/* append the trailer */
GOPTIONS GEPILOG="3B"X;

/* Generate the last graph */
    title2 "Beds";
    vbar3d country / sumvar=actual;
    where product="BED";
run;
quit;

/* Create the HTML file to view the animated GIF */
data _null_ ;
    file htmout ;
    put "<HTML>";
    put "<HEAD>";
    put "<TITLE> GIFANIM </TITLE>";
    put "</HEAD>";
    put "<BODY STYLE='background:#F2F2CF'>";
    put "<IMG STYLE='border:none' src='gifanim2.gif'>";
    put "</BODY>";
    put "</HTML>";
run;
```

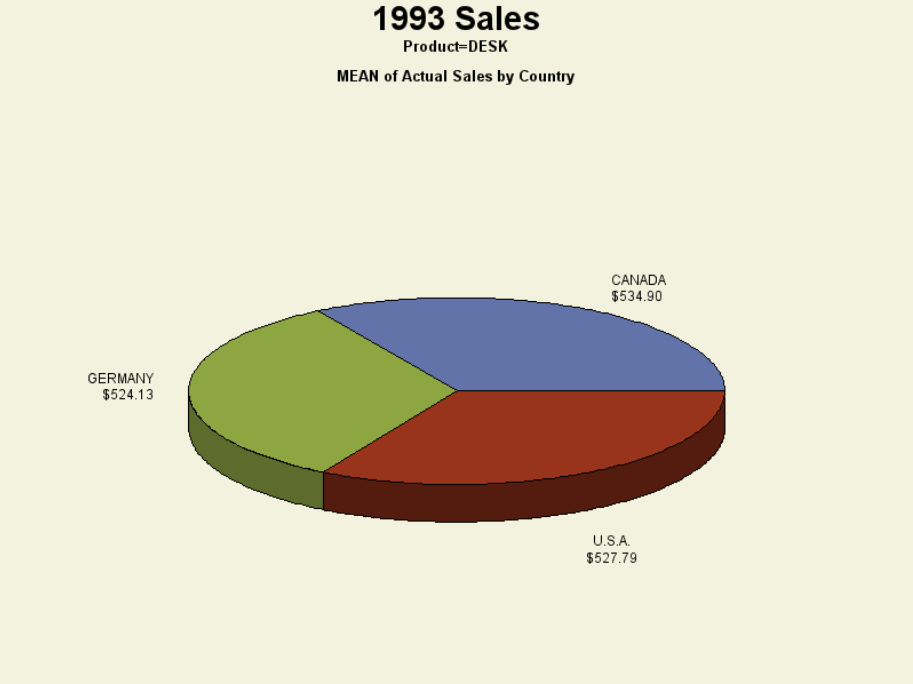
```
quit;
```

Creating an Animated GIF with the GREPLAY Procedure

Here is an example of using the GREPLAY procedure to combine several graphs stored in a catalog into an animated GIF. The GCHART procedure, with BY-group processing, generates the graphs and stores them in a catalog. The GREPLAY procedure is then used to create an animated GIF that plays the graphs in a specific product order: desks, tables, chairs, sofas, and beds.

Results Shown in a Browser

The following display shows the first image of the animated GIF only.



The animation iterates through the 1993 sales data for Canada, Germany, and the U.S.A. The animation waits two seconds between each image and iterates through the animation continuously.

SAS Code

The SAS code for this example follows.

```
/* Create file references for the output */
filename gifout "gifanim3.gif";   /* Image output */
filename htmout "gifanim3.html";  /* HTML output */

/* Create catalog Mygraphs */
libname Mygraphs "C:\";

/* Set the graph style */
ods listing style=harvest;
```

```
 /* Delete the previously created graphs before creating new ones */
proc greplay igout=Mygraphs.Sales nofs;
    delete _all_;
run; quit;

/* Create our data set by sorting sashelp.prdsale by product */
proc sort data=sashelp.prdsale out=Work.sales;
    by product;
run;
quit;

 /* Set graphics options */
goptions reset=all device=gif noborder nodisplay;

/* Generate the graphs */
title1 "1993 Sales";
proc gchart data=work.sales(where=(year=1993)) gout=Mygraphs.Sales;
    pie3d country / sumvar=actual type=mean;
    by product;
run;
quit;

 /* Specify the replay options */
goptions reset=all device=gifanim noborder
    disposal=background /* Restore the background between images */
    delay=200           /* Wait 2 seconds between each image (200 x 0.01s) */
    gsfname=gifout gsfmode=replace;
/* 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:  */
/* GCHART2, GCHART4, GCHART1, GCHART3, and GCHART */
/* Replay the first graph */
proc greplay igout=Mygraphs.Sales nofs;
    replay GCHART2;
run;
quit;

/* Set the GSFMODE= graphics option to append the remaining graphs */
goptions gsfmode=append;

/* Replay the remaining graphs */
proc greplay igout=Mygraphs.Sales nofs;
    replay GCHART4 GCHART1 GCHART3 GCHART;
run;
quit;

/* Write the trailer to the animated GIF file. */
/* Since we used BY-group processing, use a DATA step */
data _null_;
   file gifout recfm=n mod;
   put "3B"x;
run;
```

```
/* Create the HTML file to view the animated GIF */
data _null_ ;
    file htmout ;
    put "<HTML>";
    put "<HEAD>";
    put "<TITLE> GIFANIM </TITLE>";
    put "</HEAD>";
    put "<BODY>";
    put "<IMG STYLE='border:none' src='gifanim3.gif'>";
    put "</BODY>";
    put "</HTML>";
run;
quit;
```

CHAPTER

22

Generating Interactive Metagraphics Output

Developing Web Presentations for the Metaview Applet

The JAVAMETA device driver generates graphs that are stored in metagraphics format and displayed by the SAS Metaview Applet to create interactive graphical Web presentations. The metacodes that comprise the metagraphics format are simple ASCII codes that look like the following:

```
 37    8  106   97  118   97  109  101  116   97   30    0   10    1   13    5
  0    0    0   50    8   32   32   32   32   32   32   32   32   51   18   57
 46   48   48   46   48   48   77   48   68   48   56   48   49   50   48   48
```

You can use a GOPTIONS statement with a DEVICE=JAVAMETA to create metacode output from one or more SAS/GRAPH procedures. When the graph is viewed, the browser passes the metacodes as a parameter to the Metaview applet. The Metaview applet renders the output defined by the metacodes, and displays the interactive graph to the user.

Most SAS/GRAPH procedures that generate GRSEG catalog entries, as well as some other SAS procedures such as PROC GANTT, can be used with the JAVAMETA device to generate metagraphics output. For a list of these procedures, see “Metaview Applet” on page 444.

Interactive features of the Metaview Applet include pan and a play mode for animations. You can add data tips, specify resource files for language translation, specify background colors and text fonts, and drill down to HTML files, metagraphics files, and sets of metacodes. You can also provide a list of selectable drill-down URLs in the pop-up menu. Whereas regular HTML drill-down only allows a single drill-down, the metaview applet allows a selection list of multiple drill-downs per each chart element. For information on these enhancements, see “Enhancing Web Presentations for the Metaview Applet” on page 533.

To generate a Metaview applet presentations, use ODS with the JAVAMETA device driver.

To see examples of programs that generate a Web presentation for the Metaview Applet, see “Example: Generating Metacode Output With the JAVAMETA Driver” on page 536.

Advantages of Using the JAVAMETA Device

The Metaview applet offers these advantages:

- The Metaview applet runs with the Java Virtual Machine that is included with Web browser. It does not require the installation of a Java Plug-in on the user's machine.
- The images produced by the Metaview applet are vector graphics, so the zooming capability provided by the Metaview applet allows the user to zoom in on a graph without degrading the graph's appearance. The zoom control is included by default. You can disable it with the ZOOMCONTROLENABLED= parameter () .
- Compared to raster images (GIF, JPEG, PNG), the Metaview applet offers faster data tips, and the data tips stay up as long as you hold your mouse over them. DEV=JAVAMETA lets you use older versions of JAVA (as compared to DEV=JAVA, which does not allow this functionality).

Using ODS With the JAVAMETA Device

The following steps use ODS to develop a Web presentation for the Metaview Applet. This particular example displays a single graph. The metacodes for that graph are embedded in the body of the HTML output file.

1 Specify the JAVAMETA device driver.

```
goptions reset=all device=javameta;
```

2 Close the LISTING destination to conserve resources.

```
ods listing close;
```

3 Open the HTML destination. You can also specify an HTML filename with the BODY= option. If you do not specify an HTML output filename, the default filename is

```
sashtml.htm
```

. The APPLETLOC= system option specifies the default location of the applet JAR files. If necessary, you can specify another location with the CODEBASE= option in the SAS program.

```
ods html body="filename.htm"
<codebase="location-of-jar-files">;
```

You can enhance your Web presentation by specifying other applet parameters, as described in “Metaview Applet Parameters” on page 534.

4 Include the SAS/GRAPH procedure code.

```
proc gchart data=sashelp.class;
   vbar height / group=age;
run;
quit;
```

5 Close the HTML destination. You must close the HTML destination to generate output. (You may also want to reopen the LISTING destination.)

```
ods html close;
ods listing;
```

Submit the program to generate the HTML output file, which includes the metacodes generated by the JAVAMETA device.

When you view the HTML file in a Web browser, the Metaview applet renders the graph defined by the metacodes.

Enhancing Web Presentations for the Metaview Applet

Programming for the default configuration of the Metaview Applet consists of specifying the JAVAMETA device driver, specifying an HTML output file, and generating a graph. For information on programming for this default configuration, see "Developing Web Presentations for the Metaview Applet" on page 531.

You can enhance the default configuration as follows:

- Specify a non-English resource file and font for Java 1.02 presentations. See "Specifying Non-English Resource Files and Fonts" on page 533.
- Display and configure a zoom control. See the applet parameters that begin with ZOOM, in "Metaview Applet Parameters" on page 534.
- Display and configure a play button to display multiple graphs or to produce an animation effect.
- Set the background color by setting the applet parameter BACKGROUNDCOLOR.
- If you specify an ODS style, do not specify a style that uses a background image (such as Gears or Astronomy) or specify the NOIMAGEPRINT option on the GOPTIONS statement.

Note that you can combine almost all of the available enhancements, including different drill-down modes.

To learn how to specify applet parameters, see "Specifying Applet Parameters Using the ODS PARAMETERS= Statement" on page 536. Reference information on applet parameters is provided in "Metaview Applet Parameters" on page 534.

Specifying Non-English Resource Files and Fonts

The Metaview Applet supports Java 1.02, which is good in that it runs in most browsers. Unfortunately, Java 1.02 does not support the use of resource files and fonts, which would enable the automated use of translated text and localized formats as supported by Java 1.2. To overcome this limitation, the Metaview Applet enables you to name a resource file and a resource font by specifying applet parameters. In this resource file you can hard-code translated versions of the text that the Metaview Applet uses.

Follow these steps to manually translate the text in the Metaview Applet:

1. Specify the LOGRESOURCES parameter in your SAS job, generate the HTML, and view it in a browser. (See "Metaview Applet Parameters" on page 534.) The Metaview Applet will then write its tag/value pairs to the Java console.
2. Copy the tag/value pairs that you want to translate out of the Java console and paste them into your resources file. Then translate those values to your language. You do not need to translate all of the tag/value pairs. The defaults will be used where translations are not provided.

3 Name your resources file **`MVAResources.properties`**.

4 Store your resources file in the same directory as either the HTML output file or the **`sas.graph.metaviewapplet.jar`** file.

5 In the SAS program, remove the LOGRESOURCES parameter specification.

6 If your resources file requires a non-English text font, then specify that font as the value of the parameter RESOURCESFONTNAME. To display this font, your Web audience must have this font installed.

7 Run your program and test your Web output.

For information on specifying applet parameters, see "Specifying Applet Parameters Using the ODS PARAMETERS= Statement" on page 536. For reference information on the Metaview Applet parameters, see "Metaview Applet Parameters" on page 534.

Metaview Applet Parameters

The following parameters may be specified for the Metaview Applet. For information on how to specify these parameters, see "Specifying Applet Parameters Using the ODS PARAMETERS= Statement" on page 536.

BACKGROUNDCOLOR=*color*
specifies the background for the applet as an RGB color in hexadecimal. White is 0xffffff. Red is 0xff0000. If not specified, the background color is 0xd3d3d3 (gray).

Note: This parameter changes only the color of the applet. You can use the CBACK= graphics option on the GOPTIONS statement to set the background color of the graph. △

DATATIPHIGHLIGHTCOLOR=*color*
specifies an RGB color in hexadecimal that is displayed as the outline of the graph element that is displaying its data tip information. The default color is red. This parameter is valid only if the DATATIPSTYLE parameter is set to the value HIGHLIGHT.

DATATIPSTYLE= HIGHLIGHT | STICK | STICK_FIXED
specifies the style of the data tip pop-up window. Values can be:

HIGHLIGHT
causes the data tip to appear above the segment with no connecting line. The border of the graph element is highlighted.

STICK
connects the data tip pop-up window to the graph element with a line. The pop-up window is positioned over the cursor. While the cursor remains in the element, moving the cursor moves the pop-up window and the connecting line.

STICK_FIXED
connects a stationary data tip pop-up window to the graph element with a line drawn into the middle of the graph element.

DEFAULTTARGET=*target-name*
specifies where the browser will display drill-down URLs by default. The value of this parameter can be an HTML target such as _BLANK or the name of a window or frame in the Web presentation. The default value is _BLANK, which displays drill-down URLs in a new browser window. The value of the DEFAULTTARGET parameter is superseded by the optional drill-down tag TARGET.

LOGRESOURCES=TRUE | FALSE
specifying a value of TRUE logs tag/value pairs in the key definition file. The default value is FALSE. The tag value pairs are copied out of the key definition file and modified to create a resource file. The resource file is named **`MVAResources.properties`**, and it enables the Metaview Applet text to be translated to another language. See also the RESOURCESFONTNAME parameter.

METACODES=*codes-or-file-specification*
identifies a text file that contains metagraphics codes, or it provides inline metagraphics codes. The file specification is an absolute or relative URL address.

METACODES1-METACODES*n*=*codes-or-file-specification*
identifies additional metacode specifications when you need to identify more than one file or more than one set of inline metagraphics codes.

METACODESLABEL=*menu-label*
METACODES1LABEL-METACODES*n*LABEL=*menu-label*
names the text labels that are used to identify the graphs specified in the METACODES and METACODES*n* parameters. If specified, there should be as many METACODESLABEL parameters as there are METACODES*n* parameters. Always specify METACODESLABEL parameters in sequential order (METACODESLABEL, METACODES1LABEL, METACODES2LABEL, and so on). The applet displays the labels in an embedded graph-selection control.

RESOURCESFONTNAME=*font-name*
specifies the name of the font family that is used to display the resource values in a user-defined resource file. This allows the Metaview Applet, which is Java 1.02 compliant, to emulate the language translation capabilities of Java 1.2. The applet first tries to use the specified *font-name*, then it tries to use the SansSerif font, then it tries to use the Serif font, then it uses the first font that is returned by the Java.Awt.Toolkit. The first font that is found is the font that is used. See also the LOGRESOURCES parameter.

ZOOMCONTROLENABLED=TRUE | FALSE
displays the embedded zoom control under the graph. The default is TRUE. Specifying a value of FALSE suppresses the display of the zoom control.

Unless you choose to suppress it, the Metaview applet always displays a zoom control which allows a user to zoom in on and out of the image. To suppress the zoom control, specify ZOOMCONTROLENABLED=FALSE in the ODS statement, as follows:

```
ods html body="ncpop.htm"
parameters=("ATATIPSTYLE"="STICK"
"ZOOMCONTROLENABLED"="FALSE");
```

ZOOMCONTROLMIN=*minimum-percentage*
specifies a new lower limit for the zoom feature. The default value is 25 percent of initial size. Valid values range from 1 to 99.

ZOOMCONTROLMAX=*maximum-percentage*
specifies a new upper limit for the zoom feature. The default value is 500 percent of initial size. Valid values range from 100 to 25000.

Specifying Applet Parameters Using the ODS PARAMETERS= Statement

You can control the initial appearance of your Web output and configure aspects of the applet's user interface by specifying applet parameters. The applet parameters are generally specified as follows in the PARAMETERS= option of the ODS statement.

ODS HTML BODY=*HTML-output-file-specification*

PARAMETERS=(
"*parameter-name1*"="*parameter-value1*"...
"*parameter-nameN*"="*parameter-valueN*");

For example:

```
ods html body="ncpop.htm"
    parameters=("DATATIPSTYLE"="STICK"
         "ZOOMCONTROLENABLED"="FALSE");
```

You can specify any number of parameters in a single PARAMETERS= statement. The parameters can be specified in any order. Blank spaces separate multiple parameter specifications. You can also use multiple PARAMETERS= statements within a given ODS statement. The quotation marks and parentheses are required. Additional quotation marks are required in the specification of certain parameter values.

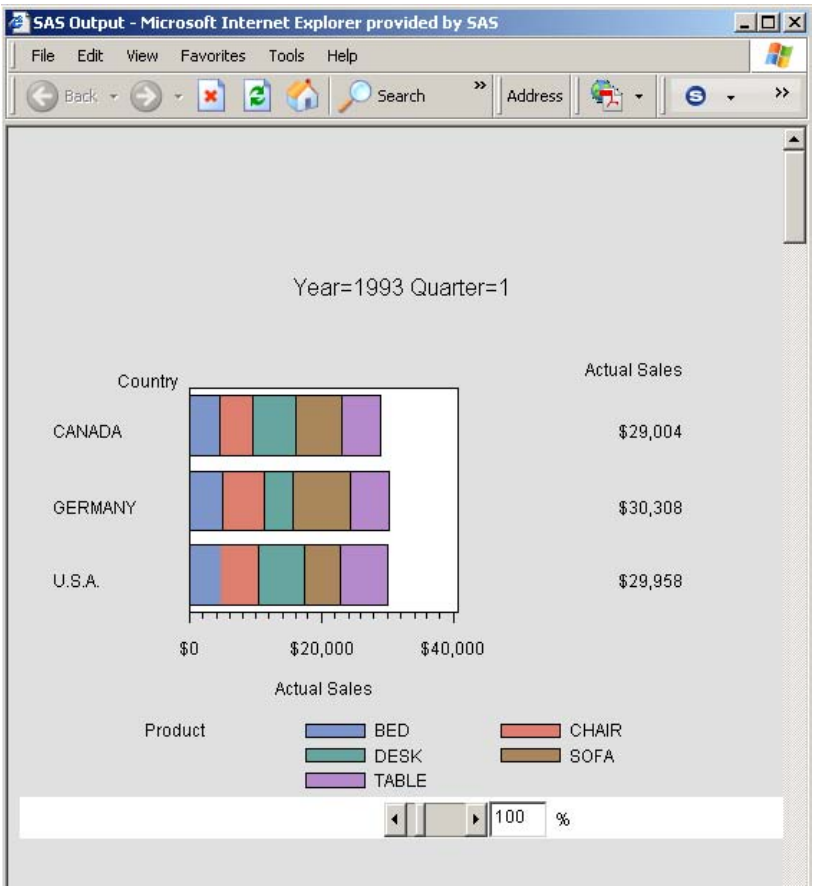
Example: Generating Metacode Output With the JAVAMETA Driver

The following example uses DEVICE=JAVAMETA to generate metcodes to be displayed by the Metaview applet. It uses ODS to create an HTML file, and GOPTIONS DEVICE=JAVAMETA with two instances of PROC GCHART to create graphical output in the form of metacodes. Because both instances of PROC GCHART contain a BY statement, the HTML file created by ODS contains multiple invocations of the applet—one invocation for each value of the BY statement for each procedure (eight invocations in all). The metacodes produced by PROC GCHART are passed to the applet as a parameter.

When you use DEVICE=JAVAMETA with ODS, only one graph can be passed to an instance of the Metaview applet at a time. ODS generates a separate invocation of the Metaview applet for each SAS/GRAPH procedure that it runs. And, if a procedure includes BY GROUP processing, then it generates another separate invocation of the Metaview applet for each BY-group chart. In sum, Metaview applet presentations generated by ODS never contain a slider page control or drop-down list graph control to allow a user to select which graph is to be displayed. Although an HTML page generated by ODS can contain multiple instances of the Metaview applet, each instance can display one picture only, and a user must scroll the HTML page to see all the pictures.

Each GCHART procedure in this example includes a BY statement to display the results of each of the four quarters of the year. Consequently, ODS generates eight separate invocations of the Metaview applet, only the first of which is shown here. A user would have to scroll the page in the browser to see all four quarters displayed.

Notice the zoom control at the bottom of the image. Because the image is displayed by the Metaview, the run-time option is available to the user to control the magnification of the chart. If you want to place multiple graphs in a single metaview applet so that you can use the slider page control or the play/pause buttons, you must script out your own HTML with the PUT statement rather than using ODS.



The following is the complete SAS code to generate a Web presentation. The HTML file is created using ODS HTML.The statement GOPTIONS DEVICE=JAVAMETA causes PROC GCHART to produce metacodes which are embedded in the HTML file produced by ODS and passed to the Metaview applet as parameters.

```
proc sort data= sashelp.prdsale out=prdsummary;
    by year quarter;
run;

goptions reset=all device=javameta
    ftext="Trebuchet" htext=1.5 hby=2;

ods listing close;
ods html;

proc gchart data=prdsummary;
    by year quarter;
    hbar country / sumvar=actual subgroup=product sum;
run;
quit;

ods html close;
```

CHAPTER 23

Generating Web Output with the Annotate Facility

Overview of Generating Web Output with the Annotate Facility

You can use the Annotate facility to enhance your Web presentation, or you can generate an entire Web presentation using Annotate and the GANNO procedure. In either case you can use the Annotate facility to generate drill-down presentations with the GIF, JPEG, or PNG device driver.

Note: You can also use the Annotate facility to enhance output from the ACTXIMG driver, but the ACTXIMG driver does not support the GANNO procedure. △

Note that your graph may conceal your annotations unless your annotations are specified with the value WHEN=A. Specifying this option causes the annotations to be displayed after the graph, so that they will not be occluded. This is particularly important for interactive presentations, where the back wall of the graph may be made visible by default.

Note also that annotations disappear when the Web user selects another graph type. The annotations reappear when the Web user selects the Refresh button in the Web browser.

To learn how to use Annotate data sets to generate drill-down Web presentations, see "Generating Web Output with the Annotate Facility" on page 539.

Reference information on generating and applying Annotate data sets is provided in Chapter 30, "Annotate Dictionary," on page 667. Usage information is provided in Chapter 29, "Using Annotate Data Sets," on page 641. For information on the GANNO procedure, see Chapter 33, "The GANNO Procedure," on page 913.

Generating Web Output with the Annotate Facility

You can use the Annotate facility to generate drill-down Web presentations in two ways: you can use PROC GANNO and an Annotate data set as the sole basis of a drill-down presentation, or you can apply an Annotate data set to add drill-down functionality to a Web presentation that is generated with the ACTXIMG, GIF, JPEG, or PNG device driver.

When to Use PROC GANNO to Generate Web Output

You can use ODS, the GANNO procedure, an Annotate data set, and a 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 JPEG image, with accompanying text, and a selectable label containing the text "Click Here". Larger presentations with multiple drill-down links are also entirely feasible.

To generate a drill-down graph with PROC GANNO, see "Generating Web Links with the Annotate Facility" on page 540.

When to Apply Annotate Data Sets to Web Output

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 ACTXIMG, GIF, JPEG, or PNG 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 drill-down Web presentation.
- 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 use the Annotate facility to add drill-down links to a Web presentation, see "Generating Web Links with the Annotate Facility" on page 540.

Generating Web 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 29.4 on page 647. To generate the Annotate data set, see Chapter 29, "Using Annotate Data Sets," on page 641.
3. Specify the ACTXIMG, GIF, JPEG, or PNG device driver using the DEVICE= option in a GOPTIONS statement.
4. Close the listing destination and open an HTML output file in ODS.

```
ods listing close;
ods html file="annodril.htm"
    style=science;
```

5 Generate a GIF, JPEG, or PNG 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

For an example of creating web output with the GANNO procedure, see Example 4 on page 925.

For examples of applying Annotate data sets to output, see "Examples" on page 658.

CHAPTER

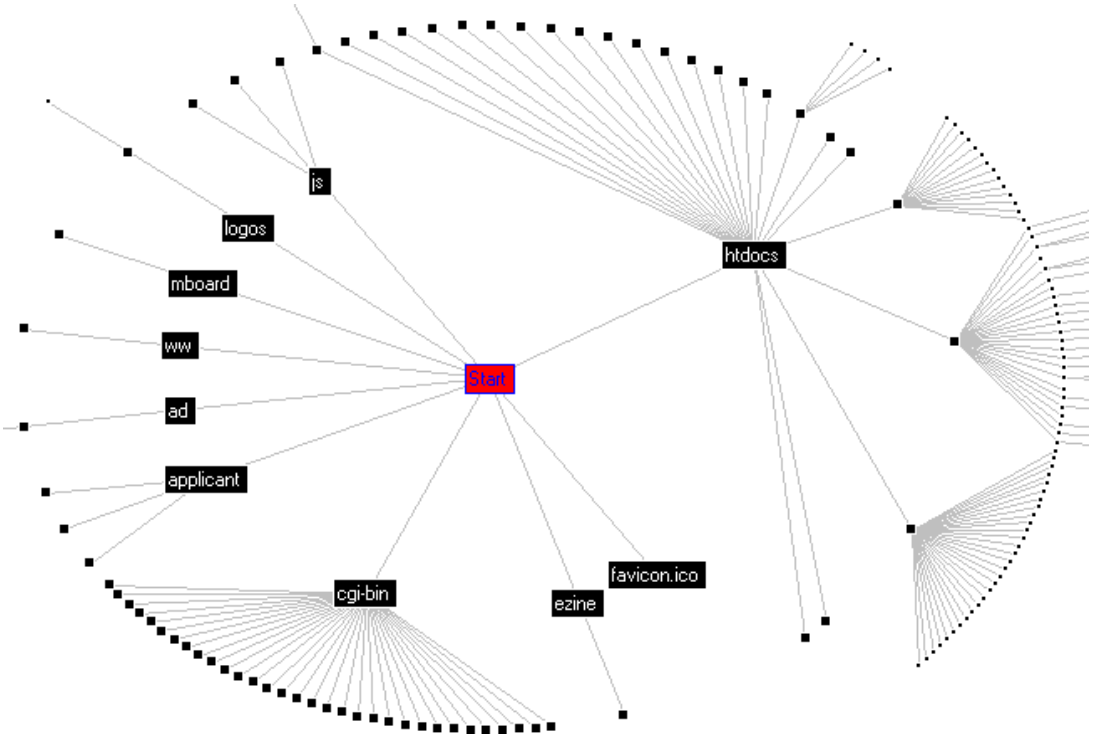
24

Creating Interactive Treeview Diagrams

Creating Treeview Diagrams

The Treeview applet generates node/link diagrams for hierarchical data, with optional fish-eye distortion that highlights the central area of interest, as shown in the following figure:

Display 24.1 A Treeview Applet Web Link Diagram



You can scroll across the diagram by selecting off-center nodes or by searching for nodes. Positioning the cursor over a node can display optional data tips. If you then right-click, you access a pop-up menu. The menu enables you to highlight or hide subtrees or drill-down to an optional URL. The menu also enables you to select all nodes, display all previously hidden nodes, reset the view, display applet help, and search for nodes using various search parameters.

SAS/GRAPH programming for the Treeview applet differs from some of the other applets in that it does not use ODS, a device driver specification, or a SAS/GRAPH procedure. Instead, the DS2TREE macro references data sets to generate and configure an HTML output file that runs the Treeview applet.

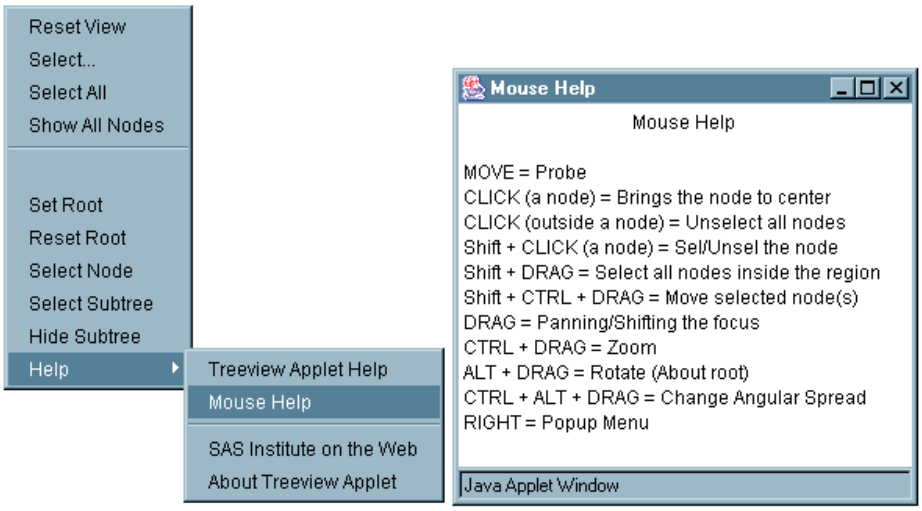
When to Use the Treeview Applet

The Treeview applet is well-suited for the illustration of hierarchical data sets. The fish-eye distortion factor, coupled with extensive node selection features, means that a single node/link diagram can accommodate large data sets. Applet parameters can be set to specify the layout of the diagram. You specify a starting node, and then you specify how the other nodes are to be drawn in relation to that node. The resulting diagram can be as complex as the Web link diagram in Display 24.1 on page 544, or as simple as an organizational tree for a department in a corporation.

If you need a higher degree of configurability to illustrate weighted relationships between the nodes and links in your diagram, then the Constellation applet might be a better choice than the Treeview applet, as described in “Creating Constellation Diagrams” on page 553.

Interactivity Enabled by the Treeview Applet

The following picture shows the pop-up menu that a user can invoke by right-clicking a Treeview diagram in a browser. The picture shows all the options that are available for interacting with the diagram. For a description of these options, right-click on any Treeview diagram and select **`Treeview Applet Help`** from the pop-up menu.



Programming with the DS2TREE Macro for the Treeview Applet

The DS2TREE macro generates HTML output files for the Treeview applet. Macro arguments enable you to generate and format an HTML file and to customize the appearance of your node/link diagram.

Follow the steps shown in the following code to generate a Web presentation that runs the Treeview applet. (Note that the ODS LISTING destination must be open in running the macro.)

```
/* 1. Define the name and storage location of the HTML output file */
/*    and the location of the jar files.                            */
%let htmlfile = your_path_and_filename.htm;
%let jarfiles = http://your_path_to_archive;

/* 2. Define a data set that contains parent-child relationships. */
data myorg;
input name $ empno mgrno deptname $22. deptcode $;
cards;
Peter      2620   1420   Documentation            DOC
Linda      6915   1420   Research & Development   R&D
Maria      1320   1420   Legal                    LGL
Vince      1420   1750   Executive                EXE
Jim        6710   6915   Quality Assurance        QA
Nancy     22560   6915   Quality Assurance        QA
Patrick   28470   6915   Quality Assurance        QA
Elsa      33075   6915   Development              DEV
Clement   22010   6915   Development              DEV
Murielle   3020   6915   Development              DEV
```

```
David     11610   6915   Research                RES
;
run;

/* 3. Specify titles and footnotes: (optional). */
title1 'Organizational Chart';
footnote1 'To display the department name, place the cursor over a node.';
footnote2 'To rotate the chart, click and drag a node.';

/* 4. Run the DS2TREE macro. */
/* You must change the CODEBASE= argument (using either http://      */
/* or a directory path such as C:/) to specify the location of your  */
/* sas.graph.treeview.jar file and its associated jar files          */
/* (sas.graph.nld.jar, sas.graph.j2d.jar). See the CODEBASE= argument in:   */
/* Arguments for the APPLET Tag*/
/* Make sure that the ods listing destination is open. */
ods listing;
%ds2tree(ndata=myorg,         /* data sets and files */
         codebase=&jarfiles,
         xmltype=inline,
         htmlfile=&htmlfile,
         nid=empno,           /* roles of variables */
         nparent=mgrno,
         ntip=deptname,
         nlabel=name,
         height=500,          /* appearance */
         width=600,
         tcolor=navy,
         fcolor=black);
```

Display the resulting HTML file in a Web browser to run the Treeview applet and display the node/link diagram.

The preceding example shows how the arguments of the DS2TREE macro identify a data set and specify how the variables in that data set are to be interpreted to generate the diagram. Appearance arguments define the size of the diagram and the color of the text in the title and footnotes.

For information on generating more complex diagrams for the Treeview applet, see "Enhancing Presentations for the Treeview Applet" on page 546.

For definitions of all DS2TREE macro arguments, see "DS2TREE Macro Arguments" on page 547.

Enhancing Presentations for the Treeview Applet

The Treeview applet displays interactive node/link diagrams. The diagrams are generated in SAS using a hierarchical data set and the DS2TREE macro, as described in "Programming with the DS2TREE Macro for the Treeview Applet" on page 545.

To enhance Treeview applet presentations, specify additional arguments for the DS2TREE macro. The following table describes some of the available enhancements and identifies the DS2TREE arguments that implement them. For a complete list of macro arguments, see "Macro Arguments" on page 569.

Table 24.1 Treeview Applet Enhancements

Enhancement	DS2TREE Argument
Specify a stylesheet to format your HTML output file.	SSFILE, SSFREF, SSHREF, SSMEDIA, SSREL, SSREV, SSTITLE, SSTYPE
Specify dash patterns for link lines.	LSTIP, LSTIPFAC
Specify a background color, image, or drill-down URL.	IBACKPOS, IBACKLOC, IBACKURL
Add pop-up data tips to nodes.	NTIP, TIPS
Add drill-down URLs to nodes.	NURL
Specify an action for the pull-down menu.	ACTION, NACTION
Change the amount of fisheye distortion.	FACTOR, FISHEYE
Determine layout of diagram.	SPREAD, TREEDIR, TREESPAN

DS2TREE Macro Arguments

The arguments of the DS2TREE macro specify the configuration of the HTML output file, the location of the data that is used to generate the diagram, and the configuration of the applet's interactive features.

The DS2TREE macro uses the following syntax:

%DS2TREE(*argument1=value1*, *argument2=value2*, ...);

The arguments of the DS2TREE macro can be divided into the following categories:

- "Arguments for the APPLET Tag" on page 569. The CODEBASE argument is required.
- "DS2TREE and DS2CONST Arguments for Data Definition" on page 571. For DS2TREE the arguments NDATA and NID are required.
- "Arguments for Generating HTML and XML Files" on page 578.
- "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579.
- "Arguments for Page Formatting" on page 585.
- "Arguments for Stylesheets" on page 587.
- "Arguments for the SAS TITLE and FOOTNOTE Tags" on page 589.
- "Arguments for Character Transcoding" on page 593.

Sample Programs: Treeview Macro

The following sample programs generate Treeview diagrams:

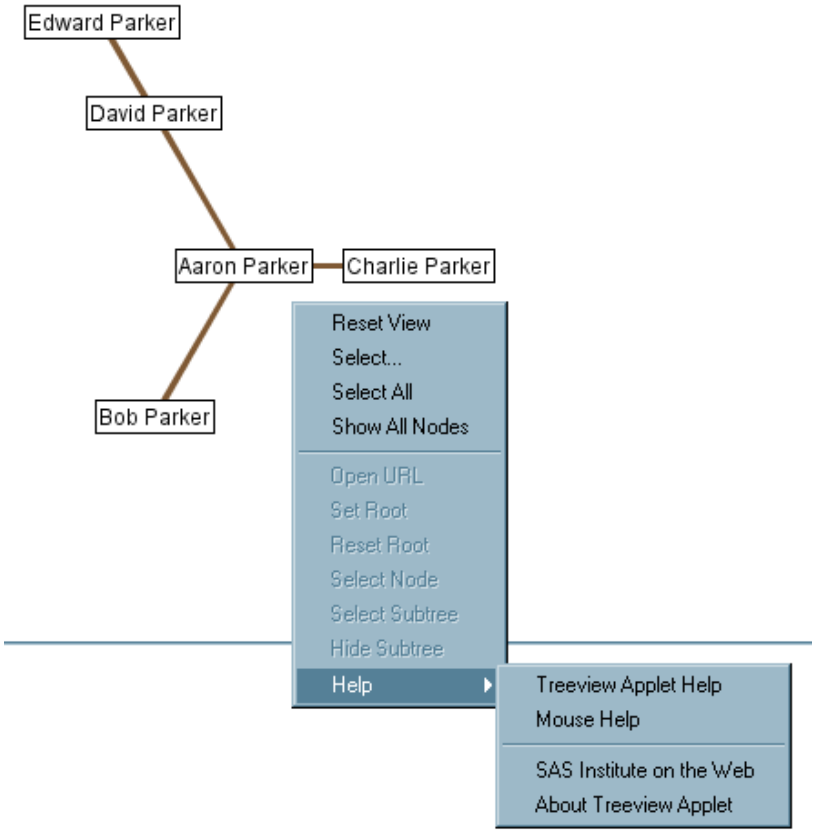
- "Sample Treeview with XML Embedded in the HTML File" on page 547
- "Sample Treeview with XML Written to an External File" on page 549
- "Treeview with Hotspots" on page 550.

Sample Treeview with XML Embedded in the HTML File

This sample program generates a very simple Treeview diagram.

Results Shown in a Browser

The following is the Treeview diagram that is generated by the sample code. Notice the pop-up menu. Because the diagram is displayed by the Treeview applet, it is not just a static picture. A user can manipulate the diagram, for example, by bringing selected nodes to the center, spreading out the nodes, and searching for nodes.



SAS Code

The following is the complete SAS code used to generate the Treeview diagram from a SAS data set. Note the following:

- The parameter HTMLFILE= specifies the complete path and name of the HTML file to be created by the DS2TREE macro. If you want to run this sample, then change the values of HTMLFILE and CODEBASE to the locations that you want to use.
- The parameter XMLTYPE=INLINE tells the DS2TREE macro that the XML it generates from the SAS data set should be included inline in the HTML file.
- The parameter CUTOFF=1 specifies that every node on the graph be labeled. Use this parameter to suppress node labels for diagrams with numerous nodes.

```
data father_and_sons;
input id $8. name $15. father $8.;
cards;
aaron   Aaron Parker
bob     Bob Parker     aaron
charlie Charlie Parker aaron
```

```
david   David Parker   aaron
edward  Edward Parker  david
;
run;

/* make sure ods listing is open when running macro */
ods listing;
 /* run the macro */
%ds2tree(ndata=father_and_sons,  /* data set */
         /* specify complete url if jar files are not in same directory as
               html file */
         codebase=http://your_path_to_archive,
         xmltype=inline,
         htmlfile=your_path_and_filename.htm,
         nid=id,        /* use this variable as the id */
         cutoff=1,      /* display the name on every node */
         nparent=father,/* this identifies the parent of each node */
         nlabel=name,   /* display this on each node */
         height=400,
         width=400,
         tcolor=navy,
         fcolor=black);
```

Sample Treeview with XML Written to an External File

This sample program generates the same Treeview as the previous example, "Sample Treeview with XML Embedded in the HTML File" on page 547, with the difference that the XML is written to an external file instead of being embedded in the HTML file.

SAS Code

The following is the complete SAS code to generate the Treeview diagram from a SAS data set. Note the following:

- The parameter HTMLFILE= specifies the complete path and name of the HTML file to be created by the DS2TREE macro. If you want to run this sample, then change the value of HTMLFILE to something that makes sense for you.
- The parameter XMLTYPE=EXTERNAL tells the DS2TREE that the XML it generates from the SAS data set should be written to an external file.
- The parameter XMLFILE= specifies the path and file name of the XML file to be created.
- The parameter XMLURL= specifies how the XML file is to be addressed from within the HTML file.
- The parameter CUTOFF=1 specifies that every node on the graph be labeled. Use this parameter with a value between 0 and 1 to suppress node labels for diagrams with numerous nodes.

```
data father_and_sons;
input id $8. name $15. father $8.;
cards;
aaron   Aaron Parker
bob     Bob Parker     aaron
charlie Charlie Parker aaron
```

```
david   David Parker   aaron
edward  Edward Parker  david
;
run;
goptions reset=all;
/* make sure ods listing is open when running macro */
ods listing;
 /* run the macro */
%ds2tree(ndata=father_and_sons,  /* data set */
         codebase=http://your_path_to_archive,
         htmlfile=your_path_and_filename.htm,
         xmltype=external,
         makexml=y,
         xmlurl=http://www.xtz.com/weboutput_treeview2_sample.xml,
         xmlfile=u:/public/weboutput_treeview2_sample.xml,
         nid=id,        /* as the id, use this variable specified here */
         cutoff=1,      /* display the name on every node */
         nparent=father,/* this identifies the parent of each node */
         nlabel=name,   /* display the value of this variable on each node */
         height=400,
         width=400,
         tcolor=navy,
         fcolor=black);
```

Treeview with Hotspots

This sample program generates the same Treeview as the previous example, "Sample Treeview with XML Embedded in the HTML File" on page 547, with the difference that a node is associated with a URL and can be activated by a user double-clicking the node.

SAS Code

The following is the complete SAS code to generate the Treeview diagram from a SAS data set. Note the following:

- The parameter NURL= specifies the URL to be opened when the corresponding node is double-clicked.
- The parameter DRILTARG=_TOP specifies that the HTML file is to be opened in the same window as the Treeview diagram instead of in a new window, as is the default.

```
data father_and_sons;
input id $8. name $15. father $8. url $30.;
cards;
aaron   Aaron Parker           http://www.xyz.com/index.html
bob     Bob Parker     aaron   http://www.xyz.com/index.html
charlie Charlie Parker aaron   http://www.xyz.com/index.html
david   David Parker   aaron   http://www.xyz.com/index.html
edward  Edward Parker  david   http://www.xyz.com/index.html
;
run;
/* make sure ods listing is open when running macro */
ods listing;
/* run the macro */
```

```
%ds2tree(ndata=father_and_sons,  /* data set */
         /* specify complete url if jar files are not in same directory as html
              file */
         codebase=http://your_path_to_archive,
         xmltype=inline,
         htmlfile=your_path_and_filename.htm,
         nid=id,        /* as the id, use the variable specified here  */
         cutoff=1,      /* display the name on every node */
         nparent=father,/* this identifies the parent of each node */
         nlabel=name,   /* display the value of this variable on each node */
         height=400,
         width=400,
         tcolor=navy,
         fcolor=black,
         nurl=url,
         driltarg=_top );
```

CHAPTER

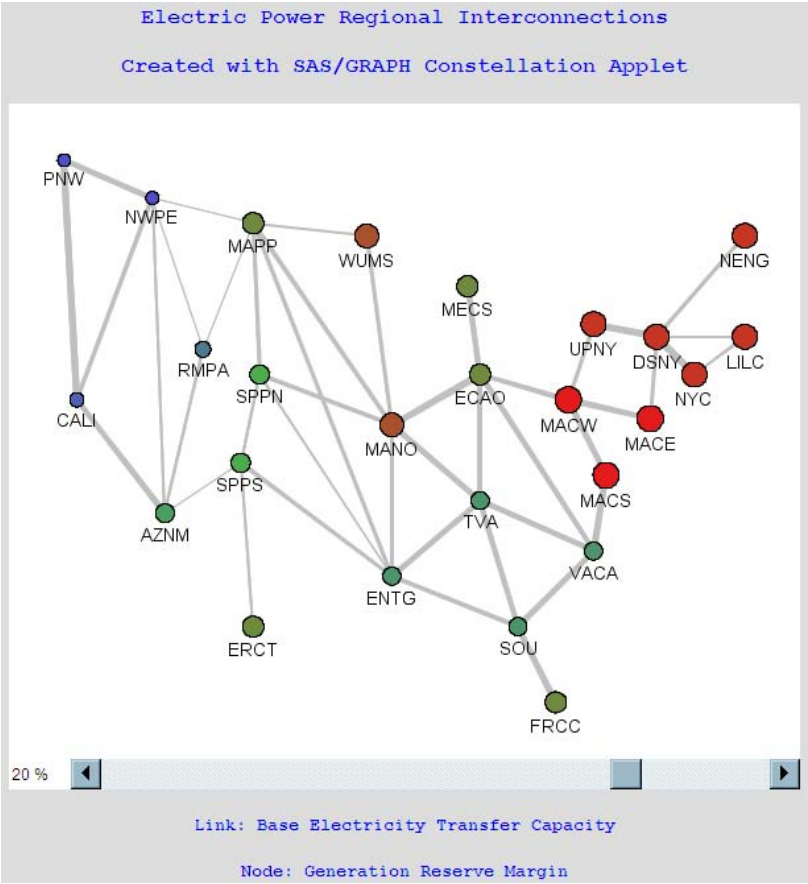
25

Creating Interactive Constellation Diagrams

Creating Constellation Diagrams

The Constellation Applet provides interactivity for node/link diagrams that illustrate data that is associative, hierarchical, or requires an arc list. Node and link color and size can be associated with specified data values.

Display 25.1 A Constellation Diagram



Interactive features of the Constellation Applet include pop-up data tips for links and nodes, subsetting of links via an embedded scroll bar, pan and zoom, and several node and link selection modes. You can define drill-down URLs for nodes, specify menu text for the drill-down action, insert a background image, and specify a drill-down URL for the background image, among other enhancements. You can also specify your own JavaScript methods to define responses to drill-down actions.

The Constellation Applet, like the Treeview applet, differs from the other applets in that the diagrams that they display are not generated by SAS/GRAPH procedures. The DS2CONST macro generates and formats an HTML output file, and specifies the appearance and behavior of the node/link diagram based on values in a data set.

When to Use the Constellation Applet

The Constellation Applet is best used to illustrate relationships between links and nodes, which can be shown in affinity, sequence, and Web-click path diagrams, for example. Colors, link line widths, and link directional indicators can be specified to illustrate relationships. Pop-up data tips can be specified for nodes and links, along with drill-down URLs for nodes and for an optional background image. For diagrams that illustrate associative data, an embedded scroll bar subsets the data in the diagram dynamically.

The Constellation Applet can be used to display hierarchical data, but so can the Treeview Applet, which should also be considered for hierarchical diagrams such as

organizational trees, because of its unique layout capabilities. For information on the Treeview Applet, see "Creating Treeview Diagrams" on page 543.

Programming with the DS2CONST Macro for the Constellation Applet

The DS2CONST macro enables you to generate complete Web presentations for the Constellation Applet. The macro has a large number of arguments that you can use to generate and format an HTML output file, configure the diagram, and describe how data sets and variables are to be applied to the diagram.

The macro arguments are structured so that you can associate a variable with an aspect of the diagram. The values of the variable are then used for that part of the diagram. For example, the NLABEL argument specifies the name of the variable whose values define the text labels that are to be applied to the nodes. Other arguments provide default values that are used when no variable value is provided.

Descriptions of all of the arguments of the DS2CONST macro are provided in "DS2CONST Macro Arguments" on page 560.

Run the following code to use the DS2CONST macro to generate the Web presentation for the Constellation Applet shown in the picture above. (Note that the ODS LISTING destination must be open in running the macro.)

```
/*--- Define name and storage location of the HTML output file,
      and the location of the jar files. */
%let htmlfile = your_path_and_filename.htm;
%let jarfiles = .; /* jar file is in same directory as this html file */
%let archive  = constapp.jar;
%let lib      = WORK; /* put everything in WORK library */

/*--- Define the node names and locations. */
data &lib..regions;
length regionName $80            /* Node text label */
       regionId $4               /* Node identifying string */
       xLoc yLoc 8;              /* Pixel position of node */
input regionID xLoc yLoc reserve RegionName $ &;
cards;
PNW 30 30 8.5 Western Systems Coordinating Council - Pacific Northwest
NWPE 100 60 8.5 Western Systems Coordinating Council - Northwest Power Pool East
CALI 40 220 9.5 Western Systems Coordinating Council - California
RMPA 140 180 10.8 Western Systems Coordinating Council - Rocky Mountain Power Area
AZNM 110 310 12.9 Western Systems Coordinating Council - AZNMSNV
MAPP 180 80 15 Mid-continent Area Power Pool
SPPN 185 200 13.6 Southwest Power Pool - North
SPPS 170 270 13.6 Southwest Power Pool - South
ERCT 180 400 15 Electric Reliability Council of Texas
WUMS 270 90 17 Wisconsin - Upper Michigan
MANO 290 240 17 Mid-America Interconnected Network - South
ENTG 290 360 12.4 Entergy
MECS 350 130 15 Michigan Electric Coordination System
ECAO 360 200 15 East Central Area Reliability Coordination Agreement - South
TVA 360 300 12.4 Tennessee Valley Authority
SOU 390 400 12.4 Southern Company
FRCC 420 460 15 Florida Reliability Coordinating Council
VACA 450 340 12.4 Virginia and Carolinas
MACS 460 280 19 Mid-Atlantic Area Council - South
MACE 495 235 19 Mid-Atlantic Area Council - East
```

```
MACW 430 220 19 Mid-Atlantic Area Council - West
UPNY 450 160 18 Upstate New York
DSNY 500 170 18 Downstate New York
NYC 530 200 18 New York City
LILC 570 170 18 Long Island Lighting Company
NENG 570 90 18 New England Power Pool
;
run;

/*--- Define the node connections. */
data &lib..links;
length from to $4 ltip $12;
format capacity comma.;
input  from to capacity;
ltip = left(put(capacity, comma.) || " MW");
if      capacity <  500 then width = 1;
else if capacity < 1000 then width = 2;
else if capacity < 2000 then width = 3;
else if capacity < 3000 then width = 4;
else                         width = 5;
cards;
MECS ECAO 2250
ECAO MECS 2250
ECAO MACW 2957
ECAO MANO 1655
ECAO TVA 1890
ECAO VACA 2334
ERCT SPPS 635
MACE MACW 1500
MACE DSNY 1130
MACS MACW 1800
MACS VACA 3075
MACW ECAO 2612
MACW MACE 3368
MACW MACS 3075
MACW UPNY 481
MANO ECAO 3033
MANO WUMS 608
MANO MAPP 531
MANO SPPN 1191
MANO TVA 2207
MANO ENTG 1245
WUMS MANO 1080
WUMS MAPP 676
MAPP MANO 1150
MAPP WUMS 324
MAPP SPPN 1172
MAPP ENTG 1000
MAPP NWPE 150
MAPP RMPA 233
NENG DSNY 1425
UPNY MACW 1418
UPNY DSNY 3750
DSNY LILC 788
```

```
DSNY MACE 308
DSNY NENG 1125
DSNY UPNY 3750
DSNY NYC 3750
NYC LILC 788
NYC DSNY 3750
LILC DSNY 938
LILC NYC 788
SPPN MANO 1228
SPPN MAPP 891
SPPN SPPS 525
SPPN ENTG 636
SPPS ERCT 569
SPPS ENTG 636
SPPS SPPN 900
SPPS AZNM 315
SPPS ENTG 1200
ENTG SOU 1136
ENTG TVA 1278
ENTG MANO 1399
ENTG SPPS 292
ENTG SPPN 292
ENTG MAPP 856
SOU FRCC 4516
SOU TVA 1810
SOU VACA 1346
SOU ENTG 1902
FRCC SOU 21
TVA ECAO 2235
TVA MANO 2331
TVA SOU 2052
TVA ENTG 2153
TVA VACA 2261
VACA ECAO 2822
VACA MACS 2794
VACA SOU 3042
VACA TVA 2240
CALI PNW 4922
CALI AZNM 0
CALI NWPE 1184
PNW CALI 5903
PNW NWPE 1050
RMPA MAPP 233
RMPA AZNM 518
RMPA NWPE 413
NWPE RMPA 413
NWPE CALI 1574
NWPE MAPP 113
NWPE AZNM 840
NWPE PNW 2145
AZNM CALI 5663
AZNM NWPE 638
AZNM RMPA 518
AZNM SPPS 315
```

```
;
run;

/*--- Make sure ods listing is open when running macro. */
ods listing;

/*--- Set chart title.  */
title1 "Electric Power Regional Interconnections";
title2 "Created with SAS/GRAPH Constellation Applet";
footnote1 "Link: Base Electricity Transfer Capacity";
footnote2 "Node: Generation Reserve Margin";

/*--- Use the DS2CONST macro to generate the chart. */
%ds2const(ndata=&lib..regions,  /* Node parameters */
          ldata=&lib..links,    /* Node linkage parameters */
          datatype=assoc,       /* Size nodes by nvalue var  */
          nvalue=reserve,       /* Var for node sizes */
          nodeshap=circle,      /* Node shape */
          cnode=red,            /* Node fill color */
          colormap=y,           /* Use colormap for link/node colors */
          height=520,           /* Applet window height */
          width=600,            /* Applet window width */
          codebase=&jarfiles,   /* Path to archive file */
          htmlfile=&htmlfile,   /* Output file name */
          openmode=replace,     /* Create a new html file */
          archive=&archive,     /* Java archive file name */
          nid=regionID,         /* Var for node ID string */
          border=y,             /* Enclose diagram */
          fntsize=14,           /* Node label font size */
          fntstyl=plain,        /* Node label font style */
          nlabel=regionID,      /* Var for node label string */
          labels=y,             /* Display node labels */
          linktype=line,        /* Do not show flow direction */
          layout=user,          /* Use nx/ny to position nodes */
          nx=xLoc,              /* x-coordinate of node */
          ny=yLoc,              /* y-coordinate of node */
          lfrom=from,           /* Var for from-node ID */
          lto=to,               /* Var for to-node ID */
          lwidth=width,         /* Var for line widths */
          ntip=RegionName,      /* Var for popup node text */
          ltip=ltip,            /* Var for popup line text */
          center=y,             /* Center chart on the page */
          tcolor=#0000FF,       /* Title text color */
          fcolor=#0000FF,       /* Footnote text color */
          tsize=3,              /* Title text size */
          fsize=2,              /* Footnote text size */
          bgtype=color,         /* Use page background color */
          bg=#DDDDDD,           /* The page background color */
          septype=none);        /* No separator line */
```

Display the resulting HTML file in a Web browser to run the applet and generate the diagram.

Arguments in the DS2CONST macro identify the name of the nodes and links data sets. In the nodes data set, arguments identify a node ID variable and a node label

variable. Other arguments identify the links data set and the variables that define the nodes at the start and end of each link line.

For information on more complex presentations for the Constellation Applet, see "Enhancing Presentations for the Constellation Applet" on page 559.

Enhancing Presentations for the Constellation Applet

The Constellation Applet displays interactive node/link diagrams. These diagrams can show relationships between nodes and links. The Constellation Applet displays affinity, sequence, and ring diagrams that are generated out of arc, associative, or hierarchical data sets. The Constellation Applet provides a number of interactive features by default, as described in "Creating Constellation Diagrams" on page 553.

Enhancements to Constellation Applet presentations are configured in your SAS/GRAPH program by specifying arguments in the DS2CONST macro. The following table lists some of the available enhancements and the DS2CONST arguments that implement them. These enhancements enable you to provide data tips and drill-down URLs for nodes and links, and to increase the visible distinctions between the data values that are associated with the nodes and links.

Table 25.1 Constellation Applet Enhancements

Enhancements	DS2CONST Arguments
Specify link weights and configure a scroll bar that controls the display of links based on weight.	LVALUE, MINLNKWT, SCLNKWT
Lay out the diagram automatically or as specified in a data set.	LAYOUT
Specify a stylesheet to format the HTML output file.	BDCLASS, SEPCLASS, SPCLASS, SSFILE, SSHREF See "Arguments for Stylesheets" on page 587.
Add pop-up data tips to nodes and links.	LTIP, NTIP
Define drill-down URLs for nodes and links.	LURL, NURL
Specify menu option text for a drill-down action.	ACTION, NACTION
Specify a browser window or frame that displays drill-down URLs.	DRILTARG
Add a background color, image, or drill-down URL.	IBACKLOC, IBACKPOS, IBACKURL,
Specify text colors, fonts, styles, and sizes.	NFNTNAME, NSFNTNAM, CTEXT, CATEXT See "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579.
Specify colors for nodes and links.	NCOLVAL, NCOLOR, CNODE, LCOLVAL, LCOLOR, CLINK
Specify dashed link lines.	LSTIP and LSTIPFAC

Note that a number of enhancements apply only to associative data sets when you specify the macro argument DATATYPE=ASSOC. The macro argument definitions identify which features apply only to associative data.

The DS2CONST macro requires you to specify node and link data sets. As an enhancement, you can define a node styles data set that contains style information only. You can use the node styles data set to standardize the appearance of a series of diagrams, among other uses.

Reference information on the arguments of the DS2CONST macro is provided in "DS2CONST Macro Arguments" on page 560.

DS2CONST Macro Arguments

The arguments of the DS2CONST macro specify the configuration of the HTML output file, the location of the data that is used to generate the diagram, and the configuration of the applet's interactive features.

The DS2CONST macro uses the following syntax:

%DS2CONST(*argument1=value1*, *argument2=value2*, ...);

The arguments of the DS2CONST macro can be divided into the following categories:

- "Arguments for the APPLET Tag" on page 569. The CODEBASE argument is required.
- "DS2TREE and DS2CONST Arguments for Data Definition" on page 571. For DS2CONST the arguments NDATA, NID, LDATA, and LTO are required.
- "Arguments for Generating HTML and XML Files" on page 578.
- "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579.
- "Arguments for Page Formatting" on page 585.
- "Arguments for Stylesheets" on page 587.
- "Arguments for the SAS TITLE and FOOTNOTE Tags" on page 589.
- "Arguments for Character Transcoding" on page 593.

Sample Programs: Constellation Macro

The following sample programs generate these kinds of Constellation diagrams:

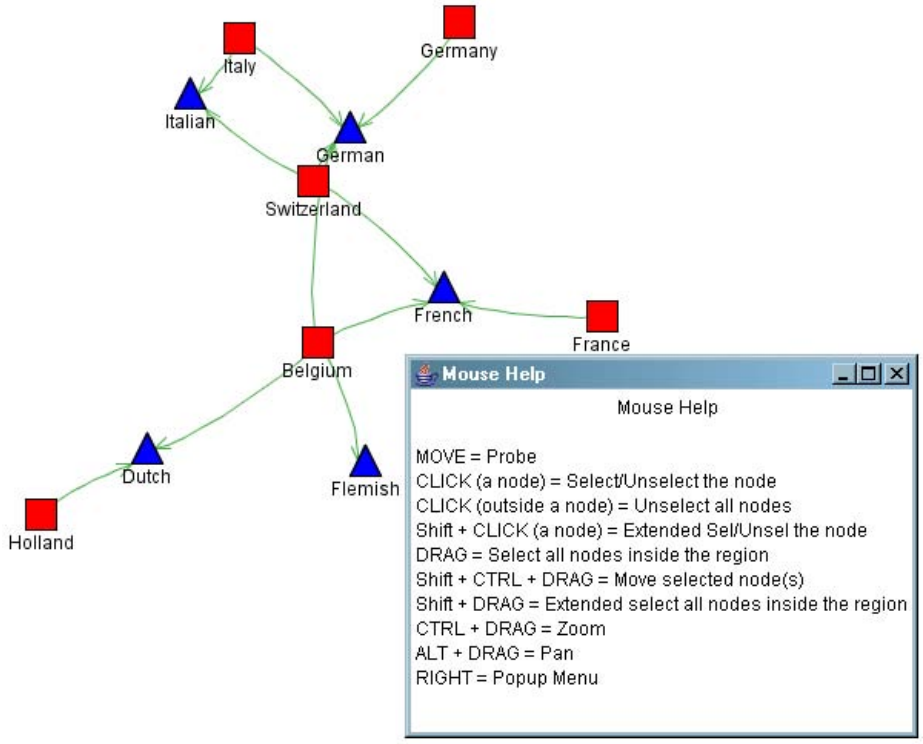
- "Constellation Chart with DATATYPE=ARCS" on page 560
- "Constellation Chart with DATATYPE=ASSOC" on page 562
- "Constellation Chart with XML Written to an External File" on page 564
- "Constellation Chart with Hotspots" on page 566.

Constellation Chart with DATATYPE=ARCS

This sample program generates a very simple Constellation diagram. It displays a number of countries and the languages spoken in those countries.

Results Shown in a Browser

The following is the Constellation diagram that is generated by the sample code shown below. Notice the help window. Because the diagram is displayed by the Constellation applet, it is not just a static picture. A user can manipulate the diagram, for example, by moving nodes and searching for nodes. The Mouse Help window in the following diagram documents for the user what interactivity is available (right-click a diagram to invoke the window).



SAS Code

The following is the complete SAS code used to generate a Constellation diagram from a SAS data set. Notice the following:

- The parameter HTMLFILE= specifies the complete path and name of the HTML file to be created by the DS2CONST macro. If you want to run this sample, then change the value of HTMLFILE to the location where you want the HTML file stored.
- The parameter NSHAPE= specifies the variable in the SAS data set that encodes the shape of each node.
- The parameter NCOLOR= specifies the variable in the SAS data set that encodes the color of each node.

```
/*Define a nodes data set of countries and languages  */
data nodedata;
input nodeLabel $15. shape $10. color $8. size;
cards;
France          square    red       .1
Germany         square    red       .1
Italy           square    red       .1
Belgium         square    red       .1
Switzerland     square    red       .1
Holland         square    red       .1
German          triangle  blue      .1
French          triangle  blue      .1
Italian         triangle  blue      .1
Flemish         triangle  blue      .1
Dutch           triangle  blue      .1
```

```
;
run;

/*Define a links data set */
data linkdata;
input from $15. to $15.;
cards;
France         French
Germany        German
Belgium        French
Belgium        German
Belgium        Flemish
Belgium        Dutch
Switzerland    French
Switzerland    German
Switzerland    Italian
Italy          Italian
Italy          German
Holland        Dutch
;
run;
goptions reset=all;
/* make sure ods listing is open when running macro */
ods listing;
 /*Run the DS2CONST macro*/
%ds2const(ndata=nodedata,
          ldata=linkdata,
          datatype=arcs,
          cnode=red,
          colormap=y,
          height=400,
          width=500,
          code=ConstChart,
          codebase=http://your_path_to_archive,
          htmlfile=your_path_and_filename.htm,
          nid=nodelabel,
          nlabel=nodelabel,
          lfrom=from,
          lto=to,
          fntsize=12,
          nshape=shape,
          ncolor=color,
          nsize=size);
```

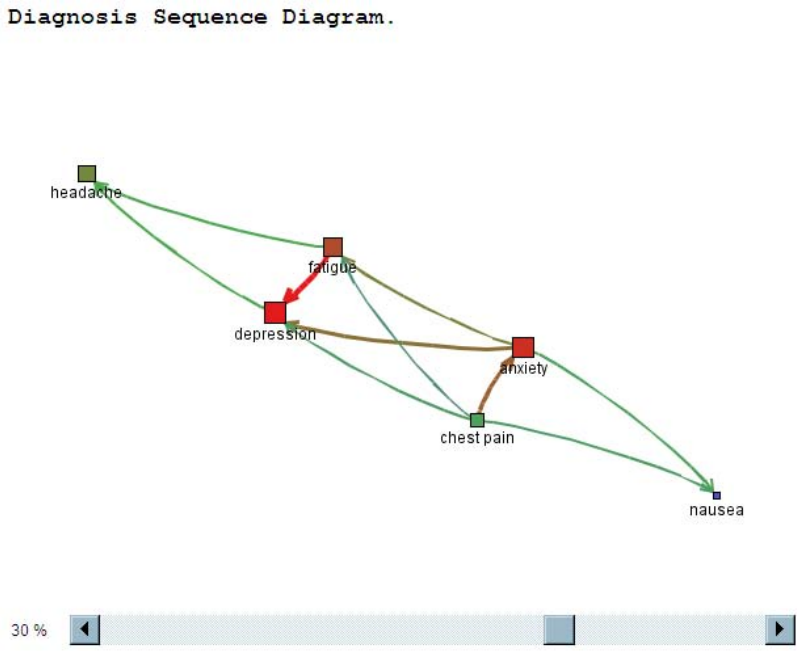
Constellation Chart with DATATYPE=ASSOC

This sample program generates a very simple Constellation diagram with DATATYPE=ASSOC.

Results Shown in a Browser

The following is the Constellation diagram that is generated by the sample code. A Constellation diagram with DATATYPE=ASSOC depicts the strength of the relationships among variables. Variables in the SAS data set determine the size and

color of nodes, as well as the width and color of the lines between nodes. At the bottom of the picture, notice the slider bar which allows a user to choose how many of the links on the diagram are displayed. Move the slider to the left, and only the most important links are displayed. Move the slider to the right, and all of the links are displayed.



SAS Code

The following is the complete SAS code to generate a Constellation diagram from a SAS data set. Notice the following:

- The parameter HTMLFILE= specifies the complete path and name of the HTML file to be created by DS2CONST. If you want to run this sample, then change the value of HTMLFILE to something that makes sense for you.
- The parameter NVALUE= specifies the data set variable that is used to determine the size and color of each node.
- The parameter LVALUE= specifies the data set variable that is used to determine the width and color of each line between nodes.

```
data nodedata;
length nodeID value 8 label $11 tip $25;
input nodeID value @11 label $char11. @25 tip $char25.;
cards;
0   6556  depression    depression: #6556
1   6322  anxiety       anxiety: #6322
2   5980  fatigue       fatigue: #5980
3   5286  headache      headache: #5286
4   4621  chest pain    chest pain: #4621
6   3149  nausea        nausea: #3149
;
run;

data linkdata;
length from to linkvalue 8 tip $40;
```

```
input from to linkvalue @13 tip $char40.;
cards;
2  0  5978  #5978, Support:63.0790, Conf:99.9833
4  1  4621  #4621, Support:48.7602, Conf:100.0000
1  0  4307  #4307, Support:45.4469, Conf:68.1272
1  2  3964  #3964, Support:41.8276, Conf:62.7017
2  3  3010  #3010, Support:31.7611, Conf:50.3429
0  3  3009  #3009, Support:31.7506, Conf:47.5957
1  6  2772  #2772, Support:29.2498, Conf:43.8469
4  6  2609  #2609, Support:27.5298, Conf:56.4596
4  0  2606  #2606, Support:27.4982, Conf:56.3947
4  2  2263  #2263, Support:23.8789, Conf:48.9721
3  0  1980  #1980, Support:20.8927, Conf:40.6821
3  1  1701  #1701, Support:17.9487, Conf:34.9497
3  2  1701  #1701, Support:17.9487, Conf:34.9497
1  3  1593  #1593, Support:16.8091, Conf:25.1977
4  3  1152  #1152, Support:12.1557, Conf:24.9297
0  6  623   #623, Support:6.5738, Conf:9.8545
2  6  623   #623, Support:6.5738, Conf:10.4198
6  3  597   #597, Support:6.2995, Conf:20.0268
3  6  372   #372, Support:3.9253, Conf:7.6433
6  0  344   #344, Support:3.6298, Conf:11.5398
run;

/* make sure ods listing is open when running macro */
ods listing;

title1 "Diagnosis Sequence Diagram.";
%ds2const(ndata=nodedata,
          ldata=linkdata,
          datatype=assoc,
          minlnkwt=30,
          height=450,
          width=600,
          codebase=http://your_path_to_archive,
          htmlfile=your_path_and_filename.htm,
          colormap=y,
          nid=nodeID,
          nlabel=label,
          nvalue=value,
          fntsize=12,
          ntip=tip,
          lfrom=from,
          lto=to,
          lvalue=linkvalue,
          ltip=tip,
          linktype=arrow);
```

Constellation Chart with XML Written to an External File

This sample program generates the same Constellation diagram as the previous example,“Constellation Chart with DATATYPE=ASSOC” on page 562, with the difference that the XML is written to an external file instead of being embedded in the HTML file.

SAS Code

The following is the complete SAS code to generate the Constellation diagram from a SAS data set. You can notice the following:

- The parameter HTMLFILE= specifies the complete path and name of the HTML file to be created by DS2CONST. If you want to run this sample, then change the value of HTMLFILE to something that makes sense for you.
- The parameter XMLTYPE=EXTERNAL tells the DS2CONST macro that the XML that it generates from the SAS data set should be written to an external file.
- The parameter XMLFILE= specifies the path and file name of the XML file to be created.
- The parameter XMLURL= specifies how the XML file is to be addressed from within the HTML file.

```
data nodedata;
length nodeID value 8 label $11 tip $25;
input nodeID value @11 label $char11. @25 tip $char25.;
cards;
0   6556  depression    depression: #6556
1   6322  anxiety       anxiety: #6322
2   5980  fatigue       fatigue: #5980
3   5286  headache      headache: #5286
4   4621  chest pain    chest pain: #4621
6   3149  nausea        nausea: #3149
;
run;

data linkdata;
length from to linkvalue 8 tip $40;
input from to linkvalue @13 tip $char40.;
cards;
2  0  5978  #5978, Support:63.0790, Conf:99.9833
4  1  4621  #4621, Support:48.7602, Conf:100.0000
1  0  4307  #4307, Support:45.4469, Conf:68.1272
1  2  3964  #3964, Support:41.8276, Conf:62.7017
2  3  3010  #3010, Support:31.7611, Conf:50.3429
0  3  3009  #3009, Support:31.7506, Conf:47.5957
1  6  2772  #2772, Support:29.2498, Conf:43.8469
4  6  2609  #2609, Support:27.5298, Conf:56.4596
4  0  2606  #2606, Support:27.4982, Conf:56.3947
4  2  2263  #2263, Support:23.8789, Conf:48.9721
3  0  1980  #1980, Support:20.8927, Conf:40.6821
3  1  1701  #1701, Support:17.9487, Conf:34.9497
3  2  1701  #1701, Support:17.9487, Conf:34.9497
1  3  1593  #1593, Support:16.8091, Conf:25.1977
4  3  1152  #1152, Support:12.1557, Conf:24.9297
0  6  623   #623, Support:6.5738, Conf:9.8545
2  6  623   #623, Support:6.5738, Conf:10.4198
6  3  597   #597, Support:6.2995, Conf:20.0268
3  6  372   #372, Support:3.9253, Conf:7.6433
6  0  344   #344, Support:3.6298, Conf:11.5398
run;
title1 "Diagnosis Sequence Diagram.";
```

```
/* make sure ods listing is open when running macro */
ods listing;

%ds2const(ndata=nodedata,
          ldata=linkdata,
          datatype=assoc,
          minlnkwt=30,
          height=450,
          width=600,
          codebase=http://your_path_to_archive,
          htmlfile=your_path_and_filename.htm,
          xmltype=external,
          makexml=y,
          xmlurl=http://www.xyz.com/Web_output/const_assoc_external.xml,
          xmlfile=u://Web_output/const_assoc_external.xml,
          colormap=y,
          nid=nodeID,
          nlabel=label,
          nvalue=value,
          fntsize=12,
          ntip=tip,
          lfrom=from,
          lto=to,
          lvalue=linkvalue,
          ltip=tip,
          linktype=arrow);
```

Constellation Chart with Hotspots

This sample program generates the same Constellation diagram as in "Constellation Chart with DATATYPE=ARCS" on page 560 and adds hotspots to the nodes of the diagram.

SAS Code

The following is the complete SAS code to generate the Constellation diagram from a SAS data set. Notice the following:

- The parameter NURL= specifies the variable in the SAS data set that contains the URL to be linked to when a user double-clicks the node.

```
/*Define a nodes data set of countries and languages  */
data nodedata;
input nodeLabel $15. shape $10. color $8. size url $40.;
cards;
France          square    red       .1 http://www.xyz.com
Germany         square    red       .1 http://www.xyz.com/rnd/webgraphs/
Italy           square    red       .1 http://www.xyz.com
Belgium         square    red       .1 http://www.xyz.com/rnd/webgraphs/
Switzerland     square    red       .1 http://www.xyz.com
Holland         square    red       .1 http://www.xyz.com
German          triangle  blue      .1 http://www.xyz.com
French          triangle  blue      .1 http://www.xyz.com/rnd/webgraphs/odssyntax.htm
```

```
Italian         triangle  blue      .1 http://www.xyz.com
Flemish         triangle  blue      .1 http://www.xyz.com
Dutch           triangle  blue      .1 http://www.xyz.com
;
run;

/*Define a links data set: */
data linkdata;
input from $15. to $15.;
cards;
France          French
Germany         German
Belgium         French
Belgium         German
Belgium         Flemish
Belgium         Dutch
Switzerland     French
Switzerland     German
Switzerland     Italian
Italy           Italian
Italy           German
Holland         Dutch
;
run;
goptions reset=all;
 /* make sure ods listing is open when running macro */
ods listing;
 /*Run the DS2CONST macro:*/
%ds2const(ndata=nodedata,
          ldata=linkdata,
                  nurl=url,
                  datatype=arcs,
          cnode=red,
          colormap=y,
          height=400,
          width=500,
          code=ConstChart,
          codebase=http://your_path_to_archive,
          htmlfile=your_path_and_filename.htm,
          nid=nodelabel,
          nlabel=nodelabel,
          lfrom=from,
          lto=to,
          fntsize=12,
          nshape=shape,
          ncolor=color,
          nsize=size);
```

CHAPTER

26

Macro Arguments for the DS2CONST and DS2TREE Macros

Macro Arguments

Macro arguments specify the configuration of the HTML output file, the location of the data that is used to generate the diagram, and the configuration of the applet's interactive features.

The macros use the following syntax:

%macroname(*argument1=value1*, *argument2=value2*, ...);

- "Arguments for the APPLET Tag" on page 569. The CODEBASE argument is required.
- "DS2TREE and DS2CONST Arguments for Data Definition" on page 571. For DS2TREE, the arguments NDATA and NID are required. For DS2CONST the arguments NDATA, NID, LDATA, and LTO are required.
- "Arguments for Generating HTML and XML Files" on page 578.
- "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579.
- "Arguments for Page Formatting" on page 585.
- "Arguments for Stylesheets" on page 587.
- "Arguments for the SAS TITLE and FOOTNOTE Tags" on page 589.
- "Arguments for Character Transcoding" on page 593.

Arguments for the APPLET Tag

The following arguments configure the APPLET tag in the HTML output file. The CODEBASE argument is required.

AHUNITS=PIXELS | PERCENT
specifies the units of the HEIGHT= argument. The default value is PIXELS. See also the AWUNITS= argument.

Used by: DS2TREE, DS2CONST

ALIGN=*position*
specifies the alignment of the applet window in the browser window or frame. Values can be LEFT, RIGHT, TOP, BOTTOM, TEXTTOP, MIDDLE, ABSMIDDLE, BASELINE, or ABSBOTTOM.

Used by: DS2TREE, DS2CONST

ALT=*text*
specifies the text that will be displayed on mouseover by browsers that understand the tag but cannot run Java applets. The default value is **`SAS Institute Inc. applet_name`**.

Used by: DS2TREE, DS2CONST

ARCHIVE=*filename*
specifies the name of the Java archive file(s).

Note: The path to the Java archive is specified in the CODEBASE argument. △

The following table shows what archive files to use with each of the macros. For DS2TREE and DS2CONST, you do not have to specify a value for ARCHIVE= because the values shown are generated by default.

DS2TREE	archive=%str(sas.graph.treeview.jar, sas.graph.nld.jar, sas.graph.j2d.jar)
DS2CONST	archive=%str(sas.graph.constapp.jar, sas.graph.nld.jar, sas.graph.j2d.jar)

Note: Before SAS 9.1, treeview.jar and constapp.jar also contained the classes that are now included in the auxiliary JAR files (sas.graph.nld.jar and sas.graph.j2d.jar). Although you can continue to use the older JAR files by specifying ARCHIVE=treeview.jar or ARCHIVE=constapp.jar, future versions may not support these older JAR files. △

Used by: DS2TREE, DS2CONST

AWUNITS=PIXELS | PERCENT
specifies the units of the WIDTH= argument. The default value is PIXELS. See also the HEIGHT= and AHUNITS= arguments.

Used by: DS2TREE, DS2CONST

CODEBASE=*path-or-URL*
specifies the path of the SAS Java archives specified in the ARCHIVE= argument.
The CODEBASE argument is required. You can specify CODEBASE="." if the HTML file and Java archive files are in the same directory.

Note: You can specify the location pointed to by the SAS system option APPLETLOC=, or you can specify a different location. To display the current value of APPLETLOC, run the following code:

```
proc options option=appletloc;
run;
```

The value of the APPLETLOC system option is not used as the default value. △

Used by: DS2TREE, DS2CONST

HEIGHT=*applet-height*
specifies the height of the applet window. The unit of measure is pixels unless changed by the AHUNITS= argument. The default value is 600 for all macros.

Used by: DS2TREE, DS2CONST

HSPACE=*pixels*
specifies the amount of horizontal space, in pixels, to the left and right of the graph or diagram.

Used by: DS2TREE, DS2CONST

NAME=*applet-name*
specifies the name for this instance of the applet. You need to use this argument only if you have more than one instance of the APPLET tag in your HTML file, and if you have included your own scripts or DHTML that communicates with or acts on a particular instance of the applet.

Used by: DS2TREE, DS2CONST

VSPACE=*pixels*
specifies the amount of vertical space, in pixels, to the top and bottom of the graph or diagram.

Used by: DS2TREE, DS2CONST

WIDTH=*applet-width*
specifies the width of the applet window. The unit of measure defaults to pixels unless specified by the AWUNITS= argument.

Used by: DS2TREE, DS2CONST

DS2TREE and DS2CONST Arguments for Data Definition

The following arguments for the DS2TREE and DS2CONST macros define how the applet will use the data set to generate the node/link diagram.

For DS2TREE the arguments NDATA and NID are required.

For DS2CONST the arguments NDATA, NID, LDATA, and LTO are required.

DATATYPE=ARCS | ASSOC | HIER
specifies the type of the XML data. Valid values are defined as follows:

ARCS
indicates that the data set is in the form of an arc list. This is the default value.

ASSOC
indicates that the data set is associative. The links can be displayed based on their weighted values, and node size and link width can represent the relative size of the node and link values.

HIER
indicates that the data set is hierarchical.

Used by: DS2CONST

LABELS=Y | N
indicates whether or not node labels are displayed in the diagram. The default value is Y.

Used by: DS2CONST, DS2TREE

LAYOUT=AUTO | USER
when the value is AUTO (default), specifies that the Constellation Applet lays out the diagram using stress and strain equations. Specifying the value USER indicates that the node positions are specified in the NX and NY arguments.

Used by: DS2CONST

LCOLOR=*variable-name*
specifies the name of the variable that determines the color of the link lines. The values of this variable must be HTML 3.2 color names, or you must use the LCOLFMT= argument to convert those values to valid color names. The default color is provided by the CLINK= argument (see "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579).

In the DS2CONST macro, the LCOLOR= argument is overridden by the LCOLVAL= argument.

Used by: DS2CONST, DS2TREE

LCOLFMT=*user-defined-format-name*
specifies the name of a user-defined SAS format that converts the values in the variable named in the LCOLOR= argument to valid HTML color names. Note that the SAS format does not change any values in the data set. The formatted values are applied to the diagram only.

Used by: DS2CONST, DS2TREE

LCOLVAL=*variable-name*
specifies the name of the variable that determines the color mapping of link lines. This argument is valid only when the value of the DATATYPE= argument is ASSOC, and only when the value of the COLORMAP= argument is Y. If the LCOLVAL= argument is not specified, the link colors are determined by the following arguments in the following order: LCOLOR= (see above) and CLINK= (see "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579).

Used by: DS2CONST

LDATA=*data-set-name*
specifies the name of the SAS data set that contains the link data that is used to generate the diagram.

This argument is required.

Used by: DS2CONST

LFROM=*variable-name*
specifies the name of the variable whose values define the nodes at the start of link lines. The LFROM variable values must be coordinated with the values of the variables that are named in the NID= and LTO= arguments.

This argument is required.

Used by: DS2CONST

LINKTYPE=LINE | ARROW
when the value is ARROW (default), indicates that link lines are to be drawn with arrowheads that indicate the direction of flow.

Used by: DS2CONST

LPT=*password*
specifies the password that is needed for accessing a password-protected link data set (specified with the LDATA= argument). The LPT= argument is required if the data set has a READ or PW password. You do not need to specify this argument if the data set has a WRITE or ALTER password.

Used by: DS2CONST

LSTIP=*variable-name*
specifies the name of the variable in the data set that determines the stipple mask. The stipple mask generates dashed or dotted link lines. The value of the variable must be an integer, which is then converted into a binary value. In the binary value, a "1" bit means that a pixel is to be drawn and a "0" bit means that

no pixel is to be drawn. For example, if the variable has a value of 61680, the binary conversion of that value will be 1111000011110000. This stipple mask generates a dashed link line with dashes and spaces that are four pixels wide. See also the LSTIPFAC= argument.

Used by: DS2CONST, DS2TREE

LSTIPFAC=*variable-name*
specifies the name of the variable in the data set whose value specifies a multiplier for the binary stipple mask (see the LSTIP= argument). The multiplier lengthens the dashes in the base mask. For example, if the multiplier is 2, a stipple mask that specifies 4-pixel dashes and 4-pixel spaces will generate link lines with 8-pixel dashes and spaces.

Used by: DS2CONST, DS2TREE

LTIP=*variable-name*
specifies the name of the variable in the data set that provides the text that is displayed in the pop-up data tips windows for links.

Used by: DS2CONST, DS2TREE

LTIPFMT=*user-defined-format-name*
specifies the name of a user-defined SAS format that is applied to the values in the variable specified in the LTIP= argument to configure those values for display in the pop-up data tips window. Note that the SAS format does not change any values in the data set. The formatted values are applied to the diagram only.

Used by: DS2CONST, DS2TREE

LTO=*variable-name*
specifies the name of the variable whose values identify the nodes at the ends of link lines. The LTO variable values must be coordinated with the values of the variables that are named in the LFROM and NID arguments.
This argument is required.

Used by: DS2CONST

LVALUE=*variable-name*
specifies the name of the variable whose values determine the weights of the link lines, which determines the color and relative thickness of link lines. The variable values must be real numbers. The link weights are used with the MINLNKWT= argument (see below) and the SCLNKWT= argument (see "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579) to control the display of link lines. The LVALUE= argument is valid only when the value of the DATATYPE= argument is ASSOC.

Used by: DS2CONST

LWHERE=*subset-expression*
specifies a WHERE clause that subsets the link data for display in the diagram. If the expression contains any special characters (for example, % or &), include %NRBQUOTE in the expression to process those characters correctly. The following example shows how to correctly specify INT%:

```
LWHERE=%NRBQUOTE(value="Int%")
```

See also the NWHERE argument.

Used by: DS2CONST

LWIDTH=*variable-name*
specifies the name of the variable in the data set that determines the width of the link lines.

For DS2CONST: When this argument is not specified, the width is determined by the LVALUE argument. This argument is valid for DS2CONST only when the value of the DATATYPE argument is ASSOC.

Used by: DS2CONST, DS2TREE

MINLNKWT=*minimum-link-weight*
specifies the initial minimum link weight, which determines which links are initially displayed. The initial diagram show only those links that have weights that are greater than or equal to the minimum weight. In the Constellation Applet, a scroll bar allows the Web user to change the minimum link weight to change the number of links that are displayed. Selecting the brower's Refresh option restores the intial minimum link weight that is specified in the MINLNKWT argument. Link weights are determined by the LVALUE argument. This argument is valid only when the value of the DATATYPE argument is ASSOC.

Used by: DS2CONST

NACTION=*variable-name*
specifies the name of the variable in the nodes data set that provides the menu text that is displayed when the Web user selects a node with the right mouse button. Selecting this menu option text displays the URL that is associated with that node in the NURL= argument. This argument overrides the ACTION= argument (see "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579). The default menu option text is **`Open URL`**.

Used by: DS2CONST, DS2TREE

NCOLFMT=*SAS-format-name*
specifies the name of a user-defined SAS format that converts the values in the variable named in the NCOLOR= argument to valid HTML color names. Note that the data in the data set is not altered; the formatted value is used in the hierarchical tree rather than the data value.

Used by: DS2CONST, DS2TREE

NCOLOR=*variable-name*
specifies the variable in the nodes data set that determines the background color of the nodes, using HTML 3.2 color names or 6-digit hexadecimal RGB values . If the variable does not contain valid HTML color names, then you can use the NCOLFMT=argument to convert those values to the HTML color names. See also the NCOLVAL= and NVALUE=arguments.

Used by: DS2CONST, DS2TREE

NCOLVAL=*variable-name*
specifies the name of the variable in the nodes data set that determines the color mapping for the nodes. This argument is valid only when the DATASET= argument is set to ASSOC, and only when the value of the COLORMAP= argument is Y. If this argument is not specified, then the node color is determined by the LVALUE= argument.

Used by: DS2CONST

NDATA=*SAS-data-set-name*
specifies the SAS data set that contains the node data.
This argument is required.

Used by: DS2CONST, DS2TREE

NFNTNAME=*node-font-variable-name*
specifies the name of the variable that determines the text font for the node labels. The variable value can be SERIF, SANSSERIF, DIALOG, DIALOGINPUT, or MONOSPACED. The default node font is specified by the FNTNAME= argument

(see "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579).

Used by: DS2CONST, DS2TREE

NFNTSIZE=*variable-name*
specifies the name of the variable in the nodes data set that determines the size of the text font used for node labels. This font size is expressed in points. This argument overrides the FNTSIZE= argument.

Used by: DS2CONST, DS2TREE

NFNTSTYL=*node-font-style-variable-name*
specifies the name of the variable that determines the font style for the node label. The valid values that can be assigned to the variable are BOLD, ITALIC, and PLAIN.

Used by: DS2CONST, DS2TREE

NID=*variable-name*
specifies the name of the variable in the nodes data set whose values are to illustrated as the nodes in the diagram. The node ID variable type can be either numeric or character. For the DS2CONST macro, the values of the NID variable must be coordinated with the values of the LFROM and LTO variables.
This argument is required.

Used by: DS2CONST, DS2TREE

NLABEL=*node-label-variable-name*
specifies the name of the variable that represents the node labels. This variable type can be either numeric or character.

Used by: DS2CONST, DS2TREE

NPARENT=*node-parent-variable-name*
specifies the name of the variable that represents the parent nodes. This variable type can be either numeric or character.

Used by: DS2TREE

NPW=*password*
specifies the password that is needed for accessing a password-protected data set. This argument is required if the data set has a READ or PW password. You do not need to specify this argument if the data set has only WRITE or ALTER passwords.

Used by: DS2CONST, DS2TREE

NSCBACK=*variable-name*
specifies the name of the variable in the node styles data set that determines the background color of the nodes. The variable values must be HTML 3.2 color names. The default value is determined by the CNODE= argument (see "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579).

Used by: DS2CONST, DS2TREE

NSCTEXT=*variable-name*
specifies the name of the variable in the node styles data set that provides the colors for the node label text. Valid variable values must be HTML 3.2 color names. The default color is provided by the CATEXT= argument.

Used by: DS2CONST, DS2TREE

NSDATA=*SAS-data-set-name*
specifies the name of the node styles data set.

Used by: DS2CONST, DS2TREE

NSFNTNAM=*variable-name*
specifies the name of the variable in the node styles data set that determines the text font that is to be used for node labels. Valid variable values can be SERIF, SANSSERIF, DIALOG, DIALOGINPUT, or MONOSPACED. This argument overrides the FNTNAME= argument.

Used by: DS2CONST, DS2TREE

NSFNTSIZ=*variable-name*
specifies the name of the variable in the node styles data set that determines the size of the node label text, in points. This argument overrides the FNTSIZE= argument.

Used by: DS2CONST, DS2TREE

NSFNTSTY=*variable-name*
specifies the name of the variable in the node styles data set that determines the style of the node label text. Valid variable values can be BOLD, ITALIC, or the default value, PLAIN. This argument overrides the FNTSTYL= argument.

Used by: DS2CONST, DS2TREE

NSHAPE=*variable-name*
specifies the name of the variable that determines the shape of the nodes. Valid variable values can be CIRCLE, DIAMOND, NONE, SQUARE, or TRIANGLE. The default value is SQUARE. This argument overrides the NODESHAP= argument.

Used by: DS2CONST

NSID=*variable-name*
specifies the name of the variable in the node styles data set that represents the nodes.

Used by: DS2CONST, DS2TREE

NSIZE=*variable-name*
specifies the name of the variable that determines the size of the nodes. The values of this variable can be real numbers. Node sizes are determined based on the value of the LAYOUT= argument. When LAYOUT=USER, the values of the NSIZE variable are interpreted as literal pixel measurements. When LAYOUT=AUTO, the values of the NSIZE variable determine the size of the nodes based on the relative size of individual values. The values of the NSIZE variable can be scaled with the SCNSIZE= argument (see “DS2TREE and DS2CONST Arguments for Diagram Appearance” on page 579). This argument is valid only when the value of the DATATYPE= argument is ASSOC.

Used by: DS2CONST

NSPW=*password*
specifies the password that is needed to access a password-protected node styles data set. This argument is required if the data set has a READ or PW password. You do not need to specify this argument if the data set has only WRITE or ALTER passwords.

Used by: DS2CONST, DS2TREE

NSTYLE=*variable-name*
specifies the name of the variable that determines the style of the nodes. This variable type can be either numeric or character, and the values must correspond to the node identifiers specified in the NSID= argument.

Used by: DS2CONST, DS2TREE

NSWHERE=*subset-expression*
specifies a WHERE clause that subsets the node styles data set for display in the diagram. If the expression contains any special characters (for example, % or &), then include %NRBQUOTE in the expression to process those characters correctly. The following example shows how to correctly specify INT%:

```
NSWHERE=%NRBQUOTE(value="Int%")
```

Used by: DS2CONST, DS2TREE

NTEXTCOL=*variable-name*
specifies the name of the variable that determines the color of the text for the node labels. Valid variable values must be HTML 3.2 color names.

Used by: DS2CONST, DS2TREE

NTIP=*variable-name*
specifies the name of the variable that provides the data or text that is displayed in the pop-up data tips window.

Used by: DS2CONST, DS2TREE

NTIPFMT=*user-defined-format-name*
specifies the name of a user-defined SAS format that is applied to the data tips variable that is named in the NTIP= argument. Note that the data set is not altered; the formatted value is used only in the diagram.

Used by: DS2CONST, DS2TREE

NURL=*drill-down-URL*
specifies the name of the variable that provides the drill-down URLs for the nodes. These URLs are displayed when the Web user double-clicks on a node or selects the node with the right mouse button and chooses an option from the pop-up menu. Menu text is determined by the NACTION= argument above and by the ACTION= argument in "DS2TREE and DS2CONST Arguments for Diagram Appearance" on page 579. The default menu option text is **Open URL**.

Used by: DS2CONST, DS2TREE

NVALUE=*variable-name*
specifies the name of the variable that determines the relative node size. This argument is valid only when DATATYPE=ASSOC.

If you do not specify a particular node color using either the NCOLOR or NCOLVAL argument (and if COLORMAP=Y), then this argument also determines a default node color. By default, the largest value of NVALUE is mapped to red, the median value to green, and the lowest value to blue. Values in between result in interpolated colors.

Used by: DS2CONST

NWHERE=*subset-expression*
specifies a WHERE clause that subsets the nodes data set for display in the diagram. If the expression contains any special characters (for example, % or &), then include %NRBQUOTE in the expression to process those characters correctly. The following example shows how to correctly specify INT%:

```
NWHERE=%NRBQUOTE(value="Int%")
```

See also the LWHERE= argument.

Used by: DS2CONST, DS2TREE

NX=*variable-name*
NY=*variable-name*
specify the variables that determine the locations of the centers of the nodes. These arguments are valid only when the LAYOUT= argument is set to USER. The values are expressed in pixels. Positive values are measured from the top-left corner of the screen. Negative values are measured from the bottom-right corner of the screen.

Used by: DS2CONST

Arguments for Generating HTML and XML Files

The following arguments determine the name, storage location, and file makeup of Web presentations that run in the Constellation Applet or the Treeview Applet.

HTMLFILE=*external-filename*
specifies the name and storage location of the HTML output file. If the external file does not exist, then it is created for you. Either this argument, or HTMLFREF=, is required if you specify MAKEHTML=Y. Note: Do not use the HTMLFILE= argument if you use the HTMLFREF= argument.

Used by: DS2TREE, DS2CONST

HTMLFREF=*fileref*
specifies the SAS fileref that identifies the name and storage location of the HTML output file. If the external file does not exist, then it is created for you. Either this argument, or HTMLFILE=filename, is required if you specify MAKEHTML=Y. Note: Do not use the HTMLFREF= argument if you use the HTMLFILE= argument, and do not use a reserved name (see "Reserved Names" on page 594).

Used by: DS2TREE, DS2CONST

MAKEHTML=Y | N
specifies whether or not an HTML file is to be generated. The default value is Y, which generates the HTML output file. If you specify MAKEHTML=N and MAKEXML=Y, then only an XML file is generated.

Used by: DS2TREE, DS2CONST

MAKEXML=Y | N
specifies whether or not an XML file is to be generated. The default value is Y, which generates the XML output file. If you specify MAKEXML=N and MAKEHTML=Y, then only an HTML file will be generated. Note that under these circumstances, you must specify a value for the XMLURL= argument.

Used by: DS2TREE, DS2CONST

OPENMODE=REPLACE | APPEND
indicates whether the new HTML or XML output or both overwrites the information that is currently in the specified file(s), or if the new output is appended to the end of the existing file(s). The default value is REPLACE. Specify APPEND to add your new HTML-enhanced output to the end of an existing file. Note: OPENMODE=APPEND is not valid if you are writing your resulting HTML to a partitioned data set (PDS) on z/OS.

Used by: DS2TREE, DS2CONST

RUNMODE=B | S
specifies whether you are running the DS2TREE macro in batch or server mode. Batch mode (RUNMODE=B, the default) means that you are submitting the DS2TREE macro in the SAS Program Editor or you have included it in a SAS

program. Server mode (RUNMODE=S) generates the HTTP header that is required by Application Dispatcher in the SAS/INTRNET software.

Used by: DS2TREE, DS2CONST

XMLFILE=*external-filename*
specifies the name and storage location of the XML output file. If the external file does not exist, then it is created for you. This argument, or XMLFREF=, is required if you specify MAKEXML=Y and XMLTYPE=EXTERNAL. Note: Do not use the XMLFILE= argument if you use the XMLFREF= argument.

Used by: DS2TREE, DS2CONST

XMLFREF=*fileref*
specifies the SAS fileref that identifies the name and storage location of the XML output file. If the external file does not exist, then it is created for you. This argument, or XMLFILE=, is required if you specify MAKEXML=Y and XMLTYPE=EXTERNAL. Note: Do not use the XMLFREF= argument if you use the XMLFILE= argument, and do not use a reserved name (see "Reserved Names" on page 594).

Used by: DS2TREE, DS2CONST

XMLTYPE=INLINE | EXTERNAL
specifies whether the XML output file is to be written to an external file or included inline with the HTML. The default value is INLINE. If you specify EXTERNAL you must also specify a value for either the XMLFILE= or XMLFREF= arguments. This argument is required if you specify MAKEXML=Y.

Used by: DS2TREE, DS2CONST

XMLURL=*URL*
specifies the URL of the existing file that contains the XML tags that define the node/link diagram. This argument is required if specified XMLTYPE=EXTERNAL.

Used by: DS2TREE, DS2CONST

DS2TREE and DS2CONST Arguments for Diagram Appearance

The following arguments for the DS2TREE and DS2CONST macros specify non-default behavior and appearance of the node/link diagram in the respective applet. None of the following arguments are required.

ACTION=*text*
specifies the default text that is displayed in a pop-up menu when the Web user selects a node with the right mouse button. Selecting this menu option displays the URL that is associated with that node in the NURL= argument. This argument is overridden by the NACTION= argument (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571). The ACTION= argument is useful when you want to use a single menu text string for most of the nodes in your diagram. The default menu option text is **`Open URL`**.

Used by: DS2CONST, DS2TREE

ANGLE=*link-angle*
works with the TREESPAN= argument to determine the direction of growth for the diagram. The ANGLE= argument is valid only when you do not specify the TREEDIR= argument. The TREESPAN= argument defines the angular width of the tree (narrow or wide layout). The TREESPAN angle can be visualized as a V shape, with the starting node positioned at the base of the V. The rest of the nodes are laid out between the spreading arms of the V. The ANGLE= argument specifies the angle of the V shape. By default, the value of the ANGLE= argument

is zero (0) and the V shape opens to the right, as if the letter V was rotated 90 degrees clockwise, to the three-o'clock position. Values of the ANGLE= argument that are greater than zero rotate the V shape counterclockwise away from the three-o'clock position. Valid values for the ANGLE= argument range from zero (0) to 360 degrees.

Used by: DS2TREE

BORDER=Y | N
specifies whether or not a border is drawn around the background area. The default value is N.

Used by: DS2CONST, DS2TREE

CATEXT=*default-text-color*
specifies a default color for the text in the diagram, using an HTML 3.2 color name or a 6-digit hexadecimal RGB value. For DS2CONST, this argument is overridden by the FNTNAME= argument (see below) and the NTEXTCOL argument (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571).

Used by: DS2CONST, DS2TREE

CBACK=*color*
specifies a background color for the Treeview Applet. The value must be a valid HTML 3.2 color name.

Used by: DS2TREE

CHANDLE=*color*
specifies the color of the Collapse/Expand handle on the nodes. The handle is represented by a small plus sign (+) that is prefixed to the label of the node when its subtree is collapsed. The value must be a valid HTML color name.

Used by: DS2TREE

CLINK=*default-link-color*
specifies a default color for the links in the diagram, using an HTML 3.2 color name or a 6-digit RGB value. For DS2CONST, this argument is overridden by the LCOLOR= and LCOLVAL= arguments (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571).

Used by: DS2CONST, DS2TREE

CNODE=*color*
specifies the node background color. The value must be a valid HTML color name. The value specified here can be overridden by specifying a value for the NCOLOR= argument.

Used by: DS2TREE

CNODE=*default-node-color*
specifies a default background color for the nodes, using an HTML 3.2 color name or a 6-digit RGB value. This argument is overridden by the NCOLOR=, NCOLVAL=, NVALUE=, or NSCBACK= arguments (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571).

Used by: DS2CONST

COLORMAP=N | Y
when the value is N (default), specifies that the Constellation Applet is to use the NCOLOR= and LCOLOR= arguments (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571) to determine node and link colors rather than using the color map.

Used by: DS2CONST

CSELECT=*color*
specifies a color for nodes that are selected by the mouse or as the result of a node search. The value must be a valid HTML 3.2 color name.

Used by: DS2CONST, DS2TREE

CUTOFF=*detail-percentage*
specifies the percentage of the nodes that will be displayed with node labels. After the percentage has been reached, nodes are drawn as rectangles. The size of those rectangles decreases as the distance from the starting node increases. Valid values range from 0.0 to 1.0 (The decimal value is mapped to a percentage from 0% to 100%). The default value is 0.5. See also the DEPTH argument.

Used by: DS2CONST, DS2TREE

DEPTH=*max-path-length*
specifies a whole number greater than zero that determines the maximum number of links that are to be displayed in the node/link diagram. Paths whose lengths exceed the limit are truncated. This argument affects only the initial display of the diagram. Nodes that are initially hidden can become visible as a user selects nodes and navigates around the diagram.

Note that this value is ignored if CUTOFF= 1.0. There is no default value for this argument.

Used by: DS2TREE

DRILTARG=*target-window-or-frame*
specifies the HTML target or the name of the browser window or frame where drill-down URLs are displayed. The default behavior is to open a new browser window and reuse it for subsequent drill-down requests. Specifically, the default value is _BLANK, which is one of several reserved names for targets in HTML. The value can also be the name of a window or frame in the Web presentation.

Used by: DS2CONST, DS2TREE

DUPCHECK=TRUE | FALSE
specifies whether or not the applet will check for duplicate node IDs. The default value is FALSE. When set to TRUE, this argument will cause the applet to update an ID if a duplicate ID is found, instead of creating a new node with the same ID. This enables you to collect node information from different locations in the data set.

Used by: DS2TREE

FACTOR=*fish-eye-distortion-factor*
specifies the distortion factor for the fish-eye lens. The distortion factor determines the amount that the central region of the display is to be expanded (or zoomed). The value specified must be greater than or equal to 1.0. The default value is 1.0, which represents the lowest amount of distortion. This argument is valid only when the value of the FISHEYE= argument is Y. The maximum effective value (beyond which no further distortion is visible) is variable depending upon the number of nodes in the diagram.

Used by: DS2TREE

FISHEYE=Y | N
indicates whether or not the diagram is to be displayed with the fish-eye distortion, which displays the central region of the diagram at a specified size and displays the rest of the diagram as if it were mapped onto a ball, with the nodes and links disappearing over a curved horizon. The Web user can move the diagram past the central region by scrolling or searching for nodes. The amount of distortion used in the fish-eye lens is determined by the FACTOR= argument. The default value is Y.

Used by: DS2TREE

FNTNAME=*default-node-label-font*
specifies the default text font for node labels. Valid values can be SERIF, SANSSERIF, DIALOG, DIALOGINPUT, or MONOSPACED. This argument is overridden by the NFNTNAME or NSFNTNAM= arguments (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571).

Used by: DS2CONST, DS2TREE

FNTSIZE=*node-font-size*
specifies the size of the node label text font, in points. This argument is overridden by the NFNTSIZE= argument.

Used by: DS2CONST, DS2TREE

FNTSTYL=*node-font-style*
specifies the text font style for node labels. Valid values are BOLD, ITALIC, and PLAIN. PLAIN is the default value. This argument is overridden by the NFNTSTYL= argument.

Used by: DS2CONST, DS2TREE

IBACKLOC=*image-URL*
specifies a URL for the image that you want to use in the background of the diagram. See also the IBACKPOS= argument.

Used by: DS2CONST, DS2TREE

IBACKPOS=CENTER | SCALE | TILE | POSITION
specifies how to display the background image in the IBACKLOC= argument. Specify one of the following options:

CENTER
centers the image in the browser window without resizing the image.

SCALE
resizes the image to fit the browser window.

TILE
fills the browser window by replicating the image at its original size.

POSITION
positions the image without resizing at the values specified by the IBACKX= and IBACKY= arguments.

Used by: DS2CONST, DS2TREE

IBACKURL=*background-drilldown-URL*
specifies the URL that is displayed when you click on the background image. This argument is valid only when the value of the IBACKPOS= argument is POSITION. If you are including the Powered by SAS logo, then you must use this argument to link the image to the SAS Web site.

Used by: DS2CONST, DS2TREE

IBACKX=*corner-coordinate*
IBACKY=*corner-coordinate*
specifies the x (horizontal) and y (vertical) pixel coordinates of the upper left-hand corner of the background image. Positive values are measured from the upper-left corner of the background area. Negative values are measured from the lower-right corner of the background area. These values are valid only if the value of the IBACKPOS= argument is POSITION. Always specify both the IBACKX= and IBACKY= arguments.

Used by: DS2CONST, DS2TREE

NODEBDR=LINE | NONE | FILL | OUTLINE
specifies the appearance of the node border line, using one of the following values:

LINE
show solid border lines around the nodes.

NONE
show no border lines or background.

FILL
show background but no border lines.

OUTLINE
show a border line and background. This is the default value.

Used by: DS2TREE

NODESHAP=*shape*
specifies the shape of the nodes. Valid values can be CIRCLE, DIAMOND, NONE, SQUARE, or TRIANGLE. The default value is SQUARE. This argument is overridden by the NSHAPE= argument (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571).

Used by: DS2CONST

RBSIZING=Y | N
the default value N indicates that size information from the resource bundle is not to be used for sizing the two dialog boxes that can be invoked from the pop-up menu that appears when a user right-mouse-clicks on a diagram. The two dialog boxes are the About dialog box and the Mouse Help dialog box.

Specify Y for this argument for languages other than English.If you specify Y, then the height and the width of the dialog box frames are read in from the resource bundle. This allows translators to set appropriate heights and widths for the frames in the resource bundle, based on the length of the message strings in each language.

Used by: DS2CONST, DS2TREE

SCLNKWT=Y | N
when the value is Y (default), specifies that the link weight values are to be scaled into the range of 0–1, which corresponds to 0–100%. When SCLNKWT=Y, the scroll bar in Constellation Applet displays a percentage of the range of the link weights. When SCLNKWT=N, the link weights are not scaled and the scroll bar reflects the actual link weight data values. These values are real numbers that are specified in the LVALUE= argument (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571). The SCLNKWT= argument is valid only when the value of the DATATYPE= argument is ASSOC. Note that the range of link weights (maximum minus minimum) must be greater than 2 when SCLNKWT=N. Otherwise, the scroll bar will not correctly map the link weights.

Used by: DS2CONST

SCLWIDTH=Y | N
when the value is Y (default), indicates that the link width values are to be scaled into the range of 0–1. Specifying N indicates that the link widths are already scaled into that range. This argument is valid only when the value of the DATATYPE= argument is ASSOC.

Used by: DS2CONST

SCNSIZE=Y | N
when the value is Y (default), indicates that the node size values are to be scaled into the range of 0–1. Specifying N indicates that the node sizes are already scaled

into that range. This argument is valid only when the value of the DATATYPE= argument is ASSOC. Node sizes are specified with the NSIZE= argument (see "DS2TREE and DS2CONST Arguments for Data Definition" on page 571).

Used by: DS2CONST

SHOWLINKS=Y | N
specifies whether initially to display all arc lines between nodes. Specifying N suppresses all arc lines. The default value is Y.

Note: This argument affects only the *initial* display. A viewer can subsequently control which arc lines are displayed by right-mouse clicking and selecting a **`Show links`** option from the pop-up menu. △

Used by: DS2CONST

SPREAD=*angular-factor*
specifies the angular spreading factor for the layout of the diagram. The value specified must be greater than or equal to 1.0. The default value is 1.25.

Used by: DS2TREE

TIPS=Y | N
indicates whether or not pop-up data tips are displayed when the cursor is positioned over nodes or links or both. The default value is Y.

Used by: DS2CONST, DS2TREE

TIPTYPE=TRACKING | STATIONARY
when the value is TRACKING (default), indicates that the pop-up data tips windows are to move with the cursor while the cursor moves within the area of a single node or link.

Used by: DS2CONST

TREEDIR=C | D | L | R | U
determines the growth direction of the node/link diagram using the following values.

C | CIRCULAR
grows the tree in a circular pattern. This is the default value.

D | DOWN
grows the tree from top to bottom using center alignment.

L | LEFT
grows the tree from left to right and top to bottom.

R | RIGHT
grows the tree from right to left and top to bottom.

U | UP
grows the tree from the bottom up using center alignment.

If the value of the TREEDIR= argument is UP or DOWN, then the value of the TREESPAN= argument is used to set the angular width of the diagram. The starting node is aligned horizontally in the center of the applet. The diagram grows out of the starting node based on the angular width specified in the TREESPAN= argument. The wider the angle, the wider the layout of the diagram.
The TREEDIR= argument overrides the ANGLE= argument.

Used by: DS2TREE

TREESPAN=*angular-diagram-width*
specifies the angular width of the diagram in degrees. Valid values must be greater than zero and less than 360. The default value is 60. For details, see the TREEDIR= and ANGLE= arguments.

Used by: DS2TREE

ZOOM=*starting-percentage*
specifies the zoom value that is used for the initial display of the diagram. After the initial display, the Web user can change the zoom percentage dragging the mouse up and down while pressing the Ctrl + left mouse button. Selecting the Refresh button on the browser runs the applet and restores the initial zoom setting. The default value is 100 percent. The initial diagram can be scaled up with a value greater than 100 or scaled down with a value less than 100.

Used by: DS2CONST

Arguments for Page Formatting

The following arguments format the HTML output file. The rendering of some of these arguments may vary in certain browsers. Several of the following arguments apply only to certain macros, as noted in the descriptions of the arguments.

The BGTYPE=, BRTITLE=, CENTER=, CTEXT=, and DOCTYPE= arguments apply to the entire page for the current invocation of the macro. If you append data to an existing HTML page, then the HTML formatting will not change. You may want to use these arguments only when you replace, rather than append, HTML files.

BDCLASS=*body-stylesheet-name*
specifies the name of the stylesheet that is to be applied to the body of the HTML output file.

Used by: DS2TREE, DS2CONST

BG=*color-or-image*
specifies the background color or image, based on the value of the BGTYPE= argument. The color can be specified as an HTML 3.2 color name or as a 6-digit hexadecimal RGB value. When BGTYPE=IMAGE, this argument specifies a background image, using a path or a URL, relative or absolute.

Used by: DS2TREE, DS2CONST

BGTYPE=NONE | COLOR | IMAGE
specifies the background type, using one of the following values:

NONE
causes the applet to display its default background color. This is the default value.

COLOR
specifies that the value of the BG= argument must be an HTML 3.2 color name or hexadecimal RGB value.

IMAGE
specifies that the value of the BG= argument must be the path or URL pointing to an image file that will be displayed in the background of the applet window.

Used by: DS2TREE, DS2CONST

BRTITLE=*browser-window-title*
specifies the text that appears in the title bar of the browser window. By default, no title is displayed.

Used by: DS2TREE, DS2CONST

CENTER= Y | N
specifies whether or not the graph or diagram is centered in the browser window. The default value is N.

Used by: DS2TREE, DS2CONST

CTEXT=*default-text-color*
specifies a default text color that replaces the default text color in the browser. Other color arguments can be used to override this new default. The color can be specified as an HTML 3.2 color name or as a six-digit hexadecimal RGB value.

Used by: DS2TREE, DS2CONST

DOCTYPE=*DOCTYPE-tag*
generates the following DOCTYPE tag by default, which specifies HTML version 3.2:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

To use a different DOCTYPE tag, specify the entire contents of the tag as the value of the DOCTYPE= argument, including the angle brackets.

If you specify DOCTYPE="", then no DOCTYPE tag is generated in the HTML output file.

Used by: DS2TREE, DS2CONST

ENCODE=Y | N
when the value is Y (default), replaces the angle bracket characters (< and >) in SAS TITLE and FOOTNOTE lines with the HTML character entities (> and <) respectively. Specifying ENCODE=N causes the browser to interpret the angle brackets as parts of HTML tags. For example, you would use ENCODE=N if you wanted to use the following TITLE statement:

```
title "<FONT COLOR="red">Out of Range Data</FONT>";
```

Used by: DS2TREE, DS2CONST

PAGEPART=ALL | HEAD | BODY | FOOT
specifies which part or parts of the HTML page are to be written into the HTML output file. This argument is helpful when are appending HTML output to the end of an existing HTML file, or when you are using separate files for the head, body, and foot of your Web page.

ALL
writes the entire HTML file, including the XML tags for the DS2CONST and DS2TREE. This is the default value. Do not use this value if you are appending an existing HTML file.

HEAD
writes the HTML header information and or XML (for DS2CONST and DS2TREE) into the HTML file. The header information consists of the HEAD and BODY tags. HTML footer information is not included.

BODY
writes only the XML tags (for DS2CONST and DS2TREE) into the HTML output file. No head or foot information is generated in the HTML output file.

FOOT
writes metagraphics codes or XML tags and the </BODY> and </HTML> tags to conclude the HTML file.

Used by: DS2TREE, DS2CONST

SASPOWER=*logo-image-file*
specifies the path or URL, relative or absolute, to the image file of the SAS Powered logo. In the HTML file, the image appears at the bottom of the page. Selecting the image displays the SAS home page. By default, the logo is omitted. To obtain the logo image file, see http://www2.sas.com/dispatcher/index.html. See also the SPCLASS= argument.

Used by: DS2TREE, DS2CONST

SEPCLASS=*page-separator-stylesheet*
specifies the path or URL, relative or absolute, to the style sheet that is used for the page separator. If the value of the SEPTYPE= argument is RULE, then the value of the SEPCLASS= argument is used on the CLASS attribute of the HTML tag <HR>. If the value of the SEPTYPE= argument is IMAGE, then the value of SEPCLASS= argument is used on the CLASS attribute of the HTML tag <IMG>.

Used by: DS2TREE, DS2CONST

SEPLOC=*separator-image*
specifies the path or URL, relative or absolute, to the image that you want to use as the separator between the graphs in your presentation. This argument is valid only if the value of the SEPTYPE= argument is IMAGE.

Used by: DS2TREE, DS2CONST

SEPTYPE= IMAGE | NONE | RULE
specifies the type of separator that is used between multiple applets in your presentation. The valid values are defined as follows:

IMAGE
specifies separate graphs using the image specified in the SEPLOC= argument.

NONE
specifies not to use a separator between applets.

RULE
inserts a line between applets. This is the default.

Used by: DS2TREE, DS2CONST

SPCLASS=*logo-stylesheet-name*
specifies the name of the style sheet class that is to be used for the Powered by SAS logo.

Used by: DS2TREE, DS2CONST

Arguments for Stylesheets

DS2CONT and DS2TREE enable the following arguments for style sheet specifications in the HTML output file. See also the BDCLASS=, SEPCLASS=, and SPCLASS= arguments in “Arguments for Page Formatting” on page 585.

Style sheet arguments reference style information in one of two ways. Most of the arguments specify parameters in the HTML LINK tag:

```
<LINK HREF="1qtr98.css" TYPE="text/css" REL="stylesheet">
```

Use these arguments when you do not want to enter your style information directly into your HTML file when you create that file.

Other arguments embed the style information into the header of the HTML file. Use these arguments when you want to collect style information from multiple style sheets. The end result must create a complete STYLE tag in your HTML file.

You can combine LINK tag arguments with arguments that embed style information, but you cannot use the same ordinal number in two arguments. For example, you can specify the arguments SSHREF1= and SSFILE2=, but you cannot specify SSHREF1= and SSFILE1=.

The following arguments link to two different style sheets and include text comments for each stylesheet.

```
ssfile1=comments1.txt,     /* embeds text                */
sshref2=/style/style1.css, /* links to stylesheet        */
sstype2=text/css,          /* parameters for style sheets */
ssrel2=stylesheet,
ssfile3=comments2.txt,     /* embeds text                */
sshref4=/style/style2.css, /* link to stylesheets        */
sstype4=text/css,
ssrel4=stylesheet,
```

SSFILE1–SSFILE5=*file-specification*
embeds in the HTML file the entire contents of the specified file.

Used by: DS2TREE, DS2CONST

SSFREF1–SSFREF5=*fileref*
embeds in the HTML file the entire contents of the file that is referenced by the SAS fileref.

Used by: DS2TREE, DS2CONST

SSHREF1–SSHREF5=*style-sheet-URL*
specifies the URL of the stylesheet in the HREF= attribute of the LINK tag. If you specify a relative URL, it must be relative to the location of the HTML output file.

Used by: DS2TREE, DS2CONST

SSMEDIA1–5=*media*
specifies the media for which the style sheet was designed. The value is applied to the MEDIA= attribute of the LINK tag. The default value is SCREEN. Examples of other valid MEDIA values include BRAILLE for tactile feedback devices, and HANDHELD for small-screen devices.

Used by: DS2TREE, DS2CONST

SSREL1–5=*relationship*
specifies the REL= attribute of the LINK tag, which describes the relationship from the linked file to the HTML file. The value of this tag is generally STYLESHEET. The arguments SSREL1–5= can also be used with the arguments SSREV1–5 to link HTML pages in a series. For example, the SSREL1= argument can specify the next document in the series, and the SSREV2= argument can specify the reverse relationship, which would be the previous document in the series. Both arguments, SSRELn= and SSREVn=, can appear in the same LINK tag.

Used by: DS2TREE, DS2CONST

SSREV1–5=*relationship*
specifies the REV= attribute of the LINK tag, which describes the relationship from the HTML file to the linked file. See the SSREL1–5= argument for details.

Used by: DS2TREE, DS2CONST

SSTITLE1–5=*title-of-linked-page*
specifies the TITLE= attribute of the LINK tag. The TITLE= attribute provides a title for the referenced page. Use this argument when you are using the SSRELn= and SSREVn= arguments to specify next and previous links in a series of Web pages.

Used by: DS2TREE, DS2CONST

SSTYPE1–5=*stylesheet-type*
specifies the TYPE= attribute of the LINK tag. For cascading style sheets, this value usually is TEXT/CSS. For JavaScript style sheets, this value is generally TEXT/JAVASCRIPT.

Used by: DS2TREE, DS2CONST

Arguments for the SAS TITLE and FOOTNOTE Tags

The following arguments determine the content and appearance of the SAS TITLE and FOOTNOTE tags in the HTML output file.

FCLASS=*footnote-style-sheet-name*
TCLASS=*title-style-sheet-name*
specify the name of the style sheet class that is to be used for the SAS TITLE or FOOTNOTE.

Used by: DS2TREE, DS2CONST.

FCOLOR=*footnote-text-color*
TCOLOR=*title-text-color*
specify the color of the text in the SAS TITLE or FOOTNOTE, using an HTML 3.2 color name or a six-digit hexadecimal RGB value.

Used by: DS2TREE, DS2CONST.

FFACE=*footnote-text-font*
TFACE=*title-text-font*
specify a text font for the SAS TITLE or FOOTNOTE. Valid values are browser-specific.

Used by: DS2TREE, DS2CONST.

FSIZE=*n* | +*n* | –*n*
TSIZE=*n* | +*n* | –*n*
specify the size of the text font that is to be used for the SAS TITLE or FOOTNOTE, where *n* is an integer. Valid values are browser-specific depending on how the browser handles the SIZE attribute on the FONT tag.

Used by: DS2TREE, DS2CONST.

FTAG=*tag-string*
TTAG=*tag-string*
specify a text string that the macro translates into one or more tags that will enclose the SAS TITLE or FOOTNOTE.

The default value is as follows:

```
PREFORMATTED + HEADER 3
```

Used by: DS2TREE, DS2CONST.

For each possible value of the TTAG= and FTAG= arguments, the following table shows the HTML tags that are generated by the macro for the SAS TITLE and FOOTNOTE lines (the corresponding end tags are generated automatically):

TTAG or FTAG Value	HTML Tag or Tags Enclosing the SAS TITLE or SAS FOOTNOTE
NO FORMATTING	(none)
STRONG	<STRONG>
EMPHASIS	<EM>

TTAG or FTAG Value	HTML Tag or Tags Enclosing the SAS TITLE or SAS FOOTNOTE
HEADER 1	<H1>
HEADER 2	<H2>
HEADER 3	<H3>
HEADER 4	<H4>
HEADER 5	<H5>
HEADER 6	<H6>
PREFORMATTED TEXT	<PRE>
CITATION TEXT	<CITE>
COMPUTER CODE TEXT	<CODE>
KEYBOARD INPUT TEXT	<KBD>
LITERAL TEXT	<SAMP>
VARIABLE TEXT	<VAR>
BOLD	<B>
ITALICIZED TEXT	<I>
UNDERLINE TEXT	<U>
TYPEWRITER	<TT>
BIG TEXT	<BIG>
SMALL TEXT	<SMALL>
STRIKE OUT TEXT	<STRIKE>
DEFINING INSTANCE TEXT	<DFN>
PREFORMATTED + STRONG	<PRE><STRONG>
PREFORMATTED + EMPHASIS	<PRE><EM>
PREFORMATTED + HEADER 1	<PRE><H1>
PREFORMATTED + HEADER 2	<PRE><H2>
PREFORMATTED + HEADER 3	<PRE><H3>
PREFORMATTED + HEADER 4	<PRE><H4>
PREFORMATTED + HEADER 5	<PRE><H5>
PREFORMATTED + HEADER 6	<PRE><H6>
PREFORMATTED + CITATION	<PRE><CITE>
PREFORMATTED + COMPUTER CODE	<PRE><CODE>
PREFORMATTED + KEYBOARD INPUT	<PRE><KBD>
PREFORMATTED + LITERAL	<PRE><SAMP>
PREFORMATTED + VARIABLE	<PRE><VAR>
PREFORMATTED + BOLD	<PRE><B>
PREFORMATTED + ITALICIZED	<PRE><I>

TTAG or FTAG Value	HTML Tag or Tags Enclosing the SAS TITLE or SAS FOOTNOTE
PREFORMATTED + TYPEWRITER	<PRE><TT>
PREFORMATTED + UNDERLINE	<PRE><U>
PREFORMATTED + BIG	<PRE><BIG>
PREFORMATTED + SMALL	<PRE><SMALL>
PREFORMATTED + STRIKE OUT	<PRE><STRIKE>
PREFORMATTED + DEFINING INSTANCE	<PRE><DFN>
STRONG + EMPHASIS	<STRONG><EM>
STRONG + ITALICIZED	<STRONG><I>
STRONG + CITATION	<STRONG><CITE>
STRONG + COMPUTER CODE	<STRONG><CODE>
STRONG + KEYBOARD INPUT	<STRONG><KBD>
STRONG + LITERAL	<STRONG><SAMP>
STRONG + VARIABLE	<STRONG><VAR>
STRONG + TYPEWRITER	<STRONG><TT>
STRONG + BIG	<STRONG><BIG>
STRONG + SMALL	<STRONG><SMALL>
EMPHASIS + CITATION	<EM><CITE>
EMPHASIS + COMPUTER CODE	<EM><CODE>
EMPHASIS + KEYBOARD INPUT	<EM><KBD>
EMPHASIS + LITERAL	<EM><SAMP>
EMPHASIS + VARIABLE	<EM><VAR>
EMPHASIS + TYPEWRITER	<EM><TT>
EMPHASIS + BIG	<EM><BIG>
EMPHASIS + SMALL	<EM><SMALL>
BOLD + EMPHASIS	<B><EM>
BOLD + ITALICIZED	<B><I>
BOLD + CITATION	<B><CITE>
BOLD + COMPUTER CODE	<B><CODE>
BOLD + KEYBOARD INPUT	<B><KBD>
BOLD + LITERAL	<B><SAMP>
BOLD + VARIABLE	<B><VAR>
BOLD + TYPEWRITER	<B><TT>
BOLD + BIG	<B><BIG>
BOLD + SMALL	<B><SMALL>
ITALICIZED + CITATION	<I><CITE>
ITALICIZED + COMPUTER CODE	<I><CODE>

TTAG or FTAG Value	HTML Tag or Tags Enclosing the SAS TITLE or SAS FOOTNOTE
ITALICIZED + KEYBOARD INPUT	<I><KBD>
ITALICIZED + LITERAL	<I><SAMP>
ITALICIZED + VARIABLE	<I><VAR>
ITALICIZED + TYPEWRITER	<I><TT>
ITALICIZED + BIG	<I><BIG>
ITALICIZED + SMALL	<I><SMALL>
STRONG + EMPHASIS + BIG	<STRONG><EM><BIG>
STRONG + CITATION + BIG	<STRONG><CITE><BIG>
STRONG + COMPUTER CODE + BIG	<STRONG><CODE><BIG>
STRONG + KEYBOARD INPUT + BIG	<STRONG><KBD><BIG>
STRONG + LITERAL + BIG	<STRONG><SAMP><BIG>
STRONG + VARIABLE + BIG	<STRONG><VAR><BIG>
STRONG + TYPEWRITER + BIG	<STRONG><TT><BIG>
EMPHASIS + CITATION + BIG	<EM><CITE><BIG>
EMPHASIS + COMPUTER CODE + BIG	<EM><CODE><BIG>
EMPHASIS + KEYBOARD INPUT + BIG	<EM><KBD><BIG>
EMPHASIS + LITERAL + BIG	<EM><SAMP><BIG>
EMPHASIS + VARIABLE + BIG	<EM><VAR><BIG>
EMPHASIS + TYPEWRITER + BIG	<EM><TT><BIG>
BOLD + EMPHASIS + BIG	<BOLD><EM><BIG>
BOLD + ITALICIZED + BIG	<BOLD><I><BIG>
BOLD + CITATION + BIG	<BOLD><CITE><BIG>
BOLD + COMPUTER CODE + BIG	<BOLD><CODE><BIG>
BOLD + KEYBOARD INPUT + BIG	<BOLD><KBD><BIG>
BOLD + LITERAL + BIG	<BOLD><SAMP><BIG>
BOLD + VARIABLE + BIG	<BOLD><VAR><BIG>
BOLD + TYPEWRITER + BIG	<BOLD><TT><BIG>
ITALICIZED + CITATION + BIG	<I><CITE><BIG>
ITALICIZED + COMPUTER CODE + BIG	<I><CODE><BIG>
ITALICIZED + KEYBOARD INPUT + BIG	<I><KBD><BIG>
ITALICIZED + LITERAL + BIG	<I><SAMP><BIG>
ITALICIZED + VARIABLE + BIG	<I><VAR><BIG>
ITALICIZED + TYPEWRITER + BIG	<I><TT><BIG>
STRONG + EMPHASIS + SMALL	<STRONG><EM><SMALL>

TTAG or FTAG Value	HTML Tag or Tags Enclosing the SAS TITLE or SAS FOOTNOTE
STRONG + ITALICIZED + SMALL	<STRONG><I><SMALL>
STRONG + CITATION + SMALL	<STRONG><CITE><SMALL>
STRONG + COMPUTER CODE + SMALL	<STRONG><CODE><SMALL>
STRONG + LITERAL + SMALL	<STRONG><SAMP><SMALL>
STRONG + VARIABLE + SMALL	<STRONG><VAR><SMALL>
STRONG + TYPEWRITER + SMALL	<STRONG><TT><SMALL>
EMPHASIS + CITATION + SMALL	<EM><CITE><SMALL>
EMPHASIS + COMPUTER CODE + SMALL	<EM><CODE><SMALL>
EMPHASIS + KEYBOARD INPUT + SMALL	<EM><KBD><SMALL>
EMPHASIS + LITERAL + SMALL	<EM><SAMP><SMALL>
EMPHASIS + TYPEWRITER + SMALL	<EM><TT><SMALL>
BOLD + EMPHASIS + SMALL	<BOLD><EM><SMALL>
BOLD + ITALICIZED + SMALL	<BOLD><I><SMALL>
BOLD + CITATION + SMALL	<BOLD><CITE><SMALL>
BOLD + COMPUTER CODE + SMALL	<BOLD><CODE><SMALL>
BOLD + KEYBOARD INPUT + SMALL	<BOLD><KBD><SMALL>
BOLD + LITERAL + SMALL	<BOLD><SAMP><SMALL>
BOLD + VARIABLE + SMALL	<BOLD><VAR><SMALL>
BOLD + TYPEWRITER + SMALL	<BOLD><TT><SMALL>
ITALICIZED + CITATION + SMALL	<I><CITE><SMALL>
ITALICIZED + COMPUTER CODE + SMALL	<I><CODE><SMALL>
ITALICIZED + KEYBOARD INPUT + SMALL	<I><KBD><SMALL>
ITALICIZED + LITERAL + SMALL	<I><SAMP><SMALL>
ITALICIZED + VARIABLE + SMALL	<I><VAR><SMALL>
ITALICIZED + TYPEWRITER + SMALL	<I><TT><SMALL>

Arguments for Character Transcoding

The following arguments allow you to specify a character set or convert character data to the corresponding Unicode Numeric Character Reference (NCR).

CHARSET=*char-set-name*
specifies the character set name that will be written into the META tag of the HTML output file. For information on available character set names, see http://www.iana.org/assignments/character-sets.

Used by: DS2TREE, DS2CONST, META2HTM

TRANLIST=*transcoding-list-name*
specifies the name and location of an existing transcoding list, either user-defined or from SAS. The transcoding list name must be a four-level name, and the fourth level must be SLIST, as in the following example:

```
TRANLIST=SASHELP.HTMLGEN.IDENTITY.SLIST
```

This argument is required if you are implementing character transcoding.

SAS provides a number of transcoding lists in the SASHELP.HTMLNLS catalog. For a description of these transcoding lists, and for information on generating your own transcoding lists, see the SAS Web site at http://support.sas.com/rnd/web/intrnet/format/lang2.html.

Used by: DS2TREE, DS2CONST, META2HTM

Reserved Names

Do not use the following names as the value of a macro variable:

Libnames and Filerefs

HTML

CATENT

HTMSS

Global Macro Variables

_htmovp

_htmcap

_htmtitl

_htmwher

Data Sets or Views

WORK._BYGRP

Catalogs

WORK._HTMLG_

SASHELP.HTMLNLS

Catalog Entries

SASHELP.HTMLGEN.DSPROP.SLIST

SASHELP.HTMLGEN.IDENTITY.SLIST

SASHELP.HTMLGEN.OUTPROP.SLIST

SASHELP.HTMLGEN.TABPROP.SLIST

SASHELP.HTMLGEN.TAGS.SLIST

CHAPTER 27

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 596
- Displaying pop-up text when the mouse pointer is over a portion of the diagram. See "Data Tips for Web Presentations" on page 598
- Adding drill-down functionality that enables you to link to other Web pages. See "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

Table 27.1 Support for Chart Descriptions, Data Tips, and Drill-Down Functionality

	Chart Descriptions		Data Tips		Drill-Down Links	
Device	Generated by default	Can be customized	Generated by default	Can be customized	Generated by default	Can be Customized
GIF	X	X		X		X
JPEG	X	X		X		X
PNG	X	X		X		X
SVG family	X	X		X		X
ACTIVEX			X	X		X
JAVA			X	X		X
ACTXIMG	X	X	X	X		X
JAVAIMG	X	X	X	X		X

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 ALT tag of 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 graphic elements. See "ACCESSIBLE" on page 328 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. 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.

```
goptions reset=all border cback="#FFFFFF";
axis1 label=("Population Est.") minor=(n=2);
symbol v=dot;
ods listing close;
ods html;
proc gplot data=sashelp.citiyr;
   format pan comma7.;
   plot pan*date/ vaxis=axis1 autovref
      description="This is the DESCRIPTION= option text.";
run;
quit;
ods html close;
ods listing;
```

Figure 27.1 Plot of Two Variables with a Custom Chart Description

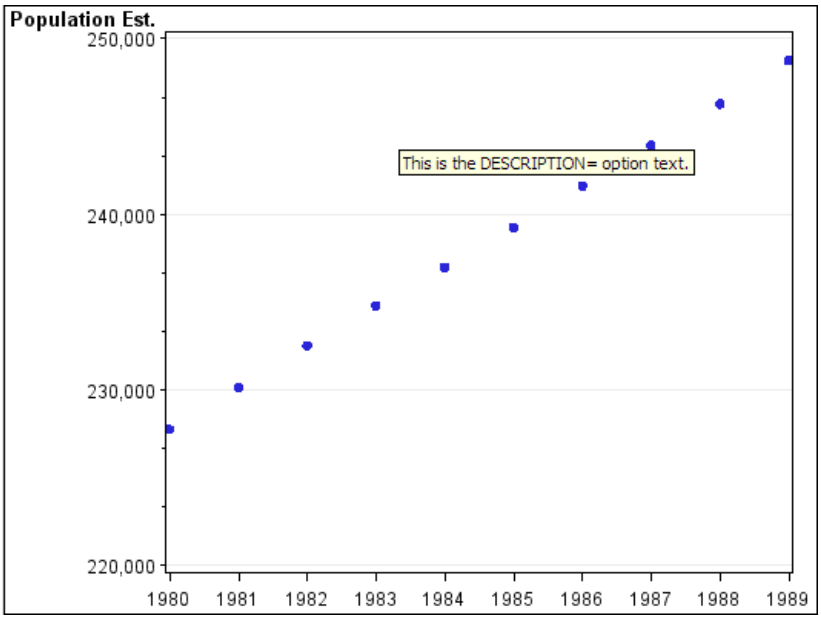


Chart Descriptions in GIF, JPG, PNG, ACTXIMG, and JAVAIMG Presentations

For output generated with the GIF, JPG, PNG, 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 ActiveX, Java, and JAVAMETA 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 created by default, and custom data tips are supported when using the HTML output destination, in combination with certain device drivers.

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 data tip values to the data tip variable using the following form:

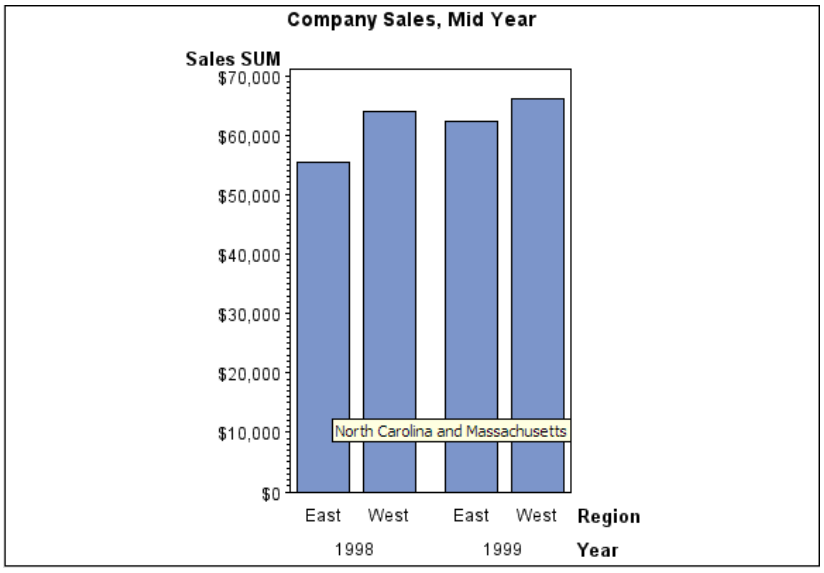
   ```
   'alt="data tip"'
   ```

   .
3. Add HTML=*data-tip-variable* to your procedure's statement. The example below specifies **HTML=RPT**.

When the user positions the mouse pointer over a data element, the browser displays the data tip. The following example generates the data tips **North Carolina and Massachusets** and **California and Oregon**.

```
/* 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 statement to assign values to the variable rpt. */
data;
   set data;
      if state in ("NC" "MA") then RPT="alt='North Carolina and Massachusets'";
      if state in ("CA" "OR") then  RPT="alt='California and Oregon'";
run;

/* Close the LISTING destination to conserve resources. */
/* Open the HTML destination and create the bar chart.  */
/* Add the HTML= option to associate custom data tips   */
/* with each graph element.                             */

goptions reset=all;
ods listing close;
ods html file="datatips.htm";
title "Company Sales, Mid Year";
proc gchart data=sales;
   vbar region / sumvar=sales
      group=year
      html=RPT;
run;
quit;
ods html close;
ods listing;
```

Figure 27.2 Bar Chart with Custom Data Tips



Data Tips in GIF, JPEG, PNG, JAVAMETA, SVG, SVGT, SVGView, and SVGZ Presentations

For output generated with the GIF, JPEG, PNG, JAVAMETA, and SVG, SVGT, SVGView, and SVGZ device drivers, data tips are not generated by default.

Custom data tips can be implemented for the output of any SAS/GRAPH procedure that supports the HTML= option. For procedures that support the HTML= option, refer to the individual procedure chapter.

For more information, see “Adding Custom Data Tips with the HTML= Option” on page 598.

Data Tips in ACTIVEX, ACTXIMG, JAVA, and JAVAIMG Presentations

For output generated with the ACTIVEX, ACTXIMG, JAVA, and JAVAIMG device drivers, data tips are created by default and 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")
```

Custom data tips can be implemented for the output of any SAS/GRAPH procedure that supports the HTML= option. For procedures that support the HTML= option, refer to the individual procedure chapter. For more information, see “Adding Custom Data Tips with the HTML= Option” on page 598.

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 with the HTML= and HTML_LEGEND= Options

The 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 to Web presentations that are generated with the following device drivers:

- GIF, JPEG, OR PNG
- JAVA and ACTIVEX
- JAVAMETA

In these Web presentations, the HTML= and HTML_LEGEND= options identify a variable that provides URLs for drill-down links. This variable is referred to as a *link variable*.

The HTML= option and HTML_LEGEND= options are also used to implement enhancements that run in the Metaview applet. In this case, the variables that are identified by the HTML= and HTML_LEGEND= options are referred to as *enhancement variables* because they do more than establish links.

Working with Link and Enhancement Variables

To use link or enhancement variables in a Web presentation, you need to define those variables, add data to those variables, and then identify those variables in the HTML= option or the HTML_LEGEND= options, or both.

The following code fragment defines a link variable named RPT and assigns that variable a length of 40 characters.

```
data regsales;
   input Region State Sales;
   length RPT $40;
```

Be sure to define your link variable with a length that will be sufficient to contain your URLs (plus the HREF= option). There is no limit on the length of the variable.

The values of the link variable use the following syntax:

```
"HREF='URL<#anchor name'>"
```

The syntax is used in the following example:

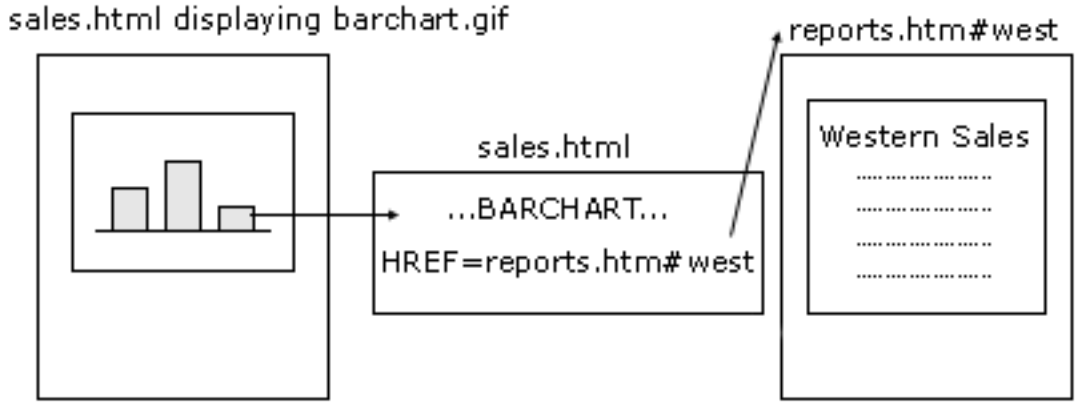
```
RPT="href='reports.html#west' "
```

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 which case you can use IF/THEN statements or variable substitution.

The following picture 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 27.3 Links in Drill-Down Graphs



Each bar in the chart links to an anchor tag in an HTML file named reports.html. The anchor names in the linked file are "Central," "South," and "West." The following DATA step the IF/ THEN statement to assign values to the the link variable.

```
/* 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;
 if Region="Central" then
  rpt="HREF='reports.htm#central'";
 else if Region="South" then
  rpt="HREF='reports.htm#south'";
 else if Region="West" then
  rpt="HREF='reports.htm#west'";
run;

goptions reset=all device=ActiveX;

ods listing close;
ods html file="sales.htm";
```

```
/* create chart that uses link targets */
title "Regional Sales";
proc gchart data=regsales;
   vbar region / sumvar=sales
     html=rpt;
run;

/* create the link targets */
ods html file="reports.htm" anchor="south";
title "Southern Sales";
proc gchart data=regsales;
 where region="South";
 vbar state /sumvar=sales;
run;

ods html anchor="central";
title "Central Sales";
proc gchart data=regsales;
 where region="Central";
 vbar state /sumvar=sales;
run;

ods html anchor="west";
title "Western Sales";
proc gchart data=regsales;
 where region="West";
 vbar state /sumvar=sales;
run;
quit;
ods html close;
ods listing;
```

Display 27.1 REGSALES Data Set

Region	State	Sales	rpt
West	CA	$13,636	href="reports.html#west"
West	OR	$18,988	href="reports.html#west"
West	WA	$14,523	href="reports.html#west"
Central	IL	$18,038	href="reports.html#central"
Central	IN	$13,611	href="reports.html#central"
Central	OH	$11,084	href="reports.html#central"
Central	MI	$19,660	href="reports.html#central"
South	FL	$14,541	href="reports.html#south"
South	GA	$19,022	href="reports.html#south"

You could use variable substitution to simplify the DATA step. The URLs used in the preceding program all use the same HTML file name, but the anchor differs depending on the value of the Region variable. You can concatenate the value of the Region variable to the common HTML file name to generate the drill-down URLs.

```
data regsales;
 set regsales;
 rpt="HREF='reports.htm#"||Region||"'";
run;
```

Links in GIF, JPEG, PNG, and SVG Presentations

To add drill-down functionality to images generated with the GIF, JPEG, PNG, SVG, SVGT, SVGZ, and SVGVIEW device drivers, do one of the following:

- 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.

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 with the HTML= and HTML_LEGEND= Options” on page 601.

Links in ACTIVEX Presentations

To add drill-down functionality to the ActiveX control created with the ACTIVEX device driver, use the HTML= option as described in "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

Links in JAVA Presentations

The Graph applet is a Java applet that provides drill-down functionality by default. The following code generates a graph using DEVICE=JAVA. When the Web page is displayed, drill-down functionality is enabled. The Graph applet retains the type and style of the initial graph for all the graphs in the presentation. The result is a sequence of three-dimensional, vertical bar charts that use the ODS style GEARS.

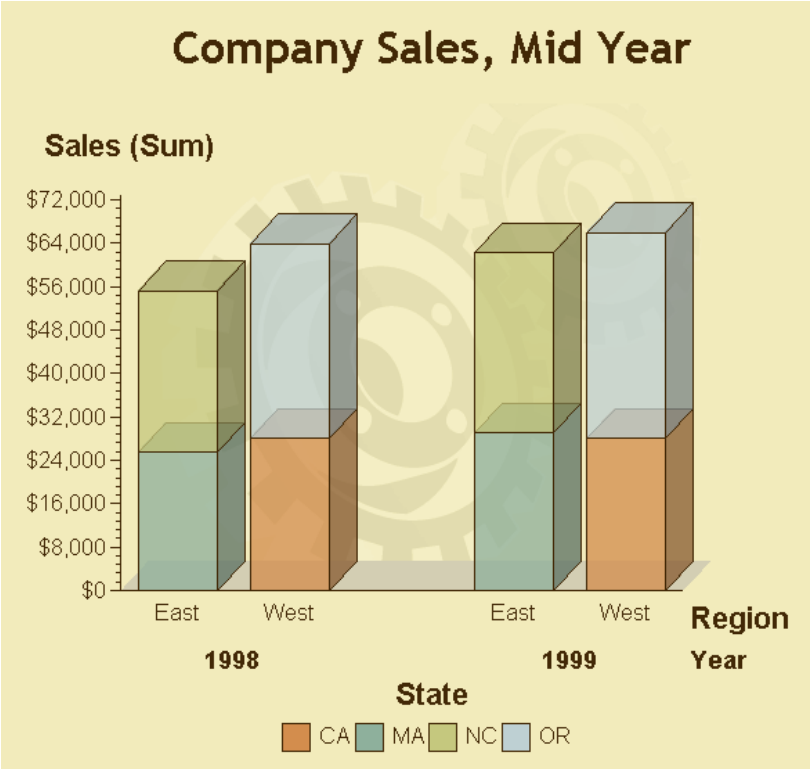
```
data sales;
   length Region $ 4 State $ 2;
   format Sales dollar8.;
   input Region State Sales Year Qtr;
   datalines;
West CA 13636 1999 1
West OR 18988 1999 1
West CA 14523 1999 2
West OR 18988 1999 2
East MA 18038 1999 1
East NC 13611 1999 1
East MA 11084 1999 2
East NC 19660 1999 2
West CA 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
;
goptions reset=all device=java;
ods listing close;
ods html file="vbarweb.htm" style=gears;
title "Company Sales, Mid Year";
proc gchart data=sales;
   vbar3d region / sumvar=sales
   group=year subgroup=state;
run;
quit;
ods html close;
ods listing;
```

The initial graph displayed by this example is shown in the example below. In this graph, REGION is the independent variable, and SALES is the dependent variable.

- SALES is the dependent variable.
- REGION is the independent variable.
- STATE is the subgroup variable.

□ YEAR is the group variable.

Figure 27.4 Graph Applet: Level 1

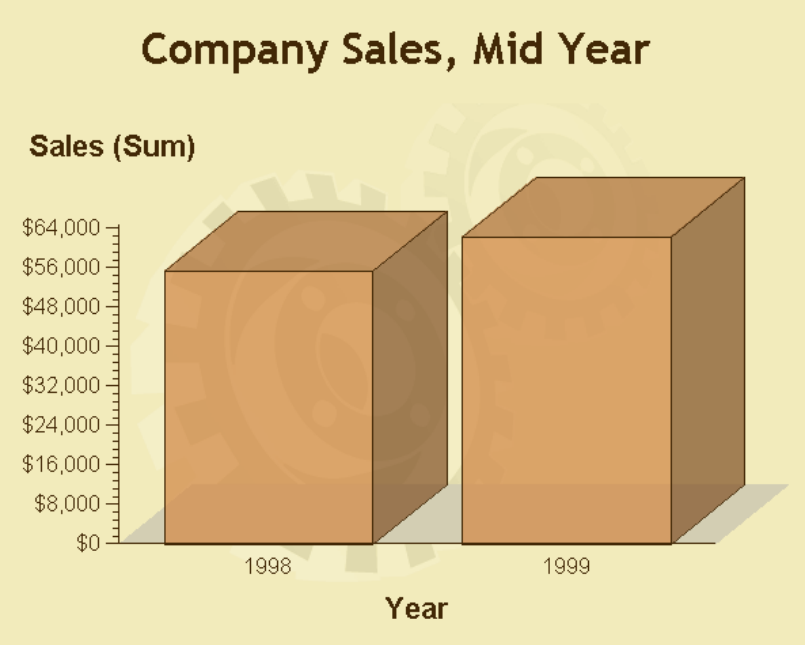


Clicking the bar segment labeled East generates the graph shown in Figure 27.5 on page 607. The Level 2 drill-down graph retains the dependent variable SALES. The group variable YEAR is promoted to the independent variable role. The drill-down action creates one bar segment for each unique value of YEAR.

□ SALES is the dependent variable.

□ YEAR is the independent variable.

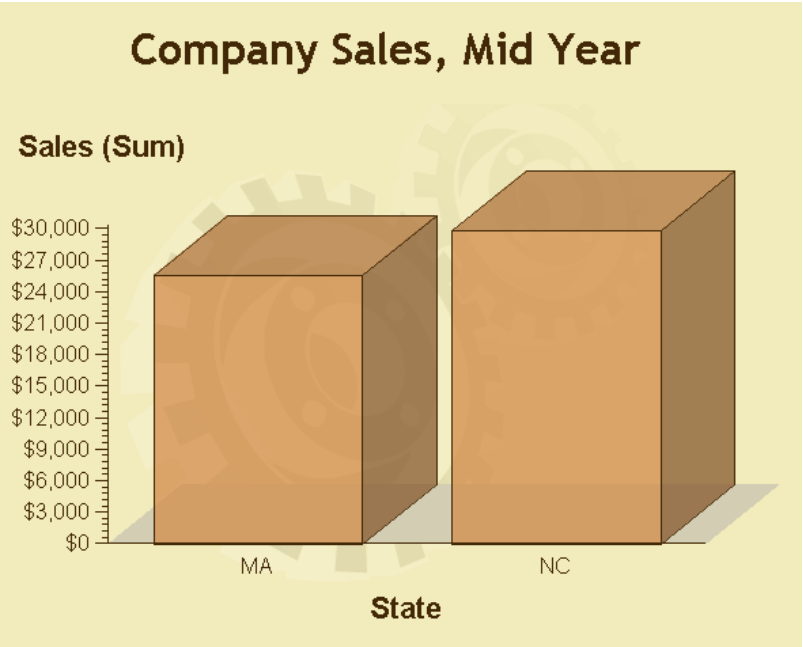
Figure 27.5 Graph Applet: Level 2



Clicking the bar segment labeled 1998 generates the graph shown in Figure 27.6 on page 607. The Level 3 drill-down graph retains the dependent variable SALES. The subgroup variable STATE is promoted to the independent variable role. STATE is the last variable that can appear as an independent variable. The drill-down action creates one bar segment for each unique value of STATE.

- SALES is the dependent variable.
- STATE is the independent variable.

Figure 27.6 Graph Applet: Level 3



Links in Metaview Applet Presentations

To generate drill-down presentations for the Metaview applet use either the HTML= or the HTML_LEGEND= options or both and an enhancement variable, as introduced in "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

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.

Controlling Drill-Down Behavior For ActiveX and Java Using Parameters

You can use parameter tags on the ODS HTML statement to specify drill-down behavior for the ActiveX control, the Graph applet, or the Map applet, in the ODS HTML statement. Parameters are specified on the ODS destination statement with the PARAMETERS= option as follows:

```
ODS HTML PARAMETERS=(options);
```

See Chapter 19, "Attributes and Parameters for Java and ActiveX," on page 485 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 Graph applet, Map applet, or ActiveX control. The following table defines the drill-down tags and explains the types of graphs to which the tags can be applied.

Table 27.2 Drill-Down Tags Used by the Graph Applet, Map Applet, and 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
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

Tag Name	Description	Definition of the Value That Follows the Tag	Applied in
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, that name must be specified exactly the way it appears in the data set, because 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"="LOCAL"|"SCRIPT"|"URL"|"HTML");
```

Local mode
: responds to drill-down actions by dynamically generating and displaying new graphs. The data in the initial graph is subset based on the graph element that was selected in the drill-down action. The user can select another graph element to generate another graph. Another graph is generated as long as the data can still be subset, or you have configured your own levels of drill-down.

  To configure a graph at a given level, you specify the applet parameter DDLEVEL*n*. The value of this parameter determines the graph type, data subset,

variable roles, and colors. Local is the default drill-down mode for the Graph applet.

Featured in: "Local Drill-Down Mode with Java" on page 475.

Restriction: Supported by the Graph applet only.

See also: "Links in JAVA Presentations" on page 605.

Script mode
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.

Featured in: "Script Drill-Down Mode with Java" on page 477, "Providing JavaScript Drill-Down with ActiveX" on page 464, and "Providing More JavaScript Drill-Down with ActiveX" on page 466.

Restriction: Supported by the Map applet and ActiveX control only.

See also: "Configuring Script Drill-Down Mode" on page 616.

URL mode
displays URLs that are provided by the HTML= variable. The URLs identify HTML files.

Featured in: "URL Drill-Down Mode with Java" on page 479.

See also: DRILLDOWNMODE=HTML in "Parameter Definitions" on page 491.

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.

HTML mode
generates drill-down URLs based on a substitution pattern that you specify in your SAS/GRAPH program. The ActiveX control, the Graph applet, and the Map applet complete the URL by inserting the specified data from the selected graph element.

```
ods html file=statepop.htm
 parameters=("DRILLDOWNMODE"="HTML"
 "DRILLPATTERN"="http://www.state.{&statename}.us");
```

The data set variable value STATENAME completes the drill-down URL.

Featured in: "HTML Drill-Down Mode" on page 482.

See also: "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

Note: Define the variable with the partial URL when creating the graphic. △

Any mode
attempts to implement the four drill-down modes in succession until a valid Web destination is found. The order is Local (Graph applet only), Script, URL, and HTML.

Restriction: Supported by Graph applet and ActiveX control only.

See: "Specifying Parameters and Attributes for Java and ActiveX" on page 485 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 applet or 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 616) and HTML (see "Configuring the Drill-Down Response In HTML and URL Modes" on page 615) make use of 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 varialbe name as a substitution string in the HTML drill-down mode, the applet or 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 610. 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 applet or 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 applet 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 that are defined in the data set.

See “URL Drill-Down Mode with Java” on page 479 for more information.

Configuring HTML Drill-Down Mode

You can use the parameters DRILLDOWNMODE, DRILLPATTERN, PATTERNSTRIP, and DRILLTARGET to configure the HTML drill-down mode for the ActiveX control, the Graph applet, and the Map applet.

In the HTML drill-down mode, the applet or 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 "Parameter Reference for Java and ActiveX" on page 488) establishes the HTML drill-down mode. The PATTERNSTRIP parameter (see "Removing Blank Spaces from Data Values In Substitution Strings" on page 611) 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 612) enables you to specify where you want the drill-down graph to appear in the browser.

Specify the DRILLPATTERN parameter in the ODS statement:

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"="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 applet or control dynamically generates URLs in response to drill-down actions. The applet or 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* 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 612. All substitution strings are enclosed in curly brackets ({ and }) and begin with an ampersand character (&).

When you specify a variable name as a substitution string in drill-down mode, the applet or control replaces the string with the variable value.

Specifying Graphs For Each Drill-Down Level

The DDLEVEL*n* parameter lets you specify the graphs that are generated at each drill-down level. The DDLEVEL*n* parameter is specified as follows in the ODS statement:

ODS *HTML*

PARAMETERS=("DDLEVEL*n*"="*string*");

n

represents the number of the drill-down level that is being configured.

string

- specifies the graph type.
- names the variable roles.
- specifies the color of the data elements.

- names the variable to subset, to create the next graph.

The syntax of the *string* argument is as follows:

{CHART} {*chart_type*} {*tag_1*} {*variable_1...}* {*...tag_n*} {*variable_n*} | {*subset_tag_1...*} <{*...subset_tag_n*}>

{CHART} {*chart_type*}
identifies the type or style of the graph. This tag is case sensitive: it must always be specified in uppercase. The values of the tag (chart types) are not case sensitive. To use the same chart type as the preceding drill-down level, do not specify the CHART tag. Available chart types are as follows:

HBAR
generates a two-dimensional horizontal bar chart.

HBAR3D
generates a three-dimensional horizontal bar chart.

VBAR
generates a two-dimensional vertical bar chart.

VBAR3D
generates a three-dimensional vertical bar chart.

PIE
generates a two-dimensional pie chart.

PIE3D
generates a three-dimensional pie chart.

SCATTER
generates a scatter plot that is similar in appearance to the plot shown in Example 4 on page 1564.

LINE
generates a line or needle plot that is similar in appearance to Figure 14.17 on page 261.

BOX
generates a box plot that is similar in appearance to Figure 14.15 on page 257.

HILO
generates a high-low chart that is similar in appearance to Figure 14.16 on page 259.

{*tag_1*} {*variable_1...*} {*...tag_n*} {*variable_n*}
associates drill-down tags with data set variables, to specify roles for variables in the new graph, and to determine the color of the elements in the new graph (optional). For definitions of the drill-down tags, see Table 19.1 on page 488.

{*subset_tag_1...*} <{*...subset_tag_n*}>
specifies one or more variable roles from the original graph whose values are used to subset the data in the preceding graph. If you specify G_GROUPV, then the data that is used to draw the new graph, is only the data that is associated with the group variable in the preceding graph. If the group variable in the preceding graph is REGION, and the data element labeled East is selected, only observations where REGION=EAST are represented in the next graph.

At least one of the following tags must be specified as the subset variable: G_INDEPV, G_GROUPV, G_SUBGRV, or G_DEPTHV. For definitions of these tags, see Table 19.1 on page 488.

Specifying multiple subset variables means that two or more values must match the value in the selected graph element for that observation to be used in the new

graph. For example, assuming that you specify {G_INDEPV}{G_SUBGRV} as the subset variables, and that the selected graph element has an independent variable of YEAR and a subgroup variable of STATE. Also assume that the values for these variables in the selected graph element were 2000 and NC. The observations that would be used in the drill-down graph would include those with YEAR=2000 and STATE=NC.

The following example shows how the DDLEVEN*n* parameter can be used to specify the default behavior for the first drill-down level.

```
ods html file=odsout
  parameters=("drilldownmode"="local"
        "ddlevel1"="{chart}{vbar3d}
              {g_dep}{sales}
              {g_indep}{year} |
              {g_indepv}" );
```

As the example shows, the value of the DDLEVEL*n* parameter is divided into two halves, which are separated by a vertical bar character. The drill-down graph is configured in the syntax that appears before the vertical bar character (|). After the vertical bar, drill-down tags specify how the data from the previous drill-down level is to be subset for use in the current drill-down graph.

The first drill-down level (DDLEVEL1) is configured as a three-dimensional vertical bar chart. The dependent variable is SALES and the independent variable is YEAR. The G_INDEPV tag specifies that the data is to be based on the values of the independent variable. In our example the independent variable in the initial graph is REGION. If the Web user selects a graph element that describes the WEST region, the graph has only observations where the value of REGION is WEST.

If you do not specify a role for a variable, then that variable does not appear in the drill-down graph. If you do not specify variables for the G_DEP and G_INDEP tags, then the Graph applet uses the independent and dependent variables of the graph in the preceding drill-down level.

You can explicitly remove a variable role from the drill-down graph by specifying a $ character as the drill-down value, as in the following code:

```
{G_GROUP} {$}
```

Web users can make this change in the Graph applet menus by selecting the None option from the list of variables that can be applied to a given variable role.

Note: Note that you cannot assign a $ to the G_INDEP and G_DEP variables, because they must always be present in the drill-down graph. △

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 applet or control displays drill-down graphs in a new Web browser window.

Specify the DRILLTARGET parameter as follows, using the PARAMETERS= option in the ODS statement:

ODS *HTML*
  PARAMETERS=(“DRILLTARGET”=
  “_BLANK” | “_SELF” | “_PARENT” | “_TOP” | *any_named_target*)

_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.

any_named_target
: displays the drill-down graph in the appropriately named frame or browser window.

Configuring Script Drill-Down Mode

You can use the parameters DRILLDOWNMODE, DRILLFUNC, PATTERNSTRIP, and DRILLTARGET to configure the Script drill-down mode for the ActiveX control, the Graph applet, and the Map applet. 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 follows:

ODS *HTML*

PARAMETERS=(“DRILLDOWNMODE”=“SCRIPT” “DRILLFUNC”=“*method*”);

The applet parameter DRILLDOWNMODE (see “Specifying the Drill-Down Mode” on page 609) 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 applet or 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 applet or control ignores this object.

To invoke the callback method, the applet or 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 applet or 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 608 and “Understanding Variable Roles” on page 610.

The first element in the array of arguments is the name of the applet or 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 filename 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 filename and file type are provided as a convenience. If you use this filename 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*

For example assume that each graph data element selected is represented by six arguments in the array.

The graph shown in "Script Drill-Down Mode with Java" on page 477 is configured for Script drill-down mode. Selecting the east region sales figures for the state of North Carolina generates the following array:

```
[appletName East1998NC.html
G_DEP Sales G_DEPV 10000 G_DEPV,F $10,000
G_INDEP Region G_INDEPV East G_INDEPV,F East
G_GROUP Year G_GROUPV 1998 G_GROUPV,F 1998
G_SUBGR State G_SUBGRV NC G_SUBGRV,F NC]
```

The output filename is East1998NC.html. The remaining triplet pairs capture the roles, and values of the variables that make up the selected data element. All variable names are case sensitive as they appear in the array. For example, the value Region is capitalized. This is the case only if the variable name is defined as Region in the DATA step.

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 an example of implementing script drill-down mode, see "Script Drill-Down Mode with Java" on page 477. For information on 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.

Disabling Drill-Down Functionality

For the Graph applet, you can specify the DISABLEDRILLDOWN parameter to disable the drill-down functionality. Specify the DRILLDOWNMODE parameter as follows in the ODS statement:

```
ODS HTML
  PARAMETERS=("DISABLEDRILLDOWN"="TRUE");
```

Example: Creating Bar Charts with Drill-Down for the Web

This example shows how to create 3-D bar charts with drill-down functionality for the Web. In addition to showing how to use the ODS HTML statement and the HTML options to create the drill-down, the example also illustrates other VBAR3D statement options.

For creating output with drill-down functionality for the Web, the example shows how to do the following tasks:

- explicitly name the HTML files and open and close them throughout the program
- specify names and destination for the GIF files created by the ODS HTML statement and the GIF device driver
- assign anchor names to the graphics output
- use the HTML= and HTML_LEGEND= procedure options to assign link targets
- use BY-group processing to store multiple graphs in one file or in individual files
- increment the anchor names and increment the file names

For more information, see "ODS HTML Statement" on page 239 in Chapter 14, "SAS/GRAPH Statements," on page 197.

For creating 3-D bar charts, the example shows how to do the following tasks:

- group the midpoints, including patterning bars by group, modifying the group axis, adjusting the space between groups of bars
- identify midpoint values with a legend instead of labeling each bar
- subgroup bars
- remove an axis and its axis plane
- add reference lines

The introduction to each part lists the VBAR3D options that it features.

Procedure Features:

VBAR3D statement

ODS HTML options:

ANCHOR=

BODY=

CONTENTS=

FRAME=
NEWFILE
NOGTITLE
PATH=

Other Features:

AXIS statement
BY statement
FORMAT statement
GOPTIONS statement option: BORDER
LEGEND statement
RUN-group processing
TITLE statement
WHERE statement

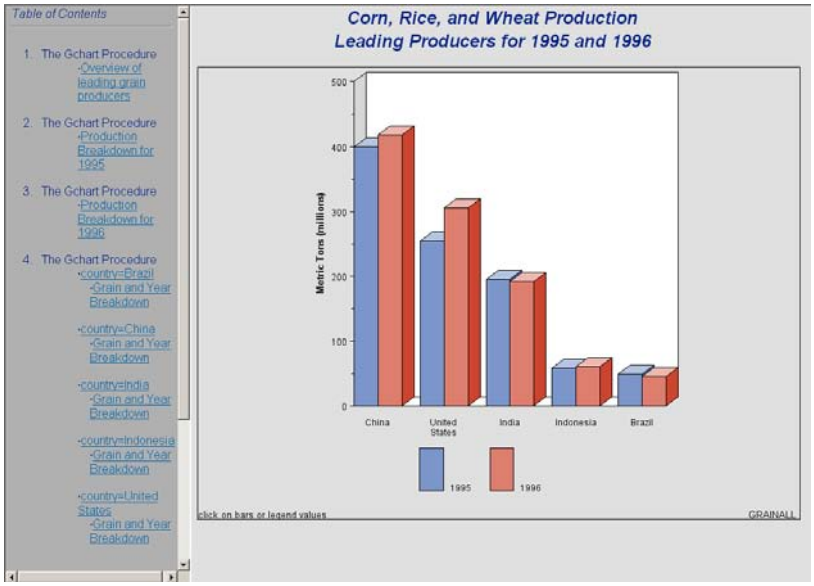
Sample library member:

GCHDDOWN

The program generates twelve linked bar charts that display data about the world's leading grain producers. The data contain the amount of grain produced by five countries in 1995 and 1996. Each of these countries is one of the three leading producers of wheat, rice, or corn, worldwide.

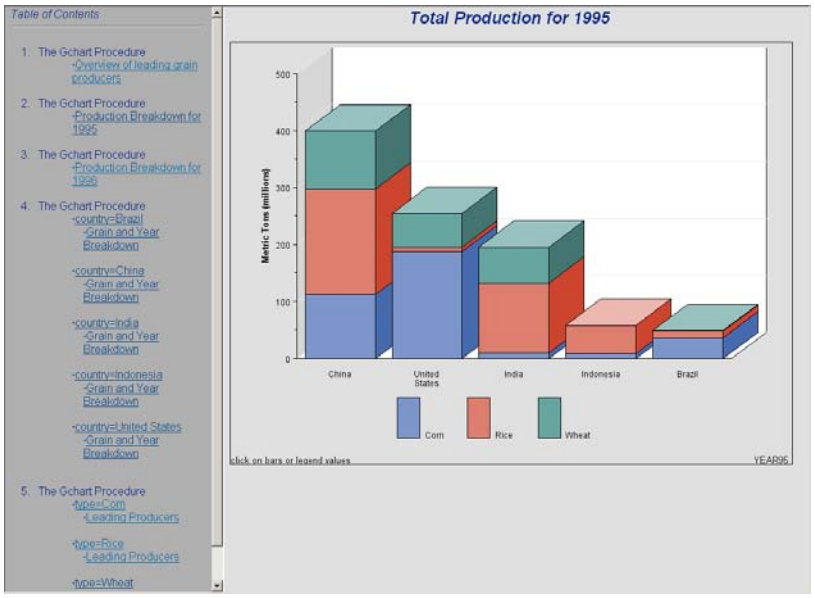
The first chart, shown in Figure 27.7 on page 619 as it appears in a browser, is an overview of the data that shows the total grain production for the five countries for both years.

Figure 27.7 Browser View of Overview Graph



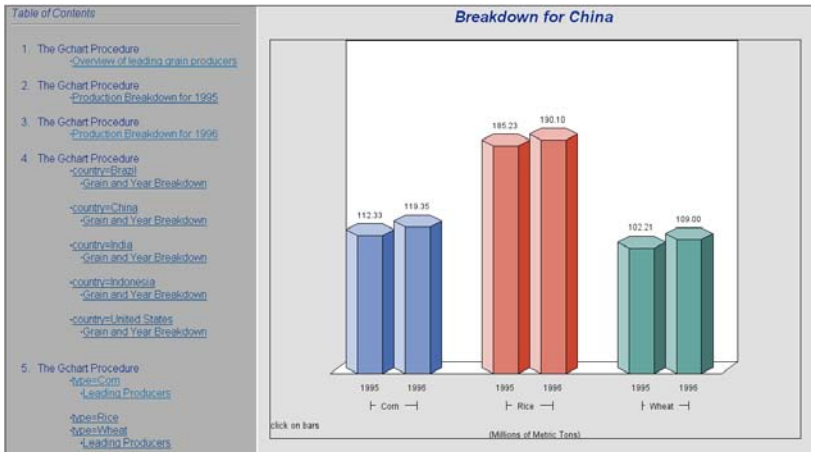
The next two charts break down grain production by year. These charts are linked to the legend values in Figure 27.7 on page 619. For example, when you select the legend value for 1995, the graph in Figure 27.8 on page 620 appears.

Figure 27.8 Browser View of Year Breakdown for 1995



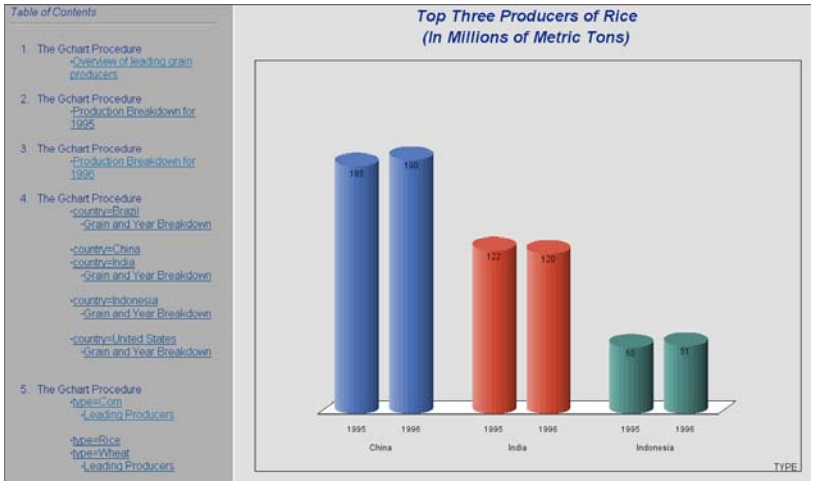
Another group of charts breaks down the data by country. These charts are linked to the bars. For example, when you drill down on the bar for China in either Figure 27.7 on page 619 or Figure 27.8 on page 620, the graph in Figure 27.9 on page 620 appears.

Figure 27.9 Browser View of Breakdown for China



Finally the data is charted by grain type. These graphs are linked to the bars in Figure 27.9 on page 620. If you select the legend value or bar for **`Rice`**, Figure 27.10 on page 621 appears.

Figure 27.10 Browser View of Breakdown for Rice



This program is divided into four parts:

- "Example Part A" on page 621 generates the graph shown in Figure 27.7 on page 619.
- "Example Part B" on page 626 generates the pair of graphs represented by Figure 27.8 on page 620.
- "Example Part C" on page 628 generates the five graphs represented by Figure 27.9 on page 620.
- "Example Part D" on page 630 generates the three graphs represented by Figure 27.10 on page 621.

Example Part A

VBAR3D options:

DES=
DISCRETE
GROUP=
GSPACE=
HTML=
HTML_LEGEND=
NAME=
SUBGROUP=

ODS HTML options:

BODY=
CONTENTS=
FRAME=
GPATH=
NOGTITLE

The first part of the program, which includes setting the graphics environment and creating the data set, does the following:

- Adds three HTML variables to the data set. The variables contain the link targets for all of the graphs that support drill-down functionality. The HREF values for the HTML variables in the data set contain this information about the link targets:
  - the name of the body file that is the target. BODY= in the ODS HTML statement names the body file.
  - the anchor name of the output if the target file contains more than one graph. By default, all output is assigned a unique anchor name unless you specify a name with ANCHOR= in the ODS HTML statement.
- Opens the HTML destination for the frame and contents files and the first body file.
- Creates one grouped 3-D vertical bar chart (shown in Figure 27.7 on page 619) with drill-down on the bars and legend values. The bars, which represent total production for each year for each country, are grouped and labeled by COUNTRY. Instead of displaying the year below each bar, the program suppresses the midpoint values with an AXIS statement and creates a legend that associates bar color and year. To create the legend, the chart variable YEAR is assigned to the SUBGROUP= option. Because the chart variable and the subgroup variable are the same, each bar contains only one "subgroup." As a result, the subgroup legend has an entry for each value of YEAR, thereby creating a legend for the midpoints. The values of COUNTRY label each group of bars.
- Assigns the HTML variables that contain link information for the bars and for the legend values to the HTML= and HTML_LEGEND= options, respectively.

Assign the Web path. FILENAME assigns the fileref ODSOUT, which specifies a destination for the HTML and GIF files produced by the example program. To assign that location as the HTML destination for program output, ODSOUT is specified later in the program in the ODS HTML statement's PATH= option. ODSOUT must point to a Web location if procedure output is to be viewed on the Web.

```
filename odsout "c:\";
```

Set the graphics environment. The BORDER goption draws a black border around the graph.

```
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 IND Wheat 63007
1995 IND Rice 122372
1995 IND Corn 9800
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 CHN Wheat 109000
1996 CHN Rice 190100
1996 CHN Corn 119350
1996 IND Wheat 62620
1996 IND Rice 120012
1996 IND Corn 8660
1996 INS Wheat .
1996 INS Rice 51165
1996 INS Corn 8925
1996 USA Wheat 62099
1996 USA Rice 7771
1996 USA Corn 236064
;
```

Add three HTML variables to GRAINLDR to create the NEWGRAIN data set. Each HTML variable is assigned the targets for a certain variable value. These targets are specified by the HREF attribute within an AREA element in the HTML file. Each HREF value specifies the HTML body file and, can also reference the name of the anchor within the body file that identifies the target graph. The HTML variable YEARDRILL contains the targets for the values of the variable YEAR.

```
data newgrain;
set grainldr;
length yeardrill typedrill countrydrill $ 40;
if year=1995 then
yeardrill="HREF='year95_body.html'";
else if year=1996 then
yeardrill="HREF='year96_body.html'";
```

The HTML variable COUNTRYDRILL contains the targets for the values of the variable COUNTRY. Because the graphs of COUNTRY are in one file, the targets must include the anchor name.

```
     if country="BRZ" then
countrydrill="HREF='country_body.html#country'";
else if country="CHN" then
```

```
countrydrill="HREF='country_body.html#country1'";
else if country="IND" then
countrydrill="HREF='country_body.html#country2'";
else if country="INS" then
countrydrill="HREF='country_body.html#country3'";
else if country="USA" then
countrydrill="HREF='country_body.html#country4'";
```

The HTML variable TYPEDRILL contains the names of the files that are the targets for the values of the variable TYPE.

```
if type="Corn" then
typedrill="HREF='type1_body.html'";
else if type="Rice" then
typedrill="HREF='type2_body.html'";
else if type="Wheat" then
typedrill="HREF='type3_body.html'";
run;
```

Create a format for the values of COUNTRY.

```
proc format;
     value $country "BRZ" = "Brazil"
                    "CHN" = "China"
                    "IND" = "India"
                    "INS" = "Indonesia"
                    "USA" = "United States";
run;
```

Define legend characteristics for all legends. OFFSET= moves the legend down.

```
legend1 label=none
        shape=bar(4,4)
        position=(bottom center)
        offset=(-3);
```

Assign the GOPTIONS for ODS HTML destination. DEVICE= generates the SAS/GRAPH output as GIF files.

```
goptions device=gif;
```

Open the ODS HTML destination for the ODS graphics output. BODY= names the file for storing the HTML output. CONTENTS= names the HTML file that contains the table of contents to the HTML procedure output. The contents file links to each of the body files written to the HTML destination. FRAME= names the HTML file that integrates the contents and body files. PATH= specifies the ODSOUT fileref as the HTML destination for all the HTML and GIF files. NOGTITLE suppress the graph titles from the SAS/GRAPH output and displays them through the HTML page.

```
ods html body="grain_body.html"
frame="grain_frame.html"
contents="grain_contents.html"
path=odsout
nogtitle;
```

Suppress the label and values for the midpoint axis. The midpoint values 1995 and 1996 do not appear below each bar.

```
axis1 label=none value=none;
```

Modify the response axis. ANGLE=90 prints the axis label vertically.

```
axis2 label=(angle=90 "Metric Tons (millions)")
minor=(n=1)
order=(0 to 500 by 100)
offset=(0,0);
```

Suppress the label and order the values for the group axis. Because the values of COUNTRY are formatted, ORDER= must specify their formatted value.

```
axis3 label=none
order=("China" "United States" "India"
"Indonesia" "Brazil")
split=" ";
```

Define titles and footnote. The footnote uses the catalog entry name to identify the graph.

```
title1 "Corn, Rice, and Wheat Production";
title2 "Leading Producers for 1995 and 1996";
footnote1 j=l "click on bars or legend values" j=r "GRAINALL ";
```

Generate the vertical bar chart that summarizes all grain production for all countries for both years. DISCRETE creates a separate bar for each unique value of YEAR. GROUP= groups the bars by country. To create a legend for midpoint values, SUBGROUP= is assigned the chart variable. GSPACE= controls the space between the groups of bars.

```
proc gchart data=newgrain;
format country $country.;
vbar3d year / discrete
sumvar=megtons
```

```
group=country
subgroup=year
legend=legend1
space=0
width=4
gspace=3
maxis=axis1
raxis=axis2
gaxis=axis3
```

HTML= specifies COUNTRYDRILL as the variable that contains the targets for the bars. HTML_LEGEND= specifies YEARDRILL as the variable that contains the targets for the legend values. Specifying HTML variables causes SAS/GRAPH to add an image map to the HTML body file. NAME= specifies the name of the catalog entry. Because the PATH= destination is a file storage location and not a specific file name, the catalog entry name GRAINALL is automatically assigned to the GIF file. DES= specifies the description that is stored in the graphics catalog and used in the Table of Contents.

```
html=countrydrill
html_legend=yeardrill
name="grainall"
des="Overview of leading grain producers";
run;
quit;
```

Example Part B

VBAR3D options:

- AUTOREF
- HTML=
- HTML_LEGEND=
- SUBGROUP=
- SPACE=
- NAME=

ODS HTML options:

- BODY=

In the second part, the PROC GCHART step continues, using RUN-group processing and WHERE statements to produce two graphs of grain production for each year, one of which is shown in Figure 27.8 on page 620. Each bar represents a country and is subgrouped by grain type. As before, both the bars and the legend values are links to other graphs. The bars link to targets stored in COUNTRYDRILL and the legend values link to targets in TYPEDRILL. These two graphs not only *contain* links, they *are* the link targets for the legend values in Figure 27.7 on page 619. Before each graph is generated, the ODS HTML statement opens a new body file in which to store the output. Because each of these graphs is stored in a separate file, the HREF attributes that are stored in the variable YEARDRILL point only to the file. The name of the file is specified by the BODY= option in the ODS HTML statement. This example shows the HREF attribute that points to the graph of 1995 and is stored in the variable YEARDRILL:

```
HREF=year95_body.html
```

YEARDRILL is assigned to the HTML_LEGEND= option in Part A.

Open a new body file for the graph of 1995 production. Assigning a new body file closes GRAIN_BODY.HTML. The contents and frame files, which remain open, will provide links to all body files.

```
ods html body="year95_body.html" path=odsout;
```

Define the title and footnote for the chart.

```
title1 "Total Production for 1995";
footnote1 j=l "click on bars or legend values" j=r "YEAR95";
```

Subset the data for 1995 and generate the vertical bar chart for 1995. The AUTOREF option draws a reference line on the backplane for every major tick mark value. The SUBGROUP= option creates a separate bar segment for each department. The SPACE= option controls the space between the bars. The HTML= option names the variable that contains the targets for the bars. The HTML_LEGEND= option names the variable that contains the targets for the legend values. The GIF files use the catalog entry name specified by the NAME= option.

```
proc gchart data=newgrain;
  format country $country.;
  where year=1995;
  vbar3d country / sumvar=megtons
  subgroup=type
  autoref
  html=countrydrill
  html_legend=typedrill
  legend=legend1
  space=3
  coutline=black
  maxis=axis3
  raxis=axis2
  name="year95"
  des="Production Breakdown for 1995";
run;
quit;
```

Open a new body file for the graph of 1996 production. Assigning a new body file closes YEAR95_BODY.HTML.

```
ods html body="year96_body.html" path=odsout;
```

Define title and footnote for the second graph.

```
title1 "Total Production for 1996";
footnote1 j=l "click on bars or legend values" j=r "YEAR96 ";
```

Subset the data for 1996 and generate the vertical bar chart for 1996.

```
proc gchart data=newgrain;
format country $country.;
where year=1996;
vbar3d country / sumvar=megtons
subgroup=type
autoref
```

```
html=countrydrill
html_legend=typedrill
legend=legend1
space=3
coutline=black
maxis=axis3
raxis=axis2
name="year96"
des="Production Breakdown for 1996";
run;
quit;
```

Sort the data set for the graphs of production by country. The data must be sorted in order of the BY variable before running PROC GCHART with BY-group processing.

```
proc sort data=newgrain out=country;
by country;
run;
```

Example Part C

VBAR3D options:

- DES=
- GAXIS=
- GROUP=
- HTML=
- NAME=
- OUTSIDE=
- PATTERNID=
- RAXIS=
- SHAPE=

ODS HTML options:

- BODY=
- ANCHOR=

The third part produces the five graphs that show the breakdowns by country. These graphs are generated with BY-group processing and are all stored in one body file. When the file is displayed in the browser, all the graphs appear in one frame that can be scrolled. Because the graphs are stored in one file, the links to them must explicitly point to the location of each graph in the file, not just to the file. This location is defined by an anchor. ODS HTML assigns anchor names by default, but you can specify anchor names with the ANCHOR= option. When the procedure uses BY-group processing to generate multiple pieces of output, ODS automatically increments the anchor name to produce a unique name for each graph. This example assigns the base name {mono country} to the ANCHOR= variable value. The graphs created by this part are referenced by the COUNTRYDRILL variable. With BY-group processing the catalog entry name also increments automatically. The NAME= option specifies COUNTRY as the base name for the graphics output. Because you cannot specify a different description for each graph, the DES= option specifies a generic description for the HTML Table of Contents.

Sort the data set for the graphs of production by country. The data must be sorted in order of the BY variable before running PROC GCHART with BY-group processing.

```
proc sort data=newgrain out=country;
by country;
run;
```

Open a new body file and specify the base anchor name for the graphs of individual countries. Assigning a new body file closes YEAR96_BODY.HTML. Because all the graphs generated by the BY-group processing are stored in one file, each one is automatically assigned an anchor name. The ANCHOR= option specifies a base name for these anchors.

```
ods html body="country_body.html"
anchor="country"
gfootnote
path=odsout
;
```

Redefine AXIS2 to change the range of values and suppress all axis elements. Setting all the label and tick mark options to NONE and assigning a line style of 0 removes the response axis. NOPLANE removes the 3-D axis plane. Specifying ORDER= makes all the graphs use the same range of values.

```
axis2 order=(0 to 250 by 50)
label=none
value=none
style=0
major=none
minor=none
noplane;
```

Suppress the axis label for the midpoint axis.

```
axis4 label=none;
```

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 "Breakdown for #byval(country)";
footnote1 j=l "click on bars";
footnote2 j=c "(Millions of Metric Tons)";
```

Generate the vertical bar chart of production for each country. The PATTERNID= option assigns patterns by group value. The GROUP= option groups the bars by country. The SHAPE= option assigns the bar shape. The OUTSIDE= option displays the SUM statistic above the bars. The HTML= option specifies TYPEDRILL as the variable that contains the targets for the bars. The RAXIS= option assigns the AXIS statement that removes all axis elements. The GAXIS= option assigns the AXIS statement that removes the label. The MAXIS= option assigns the AXIS statement to the midpoint axis. The NAME= option specifies the name of the catalog entry. The graphics catalog entry name increments so the GIF files are named sequentially from COUNTRY to COUNTRY4. The DES= option specifies a general description that appears in the table of contents for all five graphs.

```
proc gchart data=country;
format country $country.;
by country;
vbar3d year / discrete
sumvar=megtons
patternid=group
group=type
shape=hexagon
outside=sum
html=typedrill
width=9
gspace=3
space=0
raxis=axis2
gaxis=axis4
maxis=axis4
name="country"
des="Grain and Year Breakdown";
run;
quit;
```

Sort the data set for the graphs of leading producers of each grain type.

```
proc sort data=grainldr out=type;
by type;
run;
```

Example Part D

VBAR3D options:

INSIDE=

NOZERO

ODS HTML options:

BODY=

NEWFILE=TABLE

Like Part C, this part uses BY-group processing to generate three graphs that show the three leading producers for each type of grain. The program subsets the data and suppresses midpoints with no observations. Instead of storing all of the output in one body file, it stores each graph in a separate file using the ODS HTML option NEWFILE=TABLE. When NEWFILE=TABLE is used with BY-group processing, each new piece of output automatically generates a new body file and simply increments the name of the file that is specified by the BODY= option. Because each graph is stored in

a separate file, the links to these graphs reference only the file name and do not require an anchor name. The graphs created by this part are referenced by the TYPEDRILL variable.

Sort the data set for the graphs of leading producers of each grain type.

```
proc sort data=grainldr out=type;
by type;
run;
```

Open a new body file. Assigning a new body file closes COUNTRY_BODY.HTML. NEWFILE=TABLE opens a new body file for each piece of output generated by the procedure. Each new file increments the name specified by the BODY= option using the number within the body file name as the starting number.

```
ods html body="type1_body.html"
newfile=table
path=odsout;
```

Modify the group axis. Because the SPLIT= option assigns a blank as the split character, the value **United States** prints on two lines.

```
axis5 label=none
    split=" ";
```

Define title and footnote. #BYVAL inserts the value of the BY variable TYPE into the title of each report.

```
title1 "Top Three Producers of #byval(type)";
title2 "(In Millions of Metric Tons)";
footnote j=r "TYPE ";
```

Generate the vertical bar chart of leading producers for each grain type. BY-group processing generates a separate graph for each value TYPE. Each new graph generates a new body file. NOZERO suppresses the midpoints that do not have any observations. The SHAPE= option assigns the bar shape. The INSIDE= option displays the SUM statistic inside the bars.

```
proc gchart data=type (where=(megtons gt 31));
format country $country.;
by type;
vbar3d year / discrete
sumvar=megtons
group=country
nozero
shape=cylinder
noframe
patternid=group
inside=sum
width=8
maxis=axis4
```

```
raxis=axis2
gaxis=axis5
name="type"
des="Leading Producers";
run;
quit;
```

Close the ODS HTML destination, and open the ODS Listing destination. You must close the HTML destination before you can view the output with a browser.

```
ods html close;
ods listing;
```

CHAPTER

28

Troubleshooting Web Output

Troubleshooting Web Output

This chapter contains information that you can use to resolve rendering problems on client workstations.

If you or a member of your audience cannot display your presentation, then refer to the following table for solutions.

NOTE: to ensure that software requirements have been met, see “What does your audience need to view the presentation?” on page 449.

Table 28.1 Web Troubleshooting

Symptom	Cause	Remedy
Can’t 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 can’t display the file.	Browser or Java plug—in may not meet requirements.	Check the requirements. See “What does your audience need to view the presentation?” on page 449.

Symptom	Cause	Remedy
	ActiveX control may not have been installed or may be out of date.	Install the ActiveX control manually (see "Manually Installing the SAS/GRAPH ActiveX Control" on page 455). Consider updating the presentation to prompt users to install the control (see "Configuring an Existing ActiveX Presentation to Prompt Users to Install the SAS/GRAPH ActiveX Control" on page 456).
	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.
	User has not been authenticated for that browser and that Web page.	Check to see if authentication is needed, and then authenticate. See "Connecting to Web Servers that Require Authentication" on page 637.
	Browser doesn't 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 636.
Browser displays blank page.	Browser cannot access the referenced image file.	If *not* running an applet or control, check the image file at the location specified in the HTML file.
	Browser cannot run the applet or control.	For Java, ensure that the HTML file is correctly referencing the Java plug-in and SAS Java archive. See "Specifying the Location of Control and Applet Files (CODEBASE= and ARCHIVE= Options)" on page 486.
		Check browser permissions for running Java scripts. See "Checking Browser Permissions" on page 636.

Symptom	Cause	Remedy
		In the UNIX operating environment, remove any CLASSPATH environment variables. See “Removing CLASSPATH Environment Variables” on page 637.
		Open the browser’s Java Console and trace the source of the error.
Browser displays popup 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 may 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 may 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=activex;`
In ActiveX, the user gets the message `There is a pending reboot for this machine...`	**1** Virus-scanning software may 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** Be sure to close all instances of Internet Explorer before installing the control.
Text font is incorrect.	Java font is defined differently.	Change browser fonts or change the SAS/GRAPH program. See “Correcting Text Fonts” on page 638.
Text in browser shows incorrect characters.	Browser misinterpreting special characters.	Replace special characters with character entities. See “Using HTML Character Entities” on page 636.

Symptom	Cause	Remedy
Graph in browser differs from graph in SAS.	A graphics option or global statement may be unsupported or partially supported for that applet or control. See also "Resolving Differences Between Graphs Generated with Different Technologies" on page 638.	Refer to the descriptions for the options you are using and to Appendix 1, "Summary of ActiveX and Java Support," on page 1601 for information on whether a statement or option is supported.
	A default value in the applet or 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 638.
	GPLOT lines drawn in reverse order on the client.	This change was made intentionally to maintain the integrity of plots drawn with the AREAS= option.
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 cause the graph to revert to its original view.	The graph discards subsetting information if you make changes through the Data Options dialog.	Make any changes needed through the Data Options dialog 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:

- Stylesheets
- Java
- JavaScripts
- Java Console

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

If a special character in your Web presentation does not resolve in the browser, that character may 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.

For presentations that run in the Constellation and Treeview applets, the macros DS2CONST and DS2TREE enable the ENCODE argument, which you can use to automatically replace or not replace angle brackets (“<” and “>”) in TITLE and FOOTNOTE statements.

Connecting to Web Servers that Require Authentication

If you are unable to run a Java applet or install the ActiveX control, then you may 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 allow you to access that Web server.

Removing CLASSPATH Environment Variables

In the UNIX operating environment, if the Java applet does not run after you have verified that your Java archive is correctly specified, then you should remove any CLASSPATH environment variables that have been set. The Java archive files contain all the required classes to run the applets. Your CLASSPATH may point to old versions of the required classes (for example, for use with the webAF software). This can cause the applets to fail to load. Most applications allow you to specify a CLASSPATH at startup, by using a startup option. This is often safer for running multiple clients than using the environment variables.

Setting the SAS_ALT_DISPLAY Variable for X Window Systems on UNIX

You may 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 will be used to locate a graphics 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 will always be 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 on the use of Xvfb.

Correcting Text Fonts

If your presentation displays an incorrect text font on a given client computer, then the cause may 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, Java uses a default font.

When you are using the Java and ActiveX devices or the DS2TREE or DS2CONST macros, 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 JAVA or 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 157 and "TrueType Fonts That Are Supplied by SAS" on page 156 for more information.

For programs that use the JAVAMETA device, specify one of these font names: Helvetica, TimesRoman, Courier, Dialog, DialogInput, or ZapfDingbats; or, specify one of these font styles: serif, sansserif, or monospaced. You can also specify the bold, italic, or italic bold versions of any of these fonts except ZapfDingbats. For example, HelveticaBold, sansserifItalic, or DialogInputItalicBold. If you specify a font style instead of a specific font, the actual font used is determined at run time based on the fonts available on the system where the output is viewed.

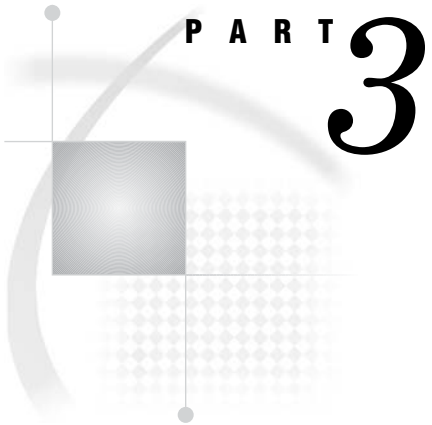
Resolving Differences Between Graphs Generated with Different Technologies

Graphics output that is rendered with one of the Java or ActiveX devices is rendered using Java or ActiveX technology, and 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 may differ from each other even if the output is generated with the same SAS procedure code. The graphs may differ in appearance, in the default values used for certain options, or in the availability of certain features.

For example, differences may occur if you are using a global statement or procedure option that is not enabled for an applet or control. Most global statement and procedure options are fully supported by the Java and ActiveX device drivers. Exceptions are identified in the procedure and statement documentation and summarized in Appendix 1, "Summary of ActiveX and Java Support," on page 1601.

In certain cases, differences between graphs can occur when an applet or 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. For example, consider a bubble plot that is being displayed in the Graph applet. The default bubble size is 5. The Graph applet overrides that default with a larger bubble size. To apply a bubble size of 5, specify BSIZE=5 in the BUBBLE statement, rather than relying on the default value of the BSIZE= option.



PART 3

The Annotate Facility

CHAPTER

29

Using Annotate Data Sets

Overview

The Annotate facility enables you to generate a special data set of graphics commands from which you can produce graphics output. This data set is referred to as an *Annotate data set*. You can use it to generate custom graphics or to enhance graphics

output from many SAS/GRAPH procedures, including GCHART, GCONTOUR, GMAP, GPLOT, GSLIDE, and G3D.

Enhancing Existing Graphs

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

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

Figure 29.1 on page 642 shows GMAP procedure output annotated with stars and labels at selected cities.

Figure 29.1 Annotate Graphics Applied to a Map

The program that creates this output is in “Labeling Cities on a Map” on page 659.

Creating Custom Graphs

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

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

Figure 29.2 on page 643 is an example of a custom graph that uses Annotate commands to draw the graphic elements.

Figure 29.2 Custom Graphics Using Only Annotate Commands

The program that creates this output is in "Drawing a Circle of Stars" on page 664.

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 643. Once you understand the way the data set works, you can follow these three steps to create Annotate graphics:

1. Determine what you want to draw, and where (location) and how (coordinate system) you want to position it on the graphics output. (See "About Annotate Graphics" on page 649.)
2. Build an Annotate data set of graphics commands using the Annotate variables and functions. (See "Creating an Annotate Data Set" on page 654.)
3. Submit a SAS/GRAPH procedure to produce the graphics output. (See "Producing Graphics Output from Annotate Data Sets" on page 655.)

About the Annotate Data Set

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

The observations in an Annotate data set use a set of predefined Annotate variables. The values of the variables in the observation determine what is done and how it is done. To create these observations, you assign values to the variables either explicitly with a DATA step or implicitly with Annotate macros. See "Creating an Annotate Data Set" on page 654.

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 29.1 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 654.)

Output 29.1 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 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, the position of the label in relation to X and Y (centered on the point), and the size of the text.

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

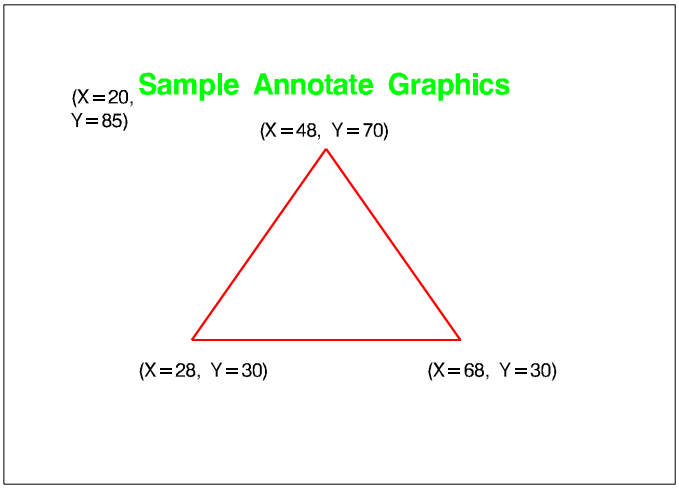
3 Draw the first line of the triangle. The value of the FUNCTION variable (DRAW) tells the program to draw a line from the current point (the one specified by MOVE in the second observation to the new point specified by X and Y. The value of the COLOR variable changes to red.

4 Draw the second line of the triangle.

5 Draw the third line of the triangle.

Figure 29.3 on page 645 shows the green title and the red triangle produced by the TRIANGLE data set and displayed with the GANNOChapter 33, "The GANNO Procedure," on page 913 procedure. Notes on the figure in black contain the X and Y coordinates of the graphics elements.

Figure 29.3 Annotate Output from the TRIANGLE Data Set



Annotate Variables

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

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

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

Table 29.1 on page 645 lists all Annotate variables, grouped by task, and briefly describes each one. See "Annotate Variables" on page 700 for a complete description of each variable.

Table 29.1 Summary of Annotate Variables

Task Group	Variable	Description
Variable that defines an action	FUNCTION	specifies a drawing or programming action; Table 29.2 on page 648 describes these actions.
Positioning variables that determine coordinate values	GROUP	uses the value of the GCHART GROUP= option in place of X or Y
	MIDPOINT	uses the value of the GCHART MIDPOINT= option in place of X or Y

Task Group	Variable	Description
	SUBGROUP	uses the value of the GCHART SUBGROUP= option in place of X or Y
	X	specifies a numeric horizontal coordinate
	Y	specifies a numeric vertical coordinate
	Z	specifies a numeric third dimensional coordinate; used with G3D procedure only
	XC	specifies a horizontal character coordinate; only used with data coordinate systems 1, 2, 7, 8
	YC	specifies a vertical character coordinate; only used with data coordinate systems 1, 2, 7, 8
Positioning variables that contain internal coordinates	XLAST, YLAST	contain the X and Y coordinates of the last nontext function
	XLSTT, YLSTT	contain the X and Y coordinates of the last text function
Positioning variables that specify coordinate systems	HSYS	specifies type of units for the SIZE variable
	XSYS	specifies coordinate system for X or XC coordinates
	YSYS	specifies coordinate system for Y or YC coordinates
	ZSYS	specifies coordinate system for Z coordinate (G3D procedure only)
Attribute variables	ANGLE	angle of text label or starting angle of a pie slice
	CBORDER	colored border around text or symbol
	CBOX	colored box behind text or symbol
	COLOR	color of a graphics primitive
	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

See Figure 29.4 on page 647 for a table that shows you which Annotate functions are used with which Annotate variables.

Figure 29.4 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 649.

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 652, "Using the LIFO Stack" on page 657, and "Debugging" on page 658.

Table 29.2 on page 648 summarizes the tasks that are performed by the Annotate functions. See "Annotate Functions" on page 669 for a complete description of the FUNCTION variable and its values.

Table 29.2 Summary of Graphics Tasks Performed by Annotate Functions

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

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

See Figure 29.4 on page 647 for a table that shows you which Annotate functions work with which Annotate variables.

About Annotate Graphics

When you create Annotate graphics, you specify these things:

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

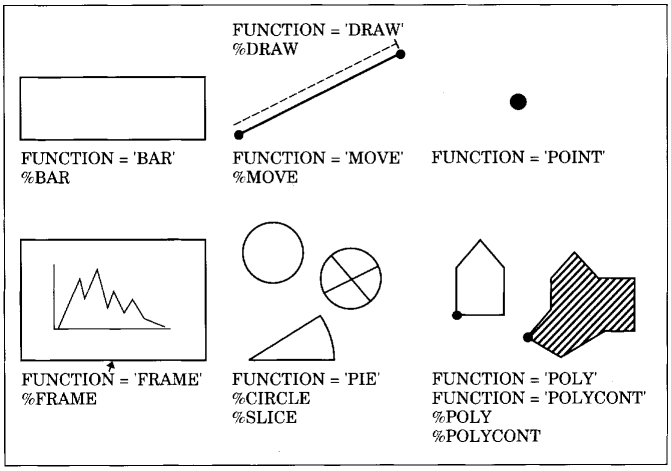
The following sections describe the components of the graphics output that are produced by an Annotate data set.

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 29.5 on page 649 along with the value of the FUNCTION variable or Annotate macro that draws them.

Figure 29.5 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

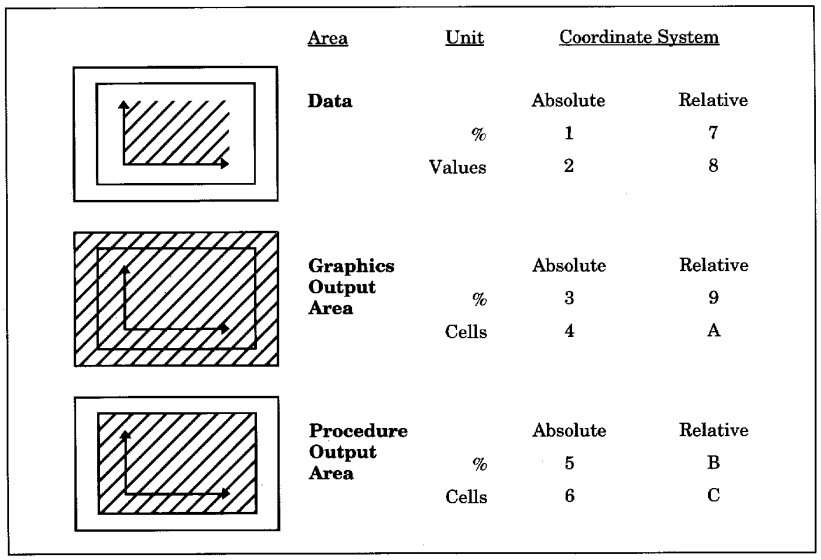
A coordinate system determines how coordinates are interpreted. You specify a coordinate system to use for each dimension, using the XSYS, YSYS, and ZSYS variables (for X, Y, and Z, respectively). Use ZSYS to annotate graphics output only from the G3D procedure.

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

These are the important components of the Annotate coordinate systems:

- *Area:* Each coordinate system refers to one of three drawing areas: data area, procedure output area, and graphics output area. Coordinates are measured from a different origin for each area; they also have different limits. Figure 29.6 on page 651 shows the areas on the graphics output and the coordinate systems that use them.

Figure 29.6 Areas and Their Coordinate Systems



- *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 652.
  - 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 29.3 on page 651 describes the coordinate system values for the XSYS, YSYS, ZSYS, and HSYS variables.

Table 29.3 Coordinate System Values for XSYS, YSYS, ZSYS, and HSYS Variables

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

Type of Coordinates	Area	Units	Range	Value for XSYS, YSYS, ZSYS, HSYS
	data	values	minimum to maximum of axis	'8' *
	graphics output area	%	0-100% of graphics output area	'9'
	graphics output area	cells	0 to limit of graphics output area	'A'
	procedure output area	%	0–100% of procedure output area	'B'
	procedure output area	cells	0 to limit of procedure output area	'C'
N/A	Text font point size	N/A	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. Additionally, 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 that are set by the HPOS= and VPOS= graphics options or by the PCOLS and LCOLS device parameters.

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

See "Overview" on page 59 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 671.) These internal variables can also provide default coordinates if X, XC, Y, or YC contains a missing value.

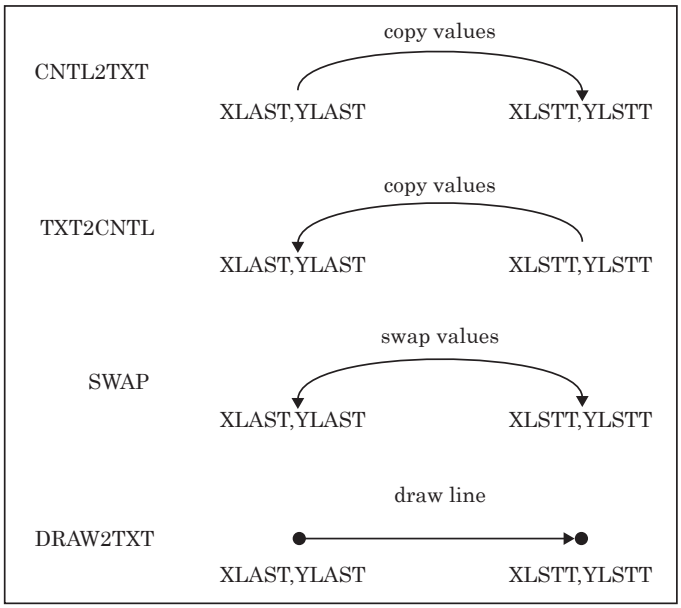
The internal coordinates are automatically updated by some of the Annotate functions. The text functions, LABEL and SYMBOL, update the (XLSTT,YLSTT) variables. The BAR, DRAW, MOVE, PIE, and POINT functions update the (XLAST,YLAST) variables.

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 29.7 on page 653.

Figure 29.7 Programming Functions That Manipulate System Variables



For a complete description, see "Annotate Internal Coordinates" on page 737.

Attribute Variables

Attribute variables control the appearance of the graphics elements. Each function uses only a subset of these variables. See Table 29.1 on page 645 for a list of attribute variables.

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

For a complete description of the attribute variables and the aspect of the graphics elements that they control, see "Annotate Variables" on page 700.

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 on creating SAS data sets, see *SAS Language Reference: Concepts*.

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 29.1:

```
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;
```

INF

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 takes as input the annotate data set “triangle” created by the previous DATA step and creates the output shown in Figure 29.3 on page 645.

Using Annotate Macros in the DATA Step

A set of Annotate macros is provided in the SAS sample library. You can use macro calls in a DATA step to create observations in an Annotate data set. You can also use Annotate macros and explicit variable assignments together in the same DATA step. For complete information, see "Annotate Macros" on page 738 and "Using Annotate Macros" on page 759.

Effect of Missing Values

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

Variables that have a missing value use a default value. For example, if the COLOR variable has a missing value, then the first color in either the color list that is defined by the COLORS= graphics option, if specified, or 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 "Annotate Variables" on page 700 for the default value of each Annotate variable.

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

Sometimes missing values are required to produce the desired results. If you have calculated the coordinates of a point and have the values stored in (XLAST,YLAST) or (XLSTT,YLSTT), you can force Annotate to use the internal coordinates by supplying missing values for the X and Y variables. See "Annotate Internal Coordinates" on page 737 for details on using the (XLAST,YLAST) and (XLSTT,YLSTT) internal coordinates.

Producing Graphics Output from Annotate Data Sets

You can display Annotate graphics in two ways:

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

Including Annotate Graphics with Procedure Output

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

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

Producing Only Annotate Graphics Output

To produce Annotate graphics 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, "The GANNO Procedure," on page 913 for information on displaying or storing Annotate graphics.
- The GSLIDE procedure can also produce graphics output consisting only of Annotate graphics. In addition, you can enhance the graphics output with TITLE, NOTE, and FOOTNOTE statements. See Chapter 51, "The GSLIDE Procedure," on page 1517 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 on the annotate functions and variables, see the Chapter 30, "Annotate Dictionary," on page 667. For information on using annotate data sets in Web output, see Chapter 23, "Generating Web Output with the Annotate Facility," on page 539.

Annotate Processing Details

Order in Which Graphics Elements Are Drawn

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

CAUTION:

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

Controlling the Processing with the WHEN Variable

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

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

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

Within each set, graphics elements are drawn in the order that they appear in the Annotate data set and overlay each other as appropriate (on devices that demonstrate overlay behavior). For details, see the description of the WHEN variable on "WHEN Variable" on page 725.

Using BY-Group Processing with the Annotate Facility

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

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

See "BY Statement" on page 216 for details.

Using the LIFO Stack

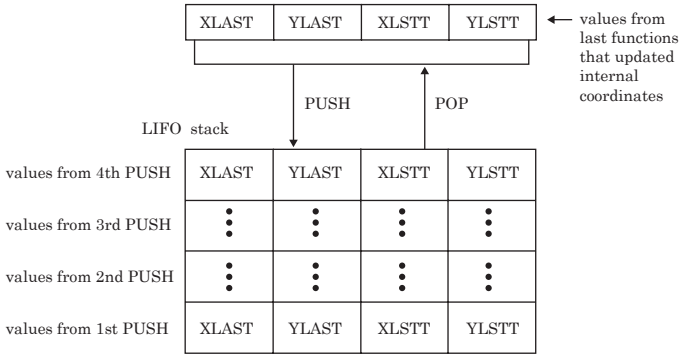
The FUNCTION variable supports several programming functions that manipulate the internal coordinates and provide other utility operations. Several of these functions use the LIFO stack to track and set variable values.

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 29.8 on page 658 illustrates how PUSH and POP functions work together.

Figure 29.8 Using PUSH and POP to Store and Retrieve Coordinate Values



See also “Internal Coordinates” on page 652.

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 29.1 provides complete information about the value that you specify for each variable in every observation.

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

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

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

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

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

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

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

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

  ```
  ERROR LIMIT REACHED IN ANNOTATE PROCESS

                  20 TOTAL ERRORS
  ```

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

Examples

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

- “Labeling Cities on a Map” on page 659
- “Labeling Subgroups in a Vertical Bar Chart” on page 661
- “Drawing a Circle of Stars” on page 664

Other examples that use Annotate data sets are as follows:

- Example 1 on page 916 (and others in that chapter)
- Example 2 on page 1208
- Example 2 on page 1524
- Example 4 on page 1416

Labeling Cities on a Map

Features:

Annotate function:	LABEL
	SYMBOL
Annotate variables:	HSYS
	POSITION
	SIZE
	TEXT
	WHEN
	X and Y
	XSYS
	YSYS
Sample library member:	GANCITY

Figure 29.9 Map with Labeled Cities

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

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

See Example 4 on page 1416 for an example of labeling a map using map coordinates in units of latitude and longitude.

See Chapter 43, "The GMAP Procedure," on page 1239 for more information on using map data sets.

Set the graphics environment.

```
goptions reset=all border;
```

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

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

Create the Annotate data set, CITYSTAR. CITYSTAR contains the commands that draw a star and a label at each of the three cities. Setting WHEN to A draws the annotation after the map.

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

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

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

Create the observation that draws the star. The text string V is the character code for the star figure in the MARKER font assigned by the STYLE variable.

```
   function="symbol"; style="marker"; text="V"; color="red"; size=5;
      output;
```

Create the observation that labels the city. TEXT is assigned the value of CITY. The default font is used. SIZE uses the units assigned by HSYS so text height is 5 percent of the height of the graphics output area. POSITION 8 places the label directly below the city location.

```
   function="label"; style=""; text=city; color="green";
      size=5; position="8"; output;
run;
```

Define the title for the map.

```
title "Distribution Center Locations";
```

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

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

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

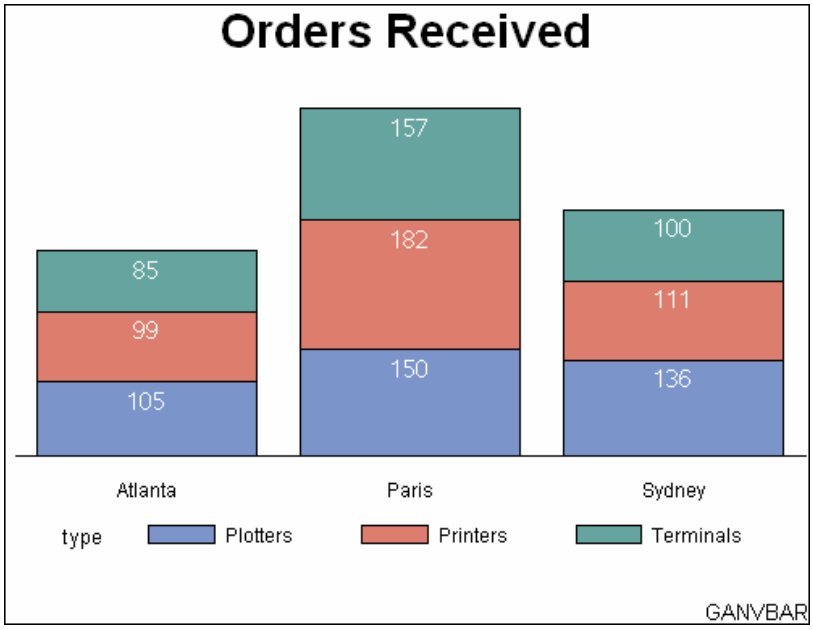
```
proc gmap data=lower48 map=lower48;
   id state;
   choro state / annotate=citystar discrete nolegend;
run;
quit;
```

Labeling Subgroups in a Vertical Bar Chart

Features:

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

Figure 29.10 Bar Chart with Labeled Subgroups



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

See Chapter 36, "The GCHART Procedure," on page 989 for more information on creating bar charts.

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;
```

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

```
data barlabel;
   length color style $ 8;
   retain color "white" when "a" style "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.

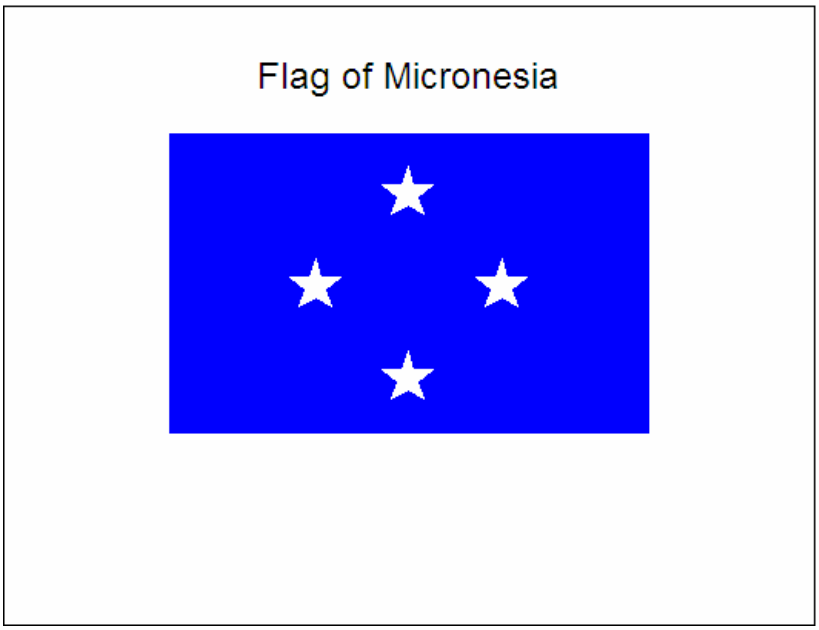
```
proc gchart data=sold;
   vbar city / type=sum
               sumvar=units
               subgroup=type
               width=17
               raxis=axis1
               maxis=axis2
               annotate=barlabel;
run;
quit;
```

Drawing a Circle of Stars

Features:

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

Figure 29.11 Stars Positioned in a Circle with GANNO



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

Set the graphics environment.

```
goptions reset=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;
```

CHAPTER

30

Annotate Dictionary

Annotate Dictionary Overview

The Annotate facility enables you to generate a special data set of graphics commands from which you can produce graphics output. This data set is referred to as an Annotate data set. You can generate a complete graph using an Annotate data set in conjunction with Chapter 33, "The GANNO Procedure," on page 913 or Chapter 51, "The GSLIDE Procedure," on page 1517, or you can apply an Annotate data set to graphics that were generated with procedures such as Chapter 36, "The GCHART Procedure," on page 989, Chapter 37, "The GCONTOUR Procedure," on page 1095, and Chapter 43, "The GMAP Procedure," on page 1239, among others.

In addition, SAS/GRAPH supports the following procedures with the Java or ActiveX devices: GCHART, GCONTOUR, GMAP, GPLOT, GRADAR, and G3D.

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 700. "Annotate Functions" on page 669 determine what is to be done with each observation. "Annotate Macros" on page 738 simplify the process of drawing a graphics element. "Annotate Error Messages" on page 761 are sent to the SAS log.

For usage information and example programs , refer to "Using Annotate Macros" on page 759 and Chapter 29, "Using Annotate Data Sets," on page 641.

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 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 "Annotate Variables" on page 700.

For a summary of drawing and programming tasks performed by the FUNCTION variable, see Table 29.2 on page 648.

The variables that are available for use with each function are listed in Figure 29.4 on page 647.

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

COLOR='*color*'
specifies the color of the arrow that is being drawn. *Color* can be any SAS/GRAPH color name.

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.

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable. See "HSYS Variable" on page 707 for an explanation of *coordinate-system*.

LINE=*length*
specifies the length of the sides of the arrowhead. The units for LINE are always a percentage of the graphics area, regardless of the value for HSYS.

Default: 1

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.

As the thickness of the line increases, it may 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. See Figure 30.7 on page 677 for examples of line thicknesses.

Figure 30.1 Sample Line Thicknesses Used with the SIZE Variable

STYLE= 'CLOSED' | 'FILLED' | 'OPEN'
specifies the type of arrowhead. Specify one of the following values:

CLOSED
the arrowhead is shaped like an empty triangle.

FILLED
the arrowhead is shaped like a filled triangle.

OPEN
the arrowhead is shaped like a V.

Default: OPEN

WHEN='B' | 'A'
specifies when to draw the line in relation to other procedure output. See "WHEN Variable" on page 725.

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the endpoint of a line drawn from (XLAST, YLAST) to (X,Y).

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable (PROC G3D only). See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

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*'

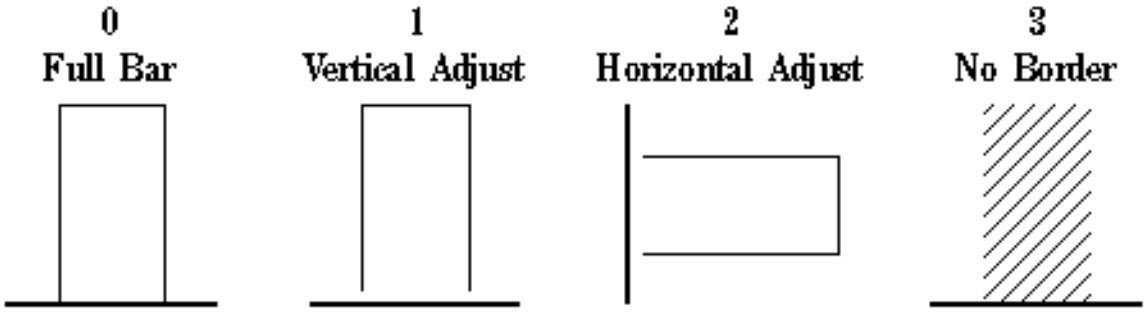
specifies the color of either the interior of the bar or the outline of the bar. *Color* can be any SAS/GRAPH color name. The part of the bar affected depends on the value of the STYLE variable. If STYLE specifies a pattern or fill, the COLOR variable determines the color of the interior. If STYLE specifies 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.

HTML='*link-string*'
specifies the text that defines the link for drill-down.

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. The following figure illustrates LINE values.

Figure 30.2 LINE Values for Bars



Default: 1

SIZE=*thickness*
specifies a line thickness for the rectangle

STYLE='*fill-pattern*'
specifies the pattern that fills the bar. *Fill-pattern* can be the following bar and block patterns:

SOLID S	a solid fill.
EMPTY E	an empty fill.
style<*density*>	a shaded pattern: *style* can be R | X | L *density* can be 1...5

WHEN='B' | 'A'
specifies when to draw the bar in relation to other procedure output. See "WHEN Variable" on page 725.

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). Use the Z variable only when you are annotating output from the G3D procedure. Figure 30.3 on page 673 illustrates the use of these coordinates.

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 729 for an explanation of *coordinate-system*.

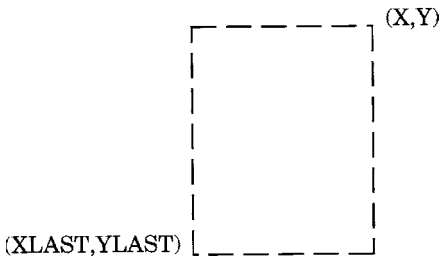
YSYS='*coordinate-system*'
specifies the coordinate system for Y or YC variable. The YC variable can only be used with YSYS='2'. See "YSYS Variable" on page 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

Details

Figure 30.3 on page 673 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 30.3 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 30.6 on page 675.

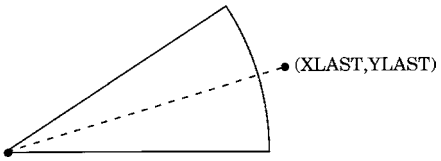
First, use the PIE function to draw the pie slice:

```
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 30.4 on page 674.

```
/* 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;
```

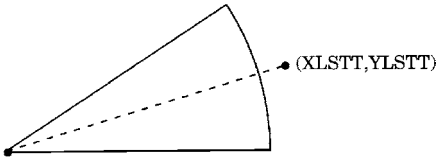
Figure 30.4 Position Calculated with the PIEXY Function



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 30.5 on page 674 shows the results.

```
function="cntl2txt"; output;
```

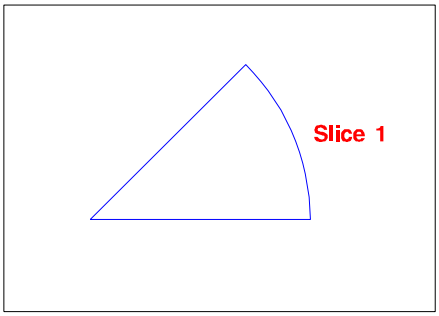
Figure 30.5 Coordinates after Using the CNTL2TXT Function



Now you can use the LABEL function to write the label as shown in Figure 30.6 on page 675. 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 30.6 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.

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.

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.

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable. See "HSYS Variable" on page 707 for an explanation of *coordinate-system*.

LINE=1...46
specifies the line type of the line that is being drawn. See "Specifying Line Types" on page 276 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.

As the thickness of the line increases, it may 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. See Figure 30.7 on page 677 for examples of line thicknesses.

Figure 30.7 Sample Line Thicknesses Used with the SIZE Variable

WHEN='B' | 'A'
specifies when to draw the line in relation to other procedure output. See "WHEN Variable" on page 725.

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the endpoint of a line drawn from (XLAST, YLAST) to (X,Y).

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable (PROC G3D only). See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

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.

HSYS='*coordinate-system*'

specifies the coordinate system for the SIZE variable. See “HSYS Variable” on page 707 for an explanation of *coordinate-system*.

LINE=1...46
specifies the line type of the line that is being drawn. See “Specifying Line Types” on page 276 for an illustration of the line types.

SIZE=*line-thickness*
specifies the thickness of the line that is being drawn. See “DRAW Function” on page 676 for details.

WHEN=’B’ | ’A’
specifies when to draw the line in relation to generation of the procedure output. See “WHEN Variable” on page 725.

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. Optionally specifies a background color for the framed area.

Syntax

FUNCTION=’FRAME’;

Note: The FRAME function is not supported by Java. △

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.

HSYS=’*coordinate-system*’
specifies the coordinate system for the SIZE variable. See “HSYS Variable” on page 707 for an explanation of *coordinate-system*.

Note: The HSYS variable is not supported by ActiveX. △

HTML=’*link-string*’
specifies the text that defines the link for drill-down.

LINE=1...46
specifies the line type with which to draw the frame. See “Specifying Line Types” on page 276 for an illustration of the line types.

SIZE=*line-thickness*
specifies the thickness of the line with which to draw the frame. See “DRAW Function” on page 676 for details.

Note: The SIZE variable is not supported by ActiveX. △

STYLE='*fill-pattern*'
specifies the pattern that fills the area that is bounded by the frame. *Fill-pattern* can be the following bar and block patterns:

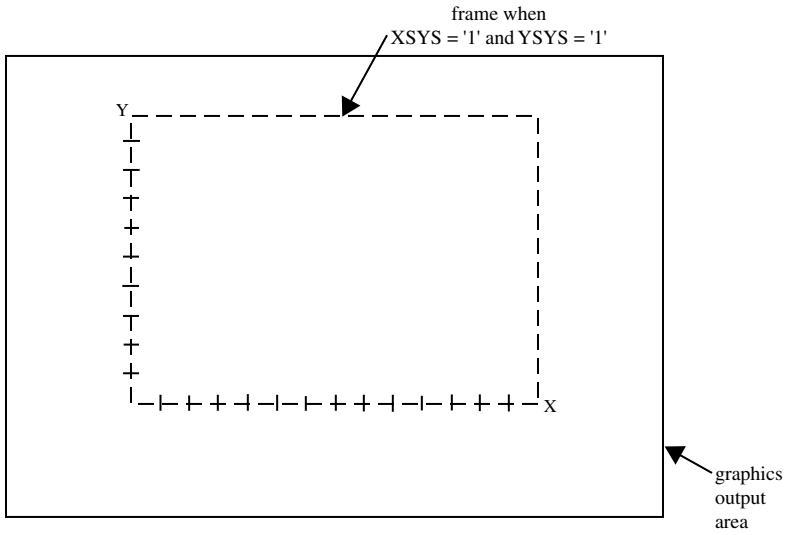
SOLID S	a solid fill.
EMPTY E	an empty fill.
style<density>	a shaded pattern: *style* can be R | X | L *density* can be 1...5

See also the discussion of fill patterns for bars and blocks in VALUE= on page 242.

WHEN='B' | 'A'
specifies when to draw the frame in relation to other procedure output. See "WHEN Variable" on page 725

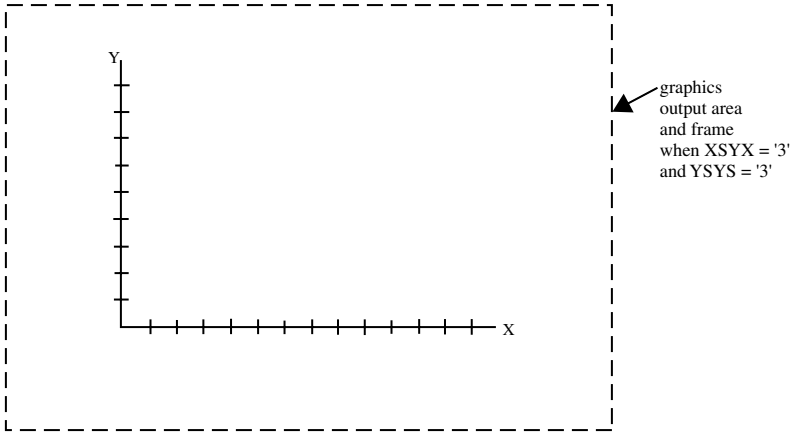
XSYS='*coordinate-system*'
YSYS='*coordinate-system*'
define the area to be enclosed by the frame. For example, if XSYS='1' and YSYS='1', the frame encloses the axis area as shown in Figure 30.8 on page 680. See "XSYS Variable" on page 729 and the YSYS variable on "YSYS Variable" on page 734 for an explanation of *coordinate-system*.

Figure 30.8 Frame Created When XSYS='1' and YSYS='1'



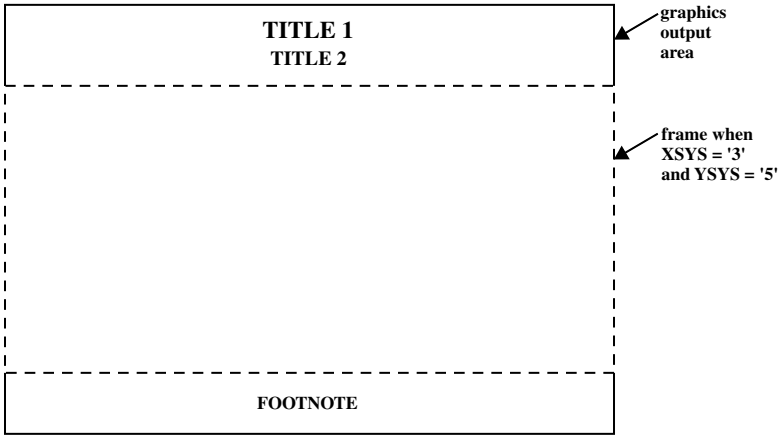
If XSYS='3' and YSYS='3', the frame encloses the entire graphics output area, as shown in Figure 30.9 on page 681.

Figure 30.9 Frame Created When XSYS='3' and YSYS='3'



The values for XSYS and YSYS do not have to be the same. If XSYS='3' and YSYS='5', the frame encloses the entire width of the graphics output area; however, vertically, the frame only encloses the procedure output area as shown in Figure 30.10 on page 681.

Figure 30.10 Frame Created When XSYS='3' and YSYS='5'



See "XSYS Variable" on page 729 and "YSYS Variable" on page 734 for an explanation of these variables and the areas that they affect.

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" on page 335 for more information on 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 link for drill-down.

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. △

STYLE = 'TILE' | 'FIT';
specifies how the image is to be applied to fill the specified area of the graphics output. The default value of TILE replicates the image to fill the area. The FIT value stretches a single instance of the image to fill the area.

X=*horizontal-coordinate*;
specifies the horizontal coordinate that determines the size of the image displayed in the graphics output.

Y=*vertical-coordinate*;
specifies the vertical coordinate that determines the size of the image displayed in the graphics output.

Z=*depth-coordinate*;
specifies the depth coordinate for 3D output.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

Details

The following example shows how the IMAGE function adds a single stretched instance of an image to the graphics output, beginning at the current coordinates and ending 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 use, see "Image File Types Supported by SAS/GRAPH" on page 181.

LABEL Function

Places text in the graphics output. Associated variables can control the color, size, font, base angle, and rotation of the characters displayed.

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.

CBORDER='*color*' | 'CTEXT'
draws a colored border around the text. *Color* can be any SAS/GRAPH color name.

CBOX='*color*' | 'CBACK
draws a solid, colored box behind the text. *Color* can be any SAS/GRAPH color name.

COLOR='*color*'
specifies the color of the text. *Color* can be any SAS/GRAPH color name.

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.

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable. See "HSYS Variable" on page 707 for an explanation of *coordinate-system*.

HTML='*link-string*'
specifies the text that defines the link for drill-down.

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. For details, see "POSITION Variable" on page 714.

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.

SIZE=*height*

specifies the height of the text string. The SIZE variable units are based on the value of the HSYS variable.

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 719 for a description of the various font specifications.

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.

WHEN='B' | 'A'
specifies when to draw the text strings in relation to other procedure output. See "WHEN Variable" on page 725

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the start point of the text string. The Z variable can be used only with the G3D procedure. Optionally, you can modify the placement of the text string with the POSITION variable.

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

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.

WHEN='B' | 'A'
specifies when to perform the move in relation to other procedure output. See also "WHEN Variable" on page 725.

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
XC='*character-type-horizontal-coordinate*'

YC='*character-type-vertical-coordinate*'
specify the coordinates to which the pen is to be moved. The Z variable can only be used with the G3D procedure.

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

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, you must specify a value for the ANGLE variable.

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.

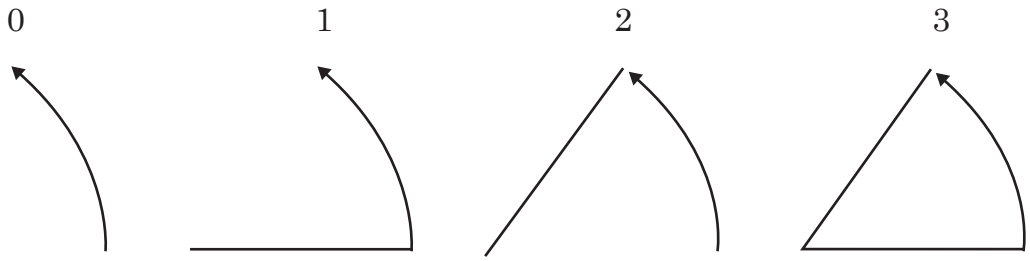
GROUP=*group-value*
MIDPOINT=*midpoint-value*
SUBGROUP=*subgroup-value*
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable. See "HSYS Variable" on page 707 for an explanation of *coordinate-system*.

HTML='*link-string*'
specifies the text that defines the link for drill-down.

LINE=0...3
specifies which slice line (or lines) to draw. See Figure 30.11 on page 687 for line values and their actions. LINE=0 draws only the outside of the arc and enables you to draw a circle.

Figure 30.11 LINE Values Used with the PIE Function



ROTATE=*rotation-angle*
specifies the angle of rotation or the delta angle of the slice arc. The default is 0.00.

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) and 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.

SIZE=*radius*
specifies the radius of the circle being drawn. The SIZE variable uses units that are determined by the HSYS variable.

STYLE='*fill-pattern*'
specifies the value of the pattern that fills the pie slices. *Fill-pattern* can be the following pie patterns:

PSOLID PS	a solid fill.
PEMPTY PE	an empty fill.
P*density*<*style*<*angle*>>	a shaded pattern: *density* can be 1...5 *style* can be X | N *angle* can be 0...360

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. See also the discussion of fill patterns for pie and star charts in VALUE= on page 245.

WIDTH='*line-thickness*'
specifies the thickness of the outline around the pie slice. See "WIDTH Variable" on page 726.

WHEN='B' | 'A'
specifies when to draw the pie slice in relation to other procedure output. See "WHEN Variable" on page 725.

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
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 PROC G3D 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.

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

See Also

"CNTL2TXT Function" on page 673

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.

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable. See "HSYS Variable" on page 707 for an explanation of *coordinate-system*.

SIZE=*radius*
specifies the new radius of the pie slice. The new radius is used by a subsequent PIEXY function. The HSYS variable determines the SIZE variable units.

WHEN='B' | 'A'
specifies when to draw the pie slice in relation to other procedure output. See "WHEN Variable" on page 725

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
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. Use the Z variable only with the G3D procedure.

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

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). The default is 0.00.

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 (where 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;
```

WHEN='B' | 'A'
specifies when to update the internal coordinate pair (XLAST, YLAST) in relation to other procedure output. See "WHEN Variable" on page 725.

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 30.12 on page 691 shows a pie slice that is drawn with the PIE function. Figure 30.13 on page 691 shows a point beyond the arc that was calculated using the PIEXY function.

Figure 30.12 Pie Slice Drawn with the PIE Function

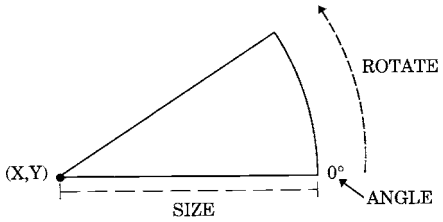
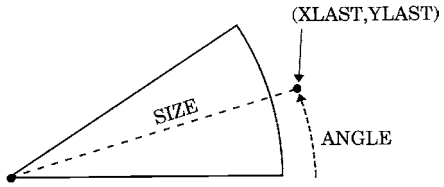


Figure 30.13 Point Calculated with the PIEXY Function



See Also

"CNTL2TXT Function" on page 673

POINT Function

Places a single point at the (X, Y) coordinates in the color 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.

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.

WHEN='B' | 'A'
specifies when to draw the point in relation to other procedure output. See "WHEN Variable" on page 725

X=*horizontal-coordinate*

Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the coordinates of the point that is to be drawn. Use the Z variable only with the G3D procedure.

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

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.

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 data coordinate systems 1, 2, 7, and 8.

HTML='*link-string*'
specifies the text that defines the link for drill-down.

LINE=1...46
specifies the line type that outlines the polygon. See "Specifying Line Types" on page 276 for an illustration of the line types.

SIZE=*thickness*
specifies a line thickness for the polygon

STYLE='*fill-pattern*'
specifies the value of the pattern that fills the polygon. *Fill-pattern* can be the following map patterns:

MSOLID a solid pattern
MS

MEMPTY an empty pattern
ME

M*density*<*style*<*angle*>> a shaded pattern:

density can be 1...5

style can be X | N

angle can be 0...360.

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. See also the discussion of fill patterns for maps in VALUE= on page 244.

WHEN='B' | 'A'
specifies when to begin the polygon in relation to other procedure output. See "WHEN Variable" on page 725

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the initial point of the polygon that is being created. Use the Z variable only with the G3D procedure.

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

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 694

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. 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.

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.

WHEN='B' | 'A'
specifies when to draw the polygon in relation to other procedure output. See "WHEN Variable" on page 725

X=*horizontal-coordinate*
Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify a point on the outline of the polygon that is being created. Use the Z variable only with the G3D procedure.

XSYS='*coordinate-system*'
specifies the coordinate system for the X and XC variable. Use the XC variable only with XSYS='2'. See "XSYS Variable" on page 729 for an explanation of *coordinate-system*.

YSYS='*coordinate-system*'
specifies the coordinate system for the Y and YC variable. Use the YC variable only with YSYS='2'. See "YSYS Variable" on page 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

Details

The polygon definition is terminated by a new POLY command or by any of these functions:

BAR

DRAW

DRAW2TXT

FRAME

LABEL

MOVE

PIE

PIECNTR

PIEXY

POINT

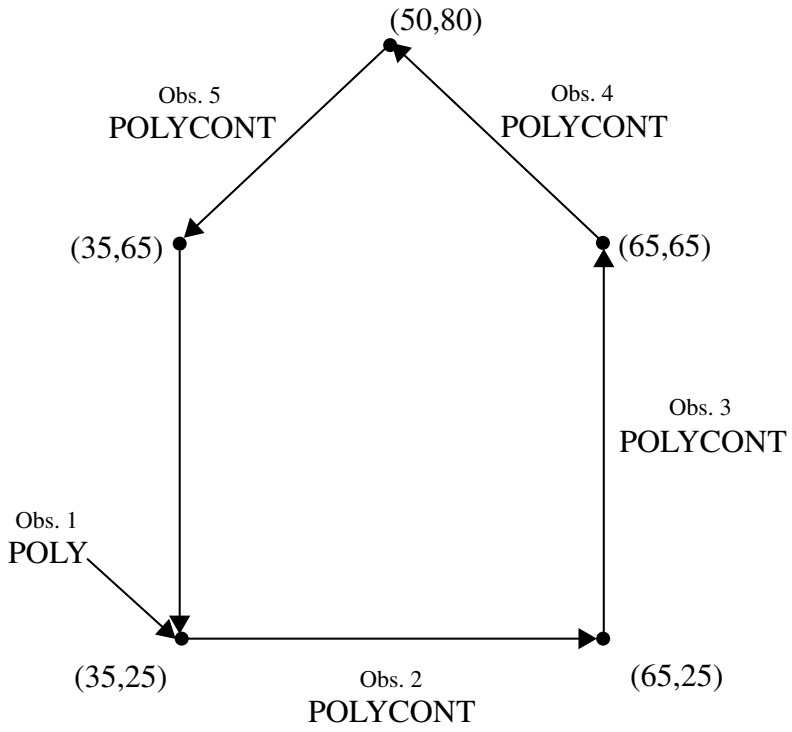
SYMBOL

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 30.14 on page 696:

```
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 30.14 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 the way that 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” on page 1250 for more information on 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';

Details

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';

Details

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';

Details

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 colored border around the text. *Color* can be any SAS/GRAPH color name.

CBOX='*color*' | 'CBACK'
draws a solid, colored box behind the text. *Color* can be any SAS/GRAPH color name.

COLOR='*color*'
specifies the symbol color. *Color* can be any SAS/GRAPH color name. The COLOR variable behaves in the same way as the COLOR= option in the SYMBOL statement. See COLOR= on page 254 for details

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.

HSYS='*coordinate-system*'
specifies the coordinate system for the SIZE variable. See "HSYS Variable" on page 707 for an explanation of *coordinate-system*.

HTML='*link-string*'
specifies the text that defines the link for drill-down.

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 HEIGHT= on page 256 for details.

STYLE='*font-specification*' | 'NONE';
specifies the font that is used to draw the symbol that is specified by the TEXT variable. See "STYLE Variable (Fonts)" on page 719 for a description of the various font specifications.

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 FONT= on page 256 for details.

TEXT='*special-symbol*' | '*text-string*';
specifies the symbol 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 and are illustrated in VALUE= on page 269.

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 VALUE= on page 269 for details.

WHEN='B' | 'A'
specifies when to draw the symbols in relation to other procedure output. See "WHEN Variable" on page 725

Y=*vertical-coordinate*
Z=*depth-coordinate* (PROC G3D only)
XC='*character-type-horizontal-coordinate*'
YC='*character-type-vertical-coordinate*'
specify the point at which the symbol is placed. Use the Z variable only with the G3D procedure.

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 729 for an explanation of *coordinate-system*.

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 734 for an explanation of *coordinate-system*.

ZSYS='*coordinate-system*'
specifies the coordinate system for the Z variable. See "ZSYS Variable" on page 736 for an explanation of *coordinate-system*.

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 Figure 14.21 on page 271.
- 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';

Details

TXT2CNTL allows nontext functions to use the ending position of a text string as a starting or ending point.

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 "Annotate Functions" on page 669 for more information about how each variable is used with each function. For a summary of Annotate variables and their uses, see Table 29.1 on page 645.

ANGLE Variable

Specifies the angle at which the graphics output is drawn.

Type: numeric

Default: function dependent

Syntax

ANGLE= *angle-value*;

Functions

The ANGLE variable is function dependent.

If function is...	then 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 postion), 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, or 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 also: CBOX

Syntax

CBORDER='*color*' | 'CTEXT';

color
specifies the color that fills the box. The *color* value can be any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 for more information about specifying colors.

Specifying a null value for the *color* value (CBOX=' ')cancels the CBOX 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.

Functions

You can use the CBORDER variable with these functions:

LABEL

SYMBOL

Details

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.

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

See also: CBORDER

Syntax

CBOX='*color*' | 'CBACK';

color
specifies the color that fills the box. *Color* is any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 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.

Functions

You can use the CBOX variable with these functions:

LABEL

SYMBOL

Details

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.

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. first color in color list of the COLORS= graphics option
2. first color in device's default color list.

Syntax

COLOR='*color*';

color
specifies any SAS/GRAPH color name. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 for more information about specifying colors.

Functions

The COLOR variable is function dependent.

If function is...	then the COLOR variable specifies...
BAR	the color that outlines and, optionally, fills the bar if a pattern is specified in the STYLE (patterns)"STYLE Variable (Patterns)" on page 721 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)"STYLE Variable (Patterns)" on page 721 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.

If function is...	then 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*';

function-name
specifies the name of an Annotate function. The *function-name* value can be any of the following.

BAR	draws and, optionally, fills a rectangle.
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. See Figure 14.21 on page 271 for a list of the symbols.
TXT2CNTL	copies the values (XLSTT, YLSTT) to (XLAST, YLAST), overwriting the previous values of (XLAST, YLAST).

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 "Annotate Functions" on page 669.

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*;

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.

Functions

You can use the GROUP variable only with the data coordinate systems 1, 2, 7, and 8, and with these functions:

BAR

DRAW

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

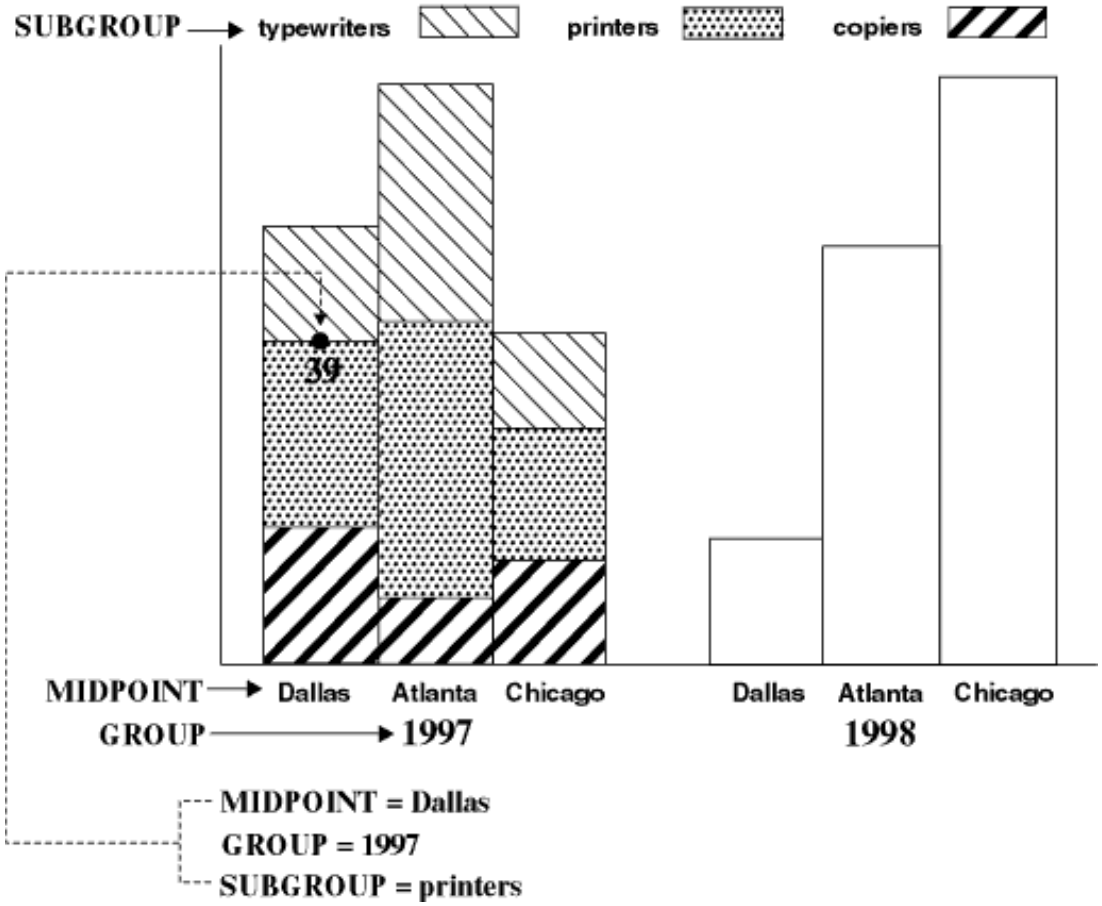
SYMBOL

Details

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 30.15 on page 707 shows how the GROUP variable works with the SUBGROUP and MIDPOINT variables to label the bars of a vertical bar chart.

Figure 30.15 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).

HSYS Variable

Defines the coordinate system and area of the output used by the SIZE variable to display the Annotate graphics. Additionally, you can use the HSYS variable with Java or ActiveX to control the markersize and linesize for the BAR, DRAW, DRAW2TXT, POLY, and SYMBOL functions.

Type: character

Length: 1

Default: 4

Syntax

HSYS='*coordinate-system*';

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 650 for a description of the areas and coordinate systems.

Functions

You can use HSYS with these functions, all of which also use the SIZE variable:

DRAW

DRAW2TXT

FRAME

LABEL

PIE

PIECNTR

SYMBOL

Details

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 percent of the graphics output area. If you use HSYS=’1’ and SIZE=10 with DRAW, the thickness of the line is 10 percent 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.

HTML Variable

Defines a link in the HTML file created for a drill-down graph. This link is associated with an area of the graph and contains valid HTML syntax that can point to a report or another graph that you want to display when the user drills down on the area.

Type: character
Length: no limit
Default: none

Syntax

HTML='*link-string*';

link-string
specifies the text that defines the link for drill-down. For more information about drill-down graphs and how to specify the link string, see "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601. For using the HTML variable for data tips, see "Adding Custom Data Tips with the HTML= Option" on page 598.

Functions

You can use the HTML variable with these functions:

BAR
FRAME
IMAGE
LABEL
PIE
POLY
SYMBOL

Details

Use a LENGTH statement to set the length of the HTML variable to the longest string 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 will use 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;
```

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*';

external-file
specifies the full path or full file name of an external image file. The format of the external file specification varies between 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. △

Details

The IMGPATH variable can be used only with the "IMAGE Function" on page 682.

The manner in which the specified image is to be displayed is determined by the "STYLE Variable (Images)" on page 720.

For a list of the file types that you use, see "Image File Types Supported by SAS/GRAPH" on page 181.

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 for all functions: 1

Syntax

LINE=*line-type*;

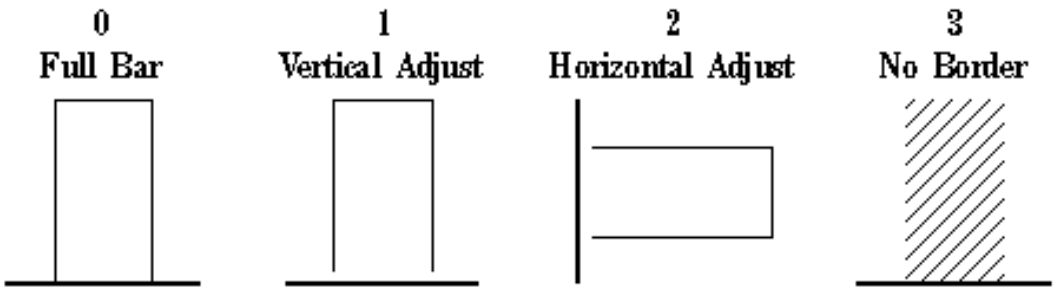
Functions

The behavior and syntax of the LINE variable is function-dependent.

ARROW
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.

BAR
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.

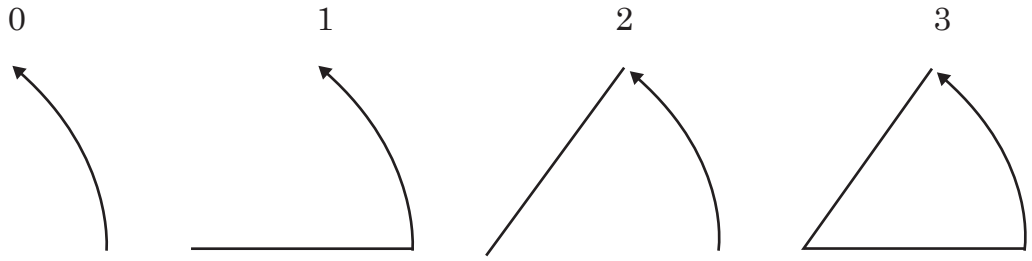
Figure 30.16 LINE Values for Bars



DRAW, DRAW2TXT, FRAME, POLY
Valid values are whole numbers from 0 to 46. A value of 0 specifies that the line not be drawn. A value of 1 specifies a solid line. The remaining values specify different segmented lines, as illustrated in Figure 14.22 on page 277.

PIE
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 30.17 on page 712.

Figure 30.17 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*;

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.

Functions

You can use the MIDPOINT variable only with the data coordinate systems 1, 2, 7, and 8, and with these functions:

BAR

DRAW

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

SYMBOL

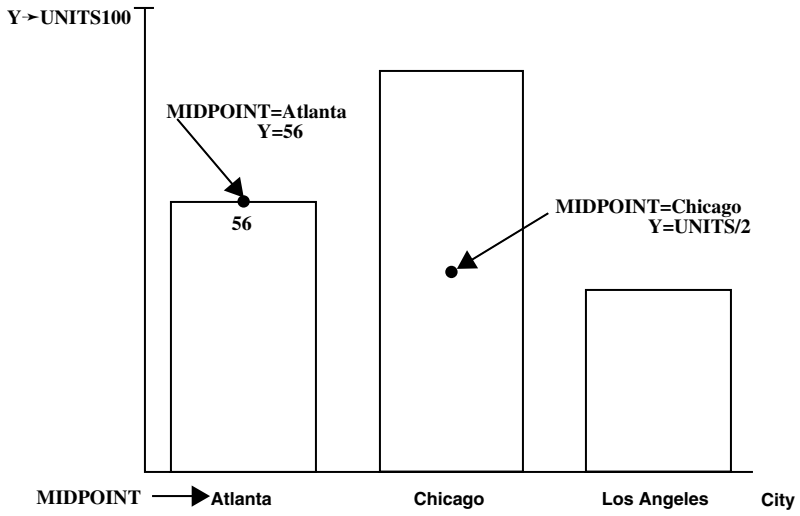
Details

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 30.18 on page 713 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 30.18 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 or the MIDPOINT and Y variables in the same data set used to annotate an HBAR chart can cause unexpected results. 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. △

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';

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:

Position	Right Aligned	Centered	Left Aligned
One cell above	1	2	3
Centered	4 | <	5 | +	6 | >
One cell below	7	8	9
Half cell above	A	B	C
Half cell below	D	E	F

These positions are illustrated in Figure 30.20 on page 716.

'0'
specifies a pause in the string in order to change an attribute, such as the color of the text.

Details

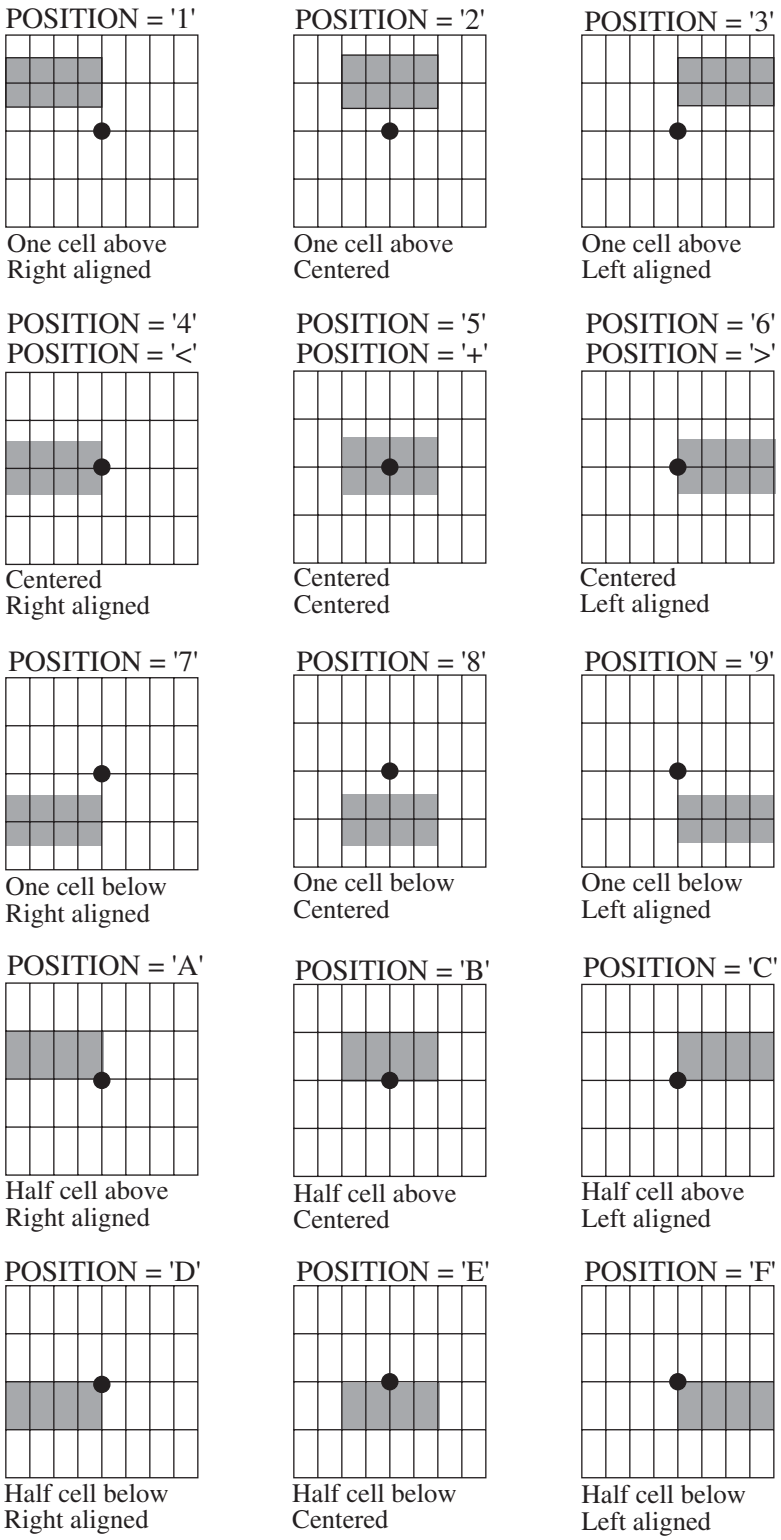
Stacking text strings To stack text strings, specify a different position value of each string. Figure 30.19 on page 715 shows two ways to stack text.

Figure 30.19 Combining POSITION Values to Stack Text

Positioning numeric labels The <, +, and > positions perform the same function as 4, 5, and 6, respectively, but are recommended only for labels that are numbers. 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 height or close to the same height as the text, you can use positions 4, 5, and 6 to center the labels.

Note: You cannot stack <, +, and > positions as you can 4, 5, and 6 positions. △

Figure 30.20 Effect of POSITION Values on Text Strings

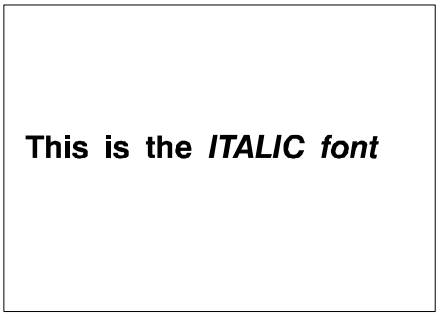


Changing attributes in the middle of a text string 0 is a special value to use when you want to pause and then continue a text string. With this value you can change colors, fonts, and so on in the middle of a line, while retaining the exact position of the text at

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 30.21 on page 717.

```
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 30.21 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*;

Functions

The ROTATE variable is function dependent.

If function is...	then the variable...
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*;

Functions

The SIZE variable is function dependent.

If function is...	then the variable...
ARROW	determines the thickness of the arrow being drawn.
DRAW, DRAW2TXT, FRAME, POLY, or POLYCONT	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.

Details

The SIZE variable uses the coordinate system that is specified by the “HSYS Variable” on page 707, which specifies the type of coordinate system used to generate the graph.

As the thickness of the line increases, it may 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 HEIGHT= on page 256 for details.

See Figure 30.7 on page 677 for examples of line thicknesses.

Figure 30.22 Sample Line Thicknesses Used with the SIZE Variable

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

Not supported by: ActiveX (Partial), Java

Syntax

STYLE=*'font-specification'* | 'NONE';

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*"

Note: If you specify a sytem font whose name is longer than eight characters, then you must enclose the name of the font in double quotes. △

NONE
specifies the default device-resident font.

See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 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.

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. △

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';

'TILE'
Uses copies of the image to fill the image area.

'FIT'
Stretches one instance of the image to fill the image area.

Details

This version of the STYLE variable can be used only with the "IMAGE Function" on page 682.

STYLE Variable (Arrows)

Specifies the type of arrowhead for arrows.

Type: character
Length: 8
Default: OPEN

Syntax

STYLE='CLOSED' | 'FILLED' | 'OPEN';

CLOSED
the arrowhead is shaped like an empty triangle.

FILLED
the arrowhead is shaped like a filled triangle.

OPEN
the arrowhead is shaped like a V.

STYLE Variable (Patterns)

Specifies a pattern for bars, pies, frames, and rectangles

Type: character
Length: 8
Default: EMPTY | PEMPTY | MEMPTY
Not supported by: Java (partial), ActiveX (partial)

Syntax

STYLE='*fill-pattern*';

fill-pattern
specifies a pattern to use with the graphics element. The value for *fill-pattern* is function-dependent:

Function
Valid Fill Pattern Values

BAR,FRAME

SOLID | S	Fill with a solid color.	
EMPTY | E	No fill.	
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.

Note: Java and ActiveX support only SOLID and EMPTY. EMPTY is the default 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.

PIE

PSOLID | PS	Solid fill.	
PEMPTY | PE	No fill, the default.	
P*density<style<angle>>*	*density*	Whole numbers 1 through 5 specify increasing thickness for the fill lines.
	style	N, the default, optionally specifies parallel fill lines; X optionally specifies crossed fill lines.
	angle	Optionally specifies 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.

Note: Java and ActiveX support only PSOLID and PEMPTY 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, the default.

M*density*<*style*<*angle*>>

density	Whole numbers 1 through 5 specify increasing thickness for the fill lines.
style	N, the default, optionally specifies parallel fill lines; X optionally specifies crossed fill lines.
angle	Optionally specifies the angle of the fill lines. Values range from 0 to 360. The angle is measured counterclockwise from the vertical. The default is 0°, which draws vertical lines.

Note: Java or ActiveX support only MSOLID and MEMPTY and default to MEMPTY is any other value is used. △

An illustration of these pattern styles is provided in the definition of the VALUE= option of the PATTERN statement.

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*;

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.

Functions

You can use the SUBGROUP variable only with the data coordinate system 1, 2, 7, or 8, and with these functions:

BAR

DRAW

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

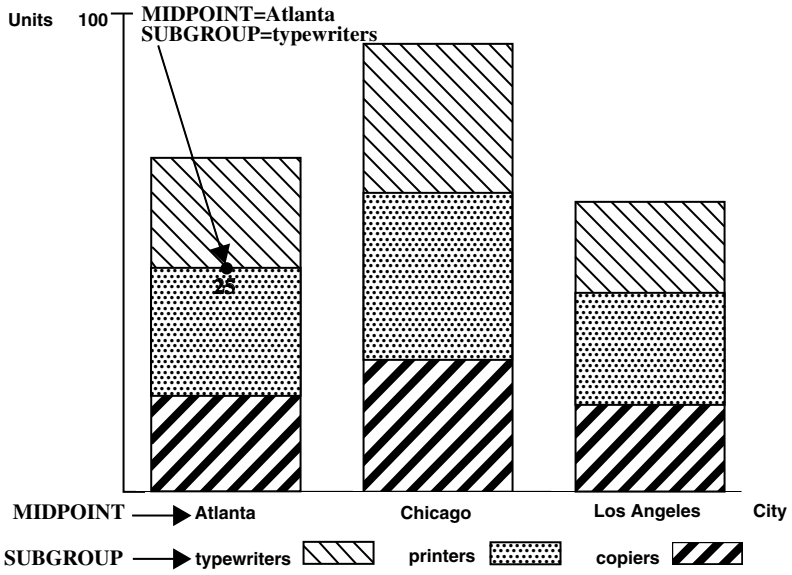
SYMBOL

Details

Using the SUBGROUP variable is similar to using the X and Y variables with data system coordinates to position the graphics elements in subgroup segments in vertical and horizontal bar charts.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 (so don't 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 30.23 on page 724 shows how the SUBGROUP variable works with the MIDPOINT variable to label the bars of a vertical bar chart.

Figure 30.23 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).

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*';

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 symbol from the special symbol table that is illustrated in Figure 14.21 on page 271. The maximum length for *special-symbol* is eight characters.

Functions

You can use the TEXT variable with these functions:

COMMENT

LABEL

SYMBOL

Details

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.

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' ;

B | A
specifies whether to draw the annotation before (B) or after (A) the graph. These values are not case sensitive. A missing value is equivalent to specifying B.

Note: The frame of some plot types is drawn before annotations where 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;
```

△

Functions

You can use the WHEN variable with these functions:

BAR

DRAW

DRAW2TXT

FRAME

LABEL

MOVE

PIE

PIECNTR

PIEXY

POINT

POLY

POLYCONT

SYMBOL

Details

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.

WIDTH Variable

Determines the thickness of a line.

Type: numeric

Length: 8

Default: 1

Syntax

WIDTH=*line-thickness*;

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*;

Functions

You can use the X variable with these functions:

ARROW

BAR

DRAW

IMAGE

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

SYMBOL

Note: The X or XC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the horizontal coordinate. △

Details

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 typed 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 is used with nontext functions and the value of the XLSTT variable is used with text functions.

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.

Default: the value of XLAST or XLSTT

Restrictions: Used only with output from the GCHART and GPLOT procedures. Ignored if the axes are numeric.

Syntax

XC='*character-type-horizontal-coordinate*';

Functions

You can use the XC variable with these functions:

BAR

DRAW

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

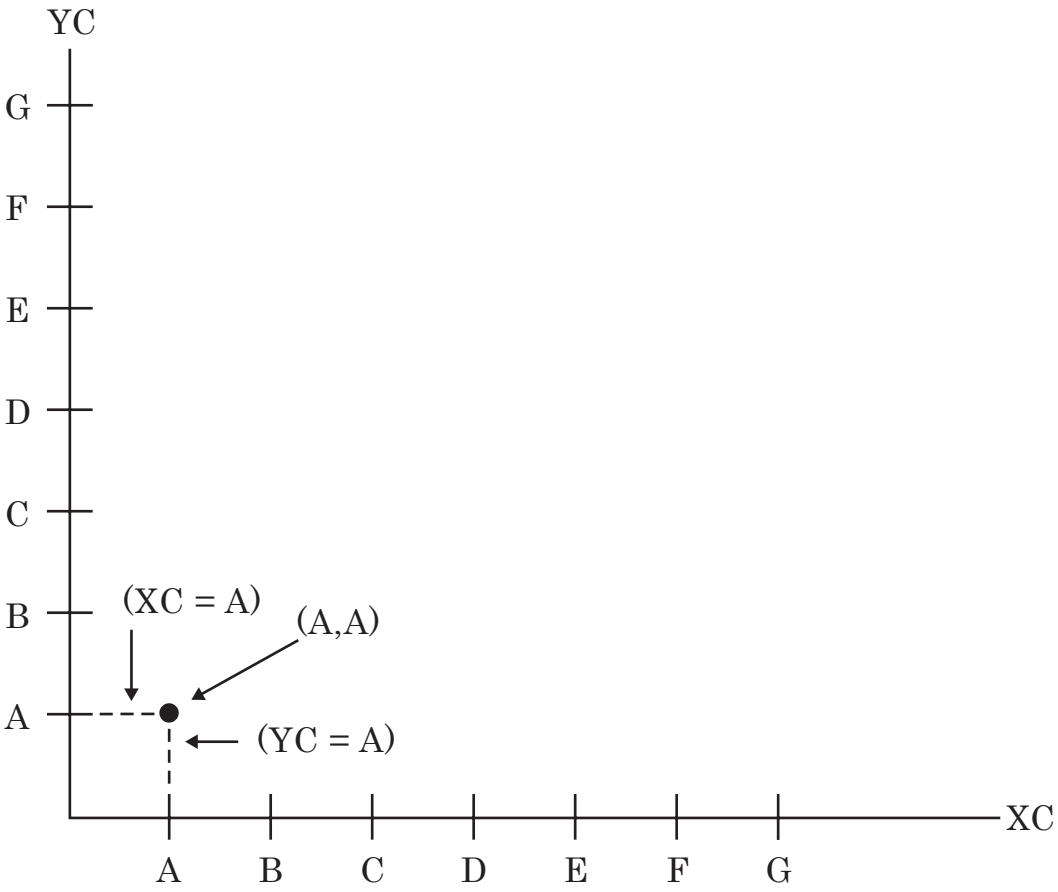
SYMBOL

Details

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 729.) If you use a value other than 2 for the XSYS variable, the graphics output is not displayed properly.

Figure 30.24 on page 729 illustrates the XC variable.

Figure 30.24 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. △

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*';

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

These values are also used by the HSYS and YSYS variables. See “Coordinate Systems” on page 650 for a description of the areas and coordinate systems.

Functions

You can use the XSYS variable with these functions:

ARROW

BAR

DRAW

FRAME

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

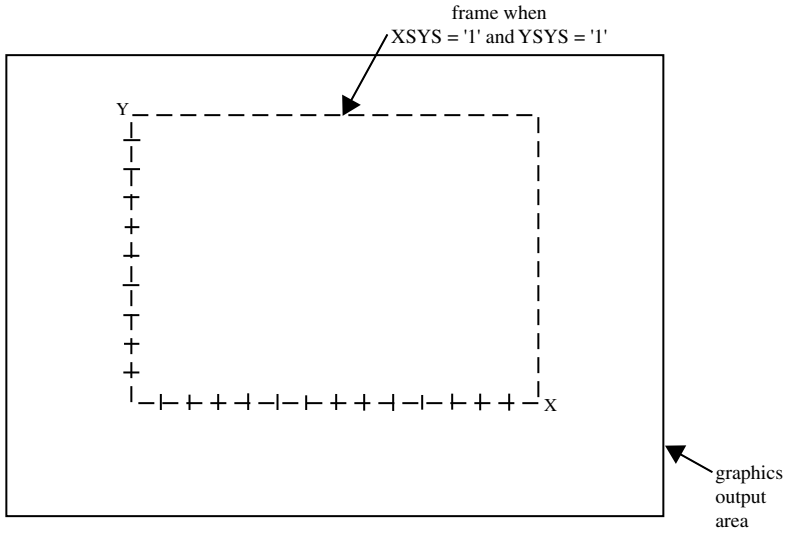
SYMBOL

The behavior of the XSYS variable is function-dependent for the following functions:.

BAR, DRAW 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.

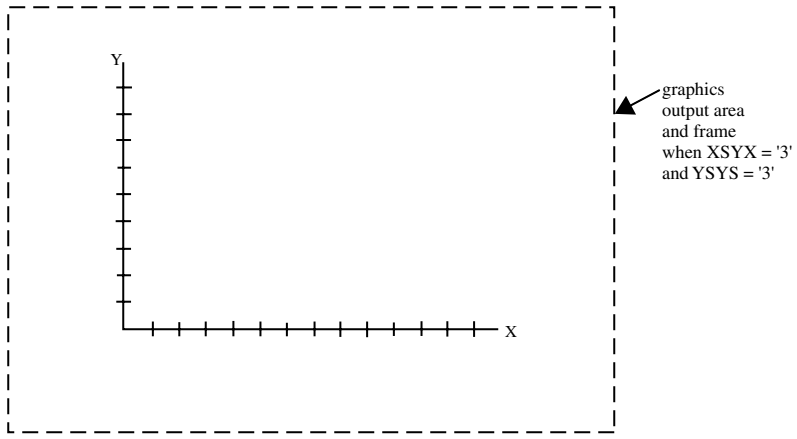
FRAME The XSYS and YSYS variables define the area enclosed by the frame. To draw a frame that encloses the axis area, use XSYS=’1’ and YSYS=’1’, as shown in the following figure.

Figure 30.25 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.

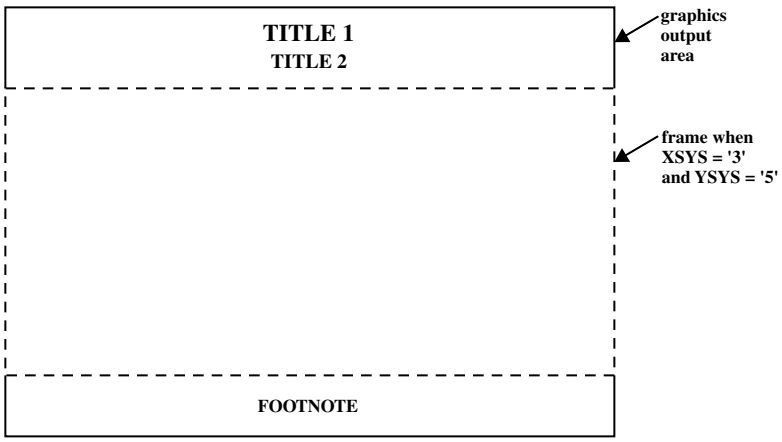
Figure 30.26 Frame Created When XSYS='3' and YSYS='3'



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 by specifying XSYS='3' and YSYS='5', as shown in the following figure.

Figure 30.27 Frame Created When XSYS='3' and YSYS='5'



Details

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. △

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*;

Functions

You can use the Y variable with these functions:

ARROW

BAR

DRAW

IMAGE

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

SYMBOL

Note: The Y or YC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the vertical coordinate. △

Details

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 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 YLAST is used for nontext functions and the value of YLSTT is used for text functions.

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.

Default: YLAST | YLSTT

Restrictions: Used only with output from the GCHART and GPLOT procedures. Ignored if the axes are numeric.

Syntax

YC='*character-type-vertical-coordinate*';

Functions

You can use the YC variable with these functions:

BAR

DRAW

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

SYMBOL

Details

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 734.) If you use a value other than 2 for the YSYS variable, the graphics output is not displayed properly.

See Figure 30.24 on page 729 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. △

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*';

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

These values are also used by the HSYS and XSYS variables. See "Coordinate Systems" on page 650 for a description of the areas and coordinate systems.

Functions

The YSYS variable is function-dependent, as defined in the "XSYS Variable" on page 729 You can use the YSYS variable with these functions:

ARROW

BAR

DRAW

FRAME

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

SYMBOL

Details

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. △

Z Variable

Identifies the *z* coordinate of where a graphics element is to be drawn.

Type: numeric

Length: 8

Default: none

Restrictions: 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*;

Functions

You can use the Z variable with these functions:

ARROW

BAR

DRAW

IMAGE

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

SYMBOL

Details

The Z variable uses the units that are specified in the ZSYS variable.

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*';

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 650 for a description of the areas and coordinate systems.

Functions

You can use the ZSYS variable with these functions:

ARROW

BAR

DRAW

IMAGE

LABEL

MOVE

PIE

PIECNTR

POINT

POLY

POLYCONT

SYMBOL

Details

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. △

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.

XLAST, YLAST Variables

Track the last values specified for the X and Y variables when X and Y are used with nontext functions.

Details

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

PIECNTR

PIEXY

POP

PUSH

SWAP

TXT2CNTL

XLSTT, YLSTT Variables

Track the last position for the X and Y variables when X and Y are used with text-handling functions.

Details

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

DRAW2TXT

POP

PUSH

SWAP

TXT2CNTL

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 on accessing and using macros, and for a summary of operations performed by the Annotate macros, see "Using Annotate Macros" on page 759.

%ANNOMAC Macro

Compiles Annotate macros and makes them available for use.

Variables written out: none directly

Syntax

%ANNOMAC;

Details

In a SAS session, you must submit the ANNOMAC macro before you can use the Annotate macros.

%ARROW Macro

Draws an arrow from (X1, Y1) to (X2,Y2).

Variables written out: ANGLE, COLOR, FUNCTION, LINE, SIZE, STYLE, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%ARROW (*x1*, *y1*, *x2*, *y2*, *color*, *line*, *size*, *angle*, *style*);

x1, y1
specify coordinates for the start point of the arrow. Values can be coordinate numbers, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

x2, y2
specify coordinates for the end point of the arrow. Values can be coordinate numbers, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

color

specifies the color of the line using a character string without quotation marks. For details, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

line
specifies the length of the sides of the arrowhead. The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate "LINE Variable" on page 710 for the ARROW function.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable. For valid numeric values, see the Annotate "SIZE Variable" on page 718 for the ARROW function.

angle
specifies the angle of the tip of the arrowhead. The value can be a number, a numeric constant, or a numeric variable. For valid numeric values, see the Annotate "ANGLE Variable" on page 700 for the ARROW function.

style
specifies the type of arrowhead. You can specify CLOSED, FILLED, or OPEN. For more information about the values, see "STYLE Variable (Arrows)" on page 720.

Details

The point from which the line is drawn is usually set with the MOVE macro.

%BAR, %BAR2 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.

Variables written out: COLOR, FUNCTION, LINE, STYLE, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%BAR (*x1*, *y1*, *x2*, *y2*, *color*, *line*, *style*);

%BAR2(*x1*, *y1*, *x2*, *y2*, *color*, *line*, *style*, *width*);

x1, y1
specify the location of the first corner of the bar. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

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.

color
specifies the outline color and optional fill color using a character string without quotation marks. For details, see the Annotate "COLOR Variable" on page 703.

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. For valid values, see the Annotate "LINE Variable" on page 710 for the BAR function.

style
specifies the fill pattern for the bar using a character string without quotation marks. For valid values, see the Annotate "STYLE Variable (Patterns)" on page 721 for the BAR function.

width
specifies the width of the outline and optional fill lines. The value can a number, a numeric constant, or a numeric variable. For details and valid values, see the Annotate "SIZE Variable" on page 718 for the DRAW function.

%CENTROID Macro

Retrieves the centroids of polygons

Variables written out: X, Y, id variables

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%CENTROID (*input-data-set*, *output-data-set*, *list-of-id-variables*);

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 to be assigned the centroid coordinates of each observation in the input-data-set. There will be one observation for each unique set of ID values. If you specify more than one ID variable, then separate each variable with a space.

%CIRCLE Macro

Draws an empty circle with the center at (*x*, *y*).

Variables written out: ANGLE, FUNCTION, ROTATE, SIZE, STYLE, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%CIRCLE (*x*, *y*, *size*, *color*);

x*, *y
specify coordinates for the center of the circle. Values can be coordinate numbers, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

size
specifies the radius of the circle. The value can be a number, a numeric constant, or a numeric variable. For details and valid values, see the Annotate "SIZE Variable" on page 718.

color
specifies the color of the circle using a character string without quotation marks. For details, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

See Also

"%SLICE Macro" on page 756 to draw a filled circle.

%CNTL2TXT Macro

Copies the values of the internal coordinates (XLAST, YLAST) to the text coordinate (XLSTT, YLSTT).

Variables written out: FUNCTION

Internal variables updated: XLSTT, YLSTT

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%CNTL2TXT;

Details

The %CNTL2TXT macro is useful when you are calculating the position of labels on a graph. For an example, see "CNTL2TXT Function" on page 673.

%COMMENT Macro

Inserts a comment into an Annotate data set.

Variables written out: FUNCTION, TEXT

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%COMMENT (*text-string*);

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. For details, see the Annotate "TEXT Variable" on page 724.

%DCLANNO Macro

Automatically sets the correct length and data type for all Annotate variables except the TEXT variable.

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%DCLANNO;

%DRAW Macro

Draws a line from (XLAST, YLAST) to the specified coordinate.

Variables written out: COLOR, FUNCTION, LINE, SIZE, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%DRAW (*x*, *y*, *color*, *line*, *size*);

x, y
specify coordinates for the end point of the line. Values can be coordinate numbers, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

color
specifies the color of the line using a character string without quotation marks. For details, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

line
specifies the line type (continuous or segmented). The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate "LINE Variable" on page 710 for the DRAW function.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable. For valid numeric values, see the Annotate "SIZE Variable" on page 718 for the DRAW function.

Details

The point from which the line is drawn is usually set with the MOVE macro.

%DRAW2TXT Macro

Draws a line from the coordinate (XLAST, YLAST) to the text coordinate (XLSTT, YLSTT).

Variables written out: COLOR, FUNCTION, LINE, SIZE

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%DRAW2TXT (*color*, *line*, *size*);

color
specifies the color of the line using a character string without quotation marks. For details, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

line
specifies the line type (continuous or segmented). The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate "LINE Variable" on page 710 for the DRAW function.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate "SIZE Variable" on page 718 for the DRAW function.

%FRAME Macro

Draws a border around the portion of the display area defined by the reference system and optionally fills the area.

Variables written out: COLOR, FUNCTION, LINE, SIZE, STYLE

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%FRAME (*color*, *line*, *size*, *style*);

color
specifies the outline color and the optional fill color using a character string without quotation marks. For details, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

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. For valid numeric values, see the Annotate "LINE Variable" on page 710 for the DRAW function.

size
specifies the width of the frame outline and fill lines. The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate "SIZE Variable" on page 718 for the DRAW function.

style
specifies the fill pattern for the frame using a character string without quotation marks. For valid values, see the Annotate "STYLE Variable (Patterns)" on page 721 for the FRAME function.

Details

See "%SYSTEM Macro" on page 758 for information on setting the reference system.

%LABEL Macro

Places a text label at the specified coordinates.

Variables written out: ANGLE, COLOR, FUNCTION, POSITION, ROTATE, SIZE, STYLE, TEXT, X, Y

Internal variables updated: XLSTT, YLSTT

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%**LABEL** (*x*, *y*, *text-string*, *color*, *angle*, *rotate*, *size*, *style*, *position*);

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. For details, see the Annotate "X Variable" on page 726.

text-string
specifies the text of the label. The value can be a character variable name or a character string enclosed in quotation marks. For details, see the Annotate "TEXT Variable" on page 724.

color
specifies the color of the text string using a character string without quotation marks. For details, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

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. For valid values, see the Annotate "ANGLE Variable" on page 700 for the LABEL function. The *x*, *y* coordinates specify the pivot point, and the *position* parameter positions the text relative to *x*, *y*.

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. For valid values, see the Annotate "ROTATE Variable" on page 717.

size
specifies the size of the text string. The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate "SIZE Variable" on page 718 for the LABEL function.

style
specifies the text font, using a character string without quotation marks. For valid values, see the Annotate "STYLE Variable (Fonts)" on page 719.

position
specifies the placement and alignment of the text string relative to the *x*, *y* coordinates, using a text string without quotation marks. For valid values, see the Annotate "POSITION Variable" on page 714.

%LINE Macro

Draws a line between two sets of coordinates.

Variables written out: COLOR, FUNCTION, LINE, SIZE, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%LINE (*x1*, *y1*, *x2*, *y2*, *color*, *line*, *size*);

x1, y1
specify the coordinates of the start of the line. Values can be numbers, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726 variable.

x1, y2
specify the coordinates of the end of the line. Values can be numbers, numeric constants, or numeric variables.

color
specifies the color of the line using a character string without quotation marks. For valid values, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

line
specifies the line type, which can be continuous or segmented. The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate"LINE Variable" on page 710 for the DRAW function.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate "SIZE Variable" on page 718 for the DRAW function.

%MAPLABEL 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

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%MAPLABEL (*map-dataset*,
attr-dataset,*output-dataset*,*label-var*,*id-list*,*font=font_name*,*color=n*,*size=n*,*hsys=n*);

map-dataset
the name of the map to be annotated.

Note: If you specify a feature table in your GMAP procedure step, then you should specify the map data set that corresponds to that feature table. △

attr-dataset
the name of the dataset 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 vars that you would issue in PROC GMAP to create the map. These values need to be on both the map and the 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 instance, 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 (Fonts)" on page 719 variable.

color
specifies a value for the "COLOR Variable" on page 703 variable.

size
specifies a value for the "SIZE Variable" on page 718 variable. Defaults to 2.

hsys
specifies a value for the "HSYS Variable" on page 707 variable. Defaults to 3.

%MOVE Macro

Moves to the (*x*, *y*) coordinate.

Variables written out: FUNCTION, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%**MOVE** (*x*, *y*);

x*, *y
specify new coordinates for the next annotation. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

%PIEXY Macro

Calculates a point in relation to the latest pie slice.

Variables written out: ANGLE, FUNCTION, SIZE, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%PIEXY (*angle*, *size*);

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. For details, see the Annotate "ANGLE Variable" on page 700 for the PIEXY function.

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. For details and valid values, see the Annotate "SIZE Variable" on page 718 for the PIEXY function.

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 689 has not been processed, you can get erroneous results.

%POLY, %POLY2 Macro

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,

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%POLY (*x*, *y*, *color*, *style*, *line*);

%POLY2(*x*, *y*, *color*, *style*, *line*, *width*);

x, y
specify the starting point for a new polygon. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate or the names of the Annotate variables "X Variable" on page 726.

color
specifies the optional polygon fill color using a character string without quotation marks. For valid values, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter. To specify the color of the polygon outline, see the "%POLYCONT Macro" on page 750.

style
specifies the fill pattern for the polygon, using a character string without quotation marks. For valid values, see the Annotate "STYLE Variable (Patterns)" on page 721 for the POLY function.

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. For valid values, see the Annotate "LINE Variable" on page 710 for the POLY function.

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. For details and valid values, see the Annotate "SIZE Variable" on page 718 for the POLY function.

See Also

"POLY Function" on page 692

%POLYCONT Macro

Continues drawing the polygon to the next specified coordinates.

Variables written out: COLOR, FUNCTION, X, Y

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%POLYCONT (*x*, *y*, *color*);

x, y
specify the end point of the next line in the polygon. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

color
specifies the color of the polygon outline using a character string without quotation marks. For valid values, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

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 Macro" on page 750 or %POLY2 macro.

%POP Macro

Removes the coordinates (XLAST, YLAST) and (XLSTT, YLSTT) from the LIFO system stack and updates the internal coordinate pairs with these retrieved values.

Variables written out: FUNCTION

Internal variables updated: XLAST, YLAST, XLSTT, YLSTT

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%POP;

Details

Use the %POP macro when you want to access the values of the XLAST, YLAST, XLSTT, and YLSTT variables that you previously stored with the %PUSH macro. For more information, see "XLAST, YLAST Variables" on page 737, "XLSTT, YLSTT Variables" on page 738, and "%PUSH Macro" on page 752.

%PUSH Macro

Enters the coordinates (XLAST, YLAST) and (XLSTT, YLSTT) in a LIFO system stack.

Variables written out: FUNCTION, internal coordinates

Internal variables updated: XLAST, YLAST, XLSTT, YLSTT

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%PUSH;

Details

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 657 for a description of the LIFO stack.

%RECT Macro

Draws a rectangle with diagonal corners at two specified points.

Variables written out: COLOR, FUNCTION, LINE, SIZE, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%RECT (*x1*, *y1*, *x2*, *y2*, *color*, *line*, *size*) ;

x1, y1
specify the coordinates of the first corner of the rectangle. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

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.

color
specifies the color of the rectangular line using a character string without quotation marks. For valid values, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

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. For details, see the Annotate "LINE Variable" on page 710 for the DRAW function.

size
specifies the width of the line. The value can be a number, a numeric constant, or a numeric variable. For valid values, see the "SIZE Variable" on page 718 for the DRAW function.

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 Macros" on page 740.

%SCALE 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

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%SCALE (*ptx*, *pty*, *x1*, *y1*, *x2*, *y2*, *vx1*, *vy1*, *vx2*, *vy2*);

ptx, pty
specifies the coordinates to scale. Values can be numbers, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

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

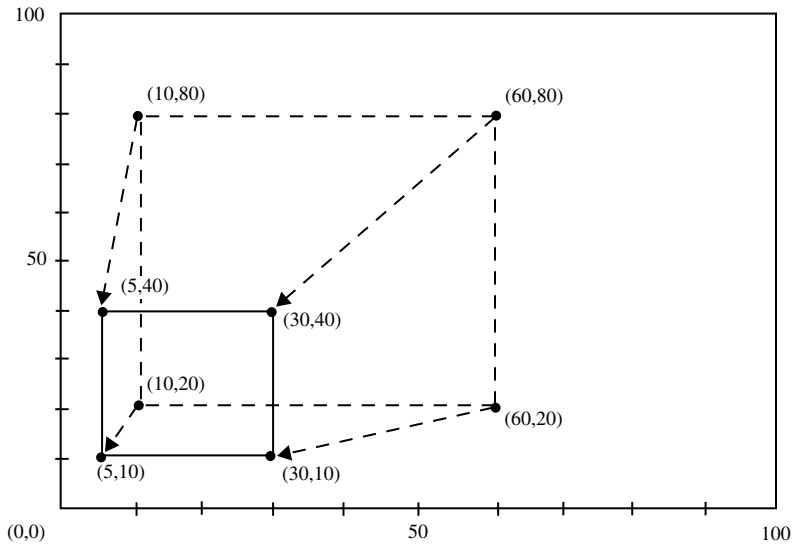
The %SCALE macro reduces or enlarges Annotate graphics elements that use two-dimensional, numeric coordinates. The %SCALE macro does not affect graphics elements that are drawn with text functions.

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 Macro" on page 754.

The following example uses the %SCALE macro to reduce *x* and *y* coordinates by 50 percent, as shown in Figure 30.28 on page 754:

```
%SCALE(x, y, 0, 0, 100, 100, 0, 0, 50, 50);
```

Figure 30.28 Using the %SCALE Macro to Reduce the Size of a Box



%SCALET 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

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%SCALET (*ptx*, *pty*, *x1*, *y1*, *x2*, *y2*, *vx1*, *vy1*, *vx2*, *vy2*);

ptx, pty
specifies the coordinates to scale. Values can be numbers, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

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

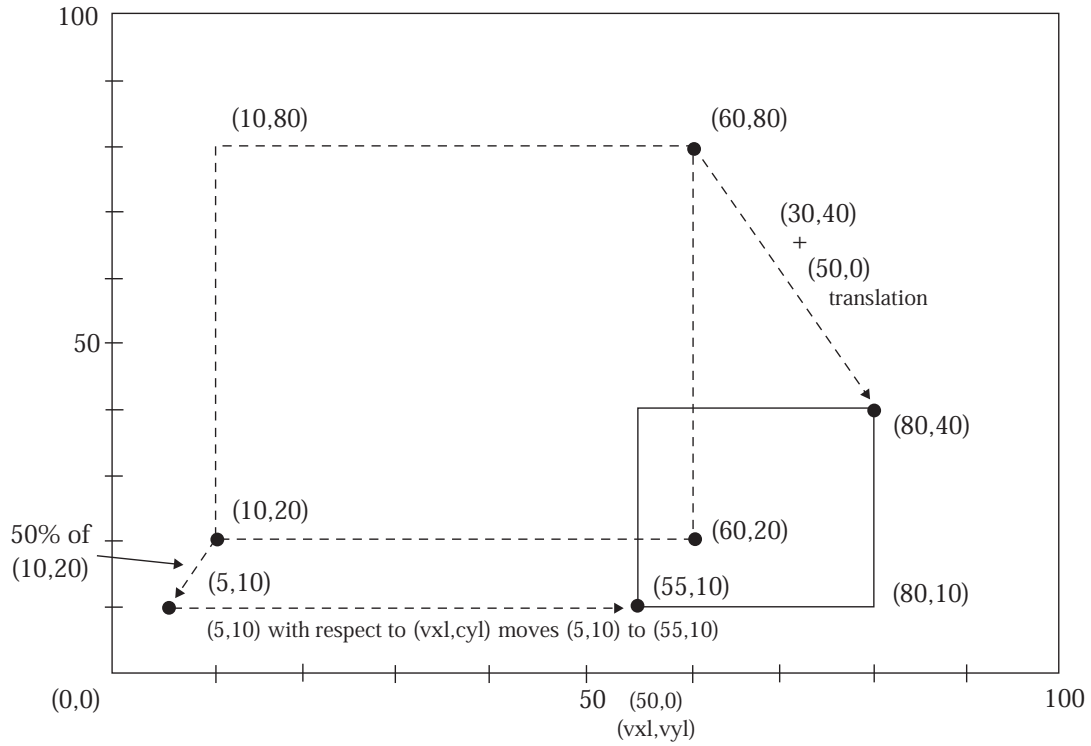
The %SCALET macro reduces or enlarges Annotate graphics elements that use two-dimensional numeric coordinates. The %SCALET macro does not affect graphics elements that are drawn with text functions.

The difference between the %SCALET and %SCALE 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 percent and plots the new coordinates with respect to (50, 0), as shown in Figure 30.29 on page 756:

```
%SCALET(x, y, 0, 0, 100, 100, 50, 0, 100, 50);
```

Figure 30.29 Using the %SCALET Macro to Reduce the Size of a Box



%SEQUENCE 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

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%SEQUENCE (*when*);

when
Values can be BEFORE or AFTER, as defined for the Annotate "WHEN Variable" on page 725.

%SLICE Macro

Draws a arc, pie slice, or circle, with available line types, colors, and fill types.

Variables written out: ANGLE, COLOR, FUNCTION, LINE, ROTATE, SIZE, STYLE, X, Y

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%SLICE (*x*, *y*, *angle*, *rotate*, *size*, *color*, *style*, *line*);

x, y
specify the center point of the arc. Values can be numbers, numeric constants, or numeric variables. For details, see the Annotate "X Variable" on page 726.

angle
specifies the starting point of the arc. The value can be a number, a numeric constant, or a numeric variable. For details and valid values, see the Annotate "ANGLE Variable" on page 700 for the PIE function.

rotate
specifies the sweep of the arc. The value can be a number, a numeric constant, or a numeric variable. For valid values, see the Annotate "ROTATE Variable" on page 717 for the PIE function.

size
specifies the radius of the arc. The value can be a number, a numeric constant, or a numeric variable. For details, see the Annotate "SIZE Variable" on page 718.

color
specifies the color of the arc outline and optional fill using a character string without quotation marks. For valid values, see the Annotate "COLOR Variable" on page 703. Use an asterisk (*) to specify the previous value of the *color* parameter.

style
specifies the fill pattern for the slice or circle, using a character string without quotation marks. For details and valid values, see the Annotate "STYLE Variable (Patterns)" on page 721 for the PIE function.

line
specifies which lines of a pie slice are to be drawn. The value can be a number, a numeric constant, or a numeric variable. For valid values and details, see the "LINE Variable" on page 710 for the PIE function.

%SWAP Macro

Exchanges control between (XLAST, YLAST) and text (XLSTT, YLSTT) coordinates.

Variables written out: FUNCTION

Internal variables updated: XLAST, YLAST, XLSTT, YLSTT

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%SWAP;

%SYSTEM Macro

Defines the Annotate reference systems and the XSYS, YSYS, and HSYS variables.

Variables written out: HSYS, XSYS, YSYS

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%SYSTEM (*xsys*, *ysys*, *hsys*);

xsys*, *ysys*, *hsys
specify values that represent a coordinate system and an area of the output, as defined for the Annotate "XSYS Variable" on page 729. The default is %SYSTEM (4, 4, 4).

Details

Note: Not all coordinate systems are valid with all Annotate variables or all SAS/GRAPH procedures. See "Annotate Functions" on page 669 for any restrictions that apply to the variable that you want to use. △

The ZSYS variable cannot be set through this macro. Use an explicit variable assignment instead:

```
zsys="value"; output;
```

See Coordinate Systems"Coordinate Systems" on page 650 for a description of the areas and coordinate systems.

%TXT2CNTL Macro

Assigns the values of the text (XLSTT, YLSTT) coordinates to the control (XLAST, YLAST) coordinates.

Variables written out: FUNCTION

Internal variables updated: XLAST, YLAST

Prerequisite: You must run the %ANNOMAC macro before using any other annotate macros. For more information, see "Making the Macros Available" on page 759.

Syntax

%TXT2CNTL;

Details

Use the %TXT2CNTL macro when you want nontext functions to use the ending position of a text string as a starting or ending point.

Using Annotate Macros

Macro Structure

The general form of an Annotate macro is

```
%MACRO (parameters);
```

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 an observation.

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 literals, that is 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.

The Annotate facility assigns the parameter values to the corresponding Annotate variables. Therefore, the observations in an Annotate data set that is created with macros that look the same as the ones that you created with 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

To use Annotate macros, you must provide your program with access to the data set that contains the macros, and you must compile the macros before you use them. Check with your SAS Software Consultant to find out if the fileref for the data set that contains the Annotate macros that are supplied with SAS/GRAPH software is allocated automatically at your site. Then access the Annotate macros in one of these ways:

- If the fileref is not set automatically, find out where the Annotate macros are stored and allocate a fileref that points to the data set:

  ```
  filename fileref "external-file";
  ```

  Then include the Annotate macros in your session:

  ```
  %include fileref (annomac);
  ```

- If the fileref is set automatically, compile the Annotate macros and make them available by simply submitting the ANNOMAC macro:

```
%annomac;
```

Note: The ANNOMAC macro must be run before any other Annotate macros are used in a SAS session. You will see a message in the SAS log that indicates that the Annotate macros are now available. The message also shows you how to get help for using the macros. △

Annotate Macro Task Summary

The following table summarizes the tasks performed by the Annotate macros.

Table 30.1 Tasks with Annotate Macros

If you want to...	Use this macro...
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*, *style*);
draw a bar	%BAR(*x*1, *y*1, *x*2, *y*2, *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*);
draw a line	%LINE(*x*1, *y*1, *x*2, *y*2, *color, line, size*);
draw a pie slice or arc	%SLICE(*x, y, angle, rotate, size, color, style, line*);
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, x*0, *y*0, *x*1, *y*1, *x*0, *vy*0, *vx*1, *vy*1);
scale input	%SCALE(*ptx, pty, x*0, *y*0, *x*1, *y*1, *x*0, *vy*0, *vx*1, *vy*1);
set the coordinate system for the observation	%SYSTEM(*xsys, ysys, hsys*);

If you want to...	Use this macro...
set when to draw an observation	%SEQUENCE(*when*);
write a comment to the data set	%COMMENT(*text*);

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 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 may 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 percent. 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 quotes 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 quotes 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 quotes 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 may 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 re-use.
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 have more values moved with the POP function than are stored with the PUSH function. At least one PUSH must occur {it 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 line-type table"Specifying Line Types" on page 276.

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 quotes 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 quotes 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

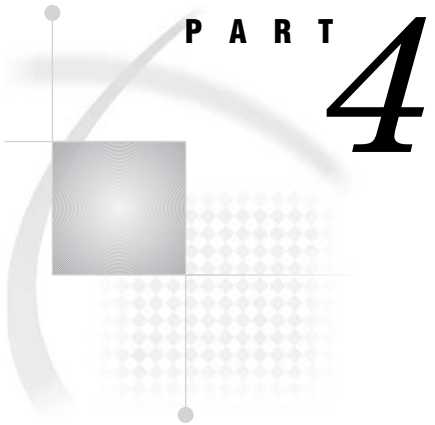
Explanation: The length is incorrect for variable indicated. Either the length of the character string exceeds the length for the variable specified in a LENGTH statement, or the variable was not specified in a LENGTH statement.

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 “Annotate Variables” on page 700.



PART 4

The Data Step Graphics Interface

CHAPTER

31

The DATA Step Graphics Interface

Overview

The DATA Step Graphics Interface (DSGI) enables you to create graphics output within the DATA step or from within a Screen Control Language (SCL) application. Through DSGI, you can call the graphics routines used by SAS/GRAPH software to generate a custom graph, or to add features to an existing graph. You can use DSGI to write a custom graphics application in conjunction with all the power of the programming statements accessible by the DATA step.

DSGI provides many of the same features as the Annotate facility, but it also has many advantages over the Annotate facility.

- you can use DSGI functions and routines through SCL
- you can save disk space. DSGI graphics can be generated through the DATA step without creating an output data set. The graphics output is stored as a catalog entry in the catalog you select, and can be displayed after the DATA step is submitted.
- DSGI generates graphics faster than the Annotate facility. With the Annotate facility, you create a data set and then submit a PROC step to display the graphics output. In DSGI, you eliminate the PROC step because the graphics output is generated after the DATA step.
- DSGI supports viewports and windows, which enable you to specify the dimensions, position, and scale of the graphics output. You cam include multiple graphs in the same graphics output.

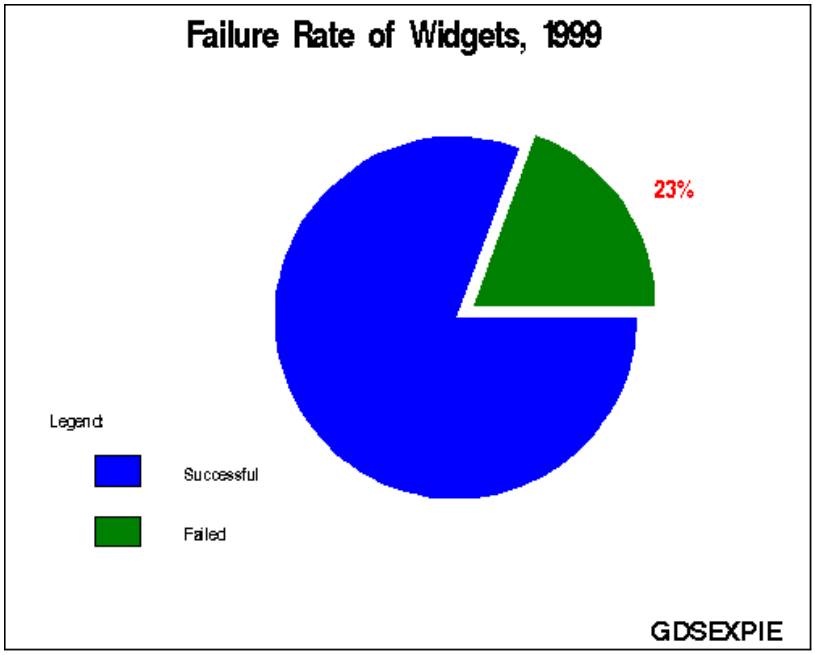
Consider using the Annotate facility for enhancing procedure output. and using DSGI for creating custom graphics without using a graphics procedure.

DSGI is based on the Graphics Kernel System (GKS) standard, although it does not follow a strict interpretation, nor is it implemented on a particular level of GKS. GKS was used to provide a recognizable interface to the user. Because of its modularity, the standard allows for enhancements to DSGI without the side effect of converting programs between versions of SAS/GRAPH software.

The concepts used to create graphics output with DSGI are explained. An overview of the functions and routines used in DSGI are provided. For complete details of each function and routine, see Chapter 32, “DATA Step Graphics Interface Dictionary,” on page 813.

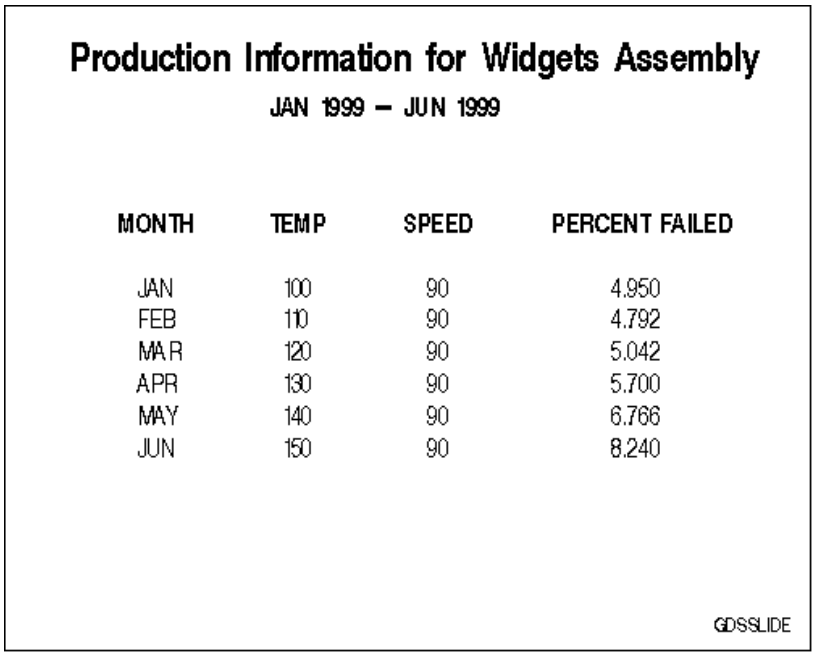
DSGI Funtions

Figure 31.1 Pie Chart Created with DSGI Functions



DSGI Statements

Figure 31.2 Text Slide Created with DSGI Statements



Syntax

DSGI uses GASK routines and functions to draw graphics elements. These statements have the following syntax:

CALL GASK(*operator*, *arguments*);

return-code-variable=function-name (*operator*, *arguments*);

where

arguments are the additional required variables or values for the routine or function.

return-code-variable is an arbitrary name and can be any numeric variable name. It holds the return code upon execution of the function.

function-name is the DSGI command you want to execute and must be one of the following: GDRAW, GINIT, GPRINT, GRAPH, GSET, or GTERM.

operator is a character string that names the function you either want to submit or for which you want the current settings. When used with functions, *operator* can take different values depending on *function-name*.

Requirements

When using DSGI statements, the following formats for arguments must be used:

- All *x* and *y* coordinates are expressed in units of the current window system. (See "The Current Window System" on page 776 for details.)
- The arguments used with DSGI functions can be expressed as either constants or variables. The arguments used with GASK routines must be variable names since values are returned through them. See Chapter 32, "DATA Step Graphics Interface Dictionary," on page 813 for a complete explanation of each argument used with DSGI functions and routines.
- All arguments that are character constants must be enclosed in either single or double quotation marks.

Applications of the DATA Step Graphics Interface

With the DATA Step Graphics Interface you can

- enhance existing graphs
- create custom graphs.

Enhancing Existing Graphs

You can use DSGI to enhance existing graphs. You can add text and other graphics elements. You can also alter the appearance of the existing graph by scaling or reducing it. To enhance a graph produced by a SAS/GRAPH graphics procedure, insert the existing graph into graphics output being generated with DSGI.

To insert a graph, provide:

- the catalog in which the existing graph is located
- the name of the existing graph
- the coordinates of the place in the graphics output where you want to insert the existing graph
- a square coordinate system ((0,0) to (100,100))
- the statements to draw enhancements to the existing graph.

The coordinates that DSGI uses to position existing graphs, enhancements to that graph, or graphics elements are based on units of percent of the window system currently defined. See "Using Viewports and Windows" on page 791.

Creating Custom Graphs

You can produce custom graphs with DSGI without using a data set to produce the graphics output. DSGI enables you to create

- arcs
- bars
- ellipses
- elliptical arcs
- lines
- markers
- pie slices
- polygons (filled areas)
- text.

To create custom graphs, provide:

- DSGI statements to draw graphics elements
- the coordinates of the graphics elements in the output.

You can also specify the color, pattern, size, style, and position of the graphics elements.

Using the DATA Step Graphics Interface

These sections provide general information about using DSGI.

Summary of Use

To create graphics output using DSGI:

1. on a grid that matches the dimensions of the graphics output, sketch the output you want to produce
2. determine the coordinates of each graphics element
3. in the DATA step, write the program to generate the graphics output

   To use the DSGI interface:

   a. initialize DSGI
   b. open a graphics segment
   c. generate graphics elements
   d. close the graphics segment
   e. end DSGI.
4. Submit the DATA step with a final RUN statement to display the output.

Note: The DISPLAY graphics option must be in effect for the graphics output to be displayed. See Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for more information about the DISPLAY graphics option. △

Producing and Storing DSGI Graphs

When you create or enhance graphs with DSGI, the DSGI graphics are displayed and stored as part of the graphics output. When you execute the DATA step, DSGI creates a catalog entry using the name from the GRAPH("CLEAR", . . .)function.

DSGI uses the name DSGI if you have not specified a name with the GRAPH("CLEAR", . . .)function. The catalog entry is stored in WORK.GSEG, unless you specify another catalog with the GSET("CATALOG", . . .)function.

If you create another graph using a name that matches an existing catalog entry in the current catalog, DSGI uses the default naming conventions for the catalog entry. See "About GRSEGs" on page 89 for a description of the conventions used to name catalog entries.

If you want to store your output in a permanent library or in a different temporary catalog, use the GSET("CATALOG", . . .)function. This function enables you to specify the libref, and catalog name for the output catalog. Before you use the GSET("CATALOG", . . .)function, assign a libref using the LIBNAME statement.

You can display DSGI graphics output stored in catalog entries multiple times, using the GREPLAY procedure or the GRAPH window.

Structure of DSGI Data Sets

The DSGI DATA step is usually not written to produce an output data set. Unlike data sets created by the Annotate facility, which contain observations for each graphics element drawn. DSGI does not usually create an observation for each graphics primitive. Only variables created in the DATA step are written to the output data set.

You can output as many observations to the data set as you want. To output these values, you must use the OUTPUT statement. You can also use any valid SAS DATA step statements in a DSGI DATA step. See *SAS Language Reference: Dictionary* for information about the statements used in the DATA step.

SAS/GRAPH Global Statements with DSGI

Some SAS/GRAPH global statements can be used with DSGI programs. DSGI recognizes FOOTNOTE, GOPTIONS, and TITLE statements. When TITLE and FOOTNOTE statements are used, the output from DSGI statements is placed in the procedure output area. See "How Graphic Elements are Placed in the Graphics Output Area" on page 65 for an explanation of how space in graphics output is allocated to titles and footnotes.

Note: DSGI ignores AXIS, LEGEND, NOTE, PATTERN, and SYMBOL statements. △

Some DSGI functions override the graphics options. The following table lists the DSGI functions that directly override graphics options. For details about the graphics options, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

DSGI Function	Graphics Option That Is Overridden
GSET('CBACK', . . .)	CBACK=
GSET('COLREP', . . .)	COLORS=
GSET('DEVICE', . . .)	DEVICE=
GSET('HPOS', . . .)	HPOS=
GSET('HSIZE', . . .)	HSIZE=
GSET('VPOS', . . .)	VPOS=
GSET('VSIZE', . . .)	VSIZE=
GSET('TEXCOLOR', . . .)	CTEXT=
GSET('TEXFONT', . . .)	FTEXT=
GSET('TEXHEIGHT', . . .)	HTEXT=

Operating States

The operating state of DSGI determines which functions and routines can be issued at any point in the DATA step. You can submit a function or routine only when the

operating state is appropriate. Reference “How Operating States Control the Order of DSGI Statements” on page 785 for how functions and routines should be ordered within the operating states.

The operating states defined by DSGI are:

GKCL	facility closed, the initial state of DSGI. No graphical resources have been allocated.
GKOP	facility open. When DSGI is open, you can check the settings of the attributes.
SGOP	segment open. At this point, graphics output primitives can be generated.
WSAC	workstation active. When the workstation is active, it can receive DSGI statements.
WSOP	workstation open. In this implementation, the graphics catalog, either the default or the one specified through the GSET(“CATALOG”, . . .)command, is opened or created.

Refer to individual functions and routines in Chapter 32, “DATA Step Graphics Interface Dictionary,” on page 813 for the operating states from which a function or routine can be issued.

The Current Window System

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

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

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

Debugging DSGI Programs

When DSGI encounters an error in a program, it flags the statement in the SAS log and displays a description of the error. (To receive SAS System messages, GSET(“MESSAGE”, . . .)must be ON.) The description provides you with an explanation of the error. The description might also provide a return code. If you get a return code, you can refer to “Return Codes for DSGI Routines and Functions” on page 908 for a description of the error and why it might have occurred.

Some of the most common errors in DSGI programs are:

- syntax errors
- an invalid number of arguments for the function or routine
- a function or routine being executed in an operating state that is not correct for the function or routine.

DSGI Graphics Summary

The following sections summarize the functions and routines you can use to create graphics output with DSGI.

DSGI Functions

DSGI provides functions that

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

Table 31.1 on page 777 summarizes the types of operations available and the functions used to invoke them. Refer to Chapter 32, "DATA Step Graphics Interface Dictionary," on page 813 for details about each function.

Table 31.1 DATA Step Graphics Interface Functions

DSGI Operations	Associated Function	Function Description
Bundling Attributes (valid values for *xxx* are FIL, LIN, MAR, and TEX)		
	GSET('ASF', . . .)	sets the aspect source flag of an attribute
	GSET('*xxx*INDEX', . . .)	selects the bundle of attributes to use
	GSET('*xxx*REP', . . .)	assigns attributes to a bundle
Setting Attributes That Affect Graphics Elements		
color index	GSET('COLREF'), . . .)	assigns a color name to color index
fill area	GSET('FILCOLOR', . . .)	selects the color of the fill area
	GSET('FILSTYLE', . . .)	selects the pattern when FILTYPE is HATCH or PATTERN
	GSET('FILTYPE', . . .)	specifies the type of interior for the fill area
	GSET('HTML', . . .)	specifies the HTML string to invoke when an affected DSGI graphic element in a web page is clicked
line	GSET('LINCOLOR', . . .)	selects the color of the line
	GSET('LINTYPE', . . .)	sets the type of line

DSGI Operations	Associated Function	Function Description
	GSET('LINWIDTH', . . .)	specifies the width of the line
marker	GSET('MARCOLOR', . . .)	selects the color of the marker
	GSET('MARSIZE', . . .)	determines the size of the marker
	GSET('MARTYPE', . . .)	sets the type of marker drawn
text	GSET('TEXALIGN', . . .)	specifies horizontal and vertical alignment of text
	GSET('TEXCOLOR', . . .)	selects the color of the text
	GSET('TEXFONT', . . .)	sets the font for the text
	GSET('TEXHEIGHT', . . .)	selects the height of the text
	GSET('TEXPATH', . . .)	determines reading direction of text
	GSET('TEXUP', . . .)	selects the angle of text
Setting Attributes That Affect Entire Graph		
	GSET('ASPECT', . . .)	sets the aspect ratio
	GSET('CATALOG', . . .)	selects the catalog to use
	GSET('CBACK', . . .)	selects the background color
	GSET('DEVICE', . . .)	specifies the output device
	GSET('HPOS', . . .)	sets the number of columns in the graphics output area
	GSET('HSIZE', . . .)	sets the width of the graphics output area in units of inches
	GSET('VPOS', . . .)	sets the number of rows in the graphics output area

DSGI Operations	Associated Function	Function Description
	GSET('VSIZE', . . .)	sets the height of the graphics output area in units of inches
Managing Catalogs		
	GRAPH('COPY', . . .)	copies a graph to another entry within the same catalog
	GRAPH('DELETE', . . .)	deletes a graph
	GRAPH('INSERT', . . .)	inserts a previously created graph into the currently open segment
	GRAPH('RENAME', . . .)	renames a graph
Drawing Graphics Elements		
arc	GDRAW("ARC', . . .)	draws a circular arc
bar	GDRAW(BAR', . . .)	draws a rectangle that can be filled
ellipse	GDRAW('ELLIPSE', . . .)	draws an oblong circle that can be filled
elliptical arc	GDRAW('ELLARC', . . .)	draws an elliptical arc
fill area	GDRAW('FILL', . . .)	draws a polygon that can be filled
line	GDRAW('LINE', . . .)	draws a single line, a series of connected lines, or a dot
marker	GDRAW('MARK', . . .)	draws one or more symbols
pie	GDRAW('PIE', . . .)	draws a pie slice that can be filled
text	GDRAW('TEXT', . . .)	draws a character string
Initializing DSGI		
	GINIT( )	initializes DSGI
	GRAPH('CLEAR', . . .)	opens a segment to receive graphics primitives
Handling Messages		

DSGI Operations	Associated Function	Function Description
	GDRAW('MESSAGE', . . .)	prints a message in the SAS log
	GPRINT(*code*)	prints the description of a DSGI error code
	GSET('MESSAGE', . . .)	turns message logging on or off
Ending DSGI		
	GRAPH('UPDATE', . . .)	closes the currently open segment and, optionally, displays it
	GTERM()	ends DSGI
Activating Transformations		
	GET('TRANSNO', . . .)	selects the transformation number of the viewport or window to use
Defining Viewports		
	GSET('CLIP', . . .)	turns clipping on or off
	GSET('VIEWPORT', . . .)	sets the coordinates of the viewport and assigns it a transformation number
Defining Windows		
	GSET('WINDOW', . . .)	sets the coordinates of the window and assigns it a transformation number

DSGI Routines

DSGI routines return the values set by some of the DSGI functions. Table 31.2 on page 781 summarizes the types of values that the GASK routines can check. Refer to Chapter 32, "DATA Step Graphics Interface Dictionary," on page 813 for details about each routine.

Table 31.2 DATA Step Graphics Interface Routines

DSGI Operations	Associated Routine	Routine Description
Checking Attribute Bundles (valid values for *xxx* are FIL, LIN, MAR, and TEX)		
	GASK('ASK', . . .)	returns the aspect source flag of the attribute
	GASK('*xxx*INDEX', . . .)	returns the index of the active bundle
	GASK('*xxx*REP', . . .)	returns the attributes assigned to the bundle
Checking Attribute Settings		
color index	GASK('COLINDEX', . . .)	returns the color indices that currently have colors assigned to them
	GASK('COLREP', . . .)	returns the color name assigned to the color index
fill area	GASK('FILCOLOR', . . .)	returns the color of the fill area
	GASK('FILSTYLE', . . .)	returns the index of the pattern when the FILTYPE is HATCH or PATTERN
	GASK('FILTYPE', . . .)	returns the index of the type of interior
	GASK('HTML', . . .)	finds the HTML string that is in effect when one of the following graphic elements is drawn: bar, ellipse, fill, mark, pie, and text.
line	GASK('LINCOLOR', . . .)	returns the color index of the color of the line
	GASK('LINTYPE', . . .)	returns the index of the type of line
	GASK('LINWIDTH', . . .)	returns the width of the line
marker	GASK('MARCOLOR', . . .)	returns the color index of the color of markers
	GASK('MARSIZE', . . .)	returns the size of markers
	GASK('MARTYPE', . . .)	returns the index of the type of marker drawn
text	GASK('TEXALIGN', . . .)	returns the horizontal and vertical alignment of text
	GASK('TEXCOLOR', . . .)	returns the color index of the color of text
	GASK('TEXEXTENT', . . .)	returns the coordinates of text extent rectangle and the text concatenation point of the character string
	GASK('TEXFONT', . . .)	returns the text font

DSGI Operations	Associated Routine	Routine Description
	GASK('TEXHEIGHT', . . .)	returns the height of text
	GASK('TEXPATH', . . .)	returns the reading direction of text
	GASK('TEXUP', . . .)	returns the character up vector in *x* vector and *y* vector
Checking Attributes That Affect Entire Graph		
	GASK('ASPECT', . . .)	returns the aspect ratio
	GASK('CATALOG', . . .)	returns the current catalog
	GASK('CBACK', . . .)	returns the background color
	GASK('DEVICE', . . .)	returns the current output device
	GASK('HPOS', . . .)	returns the number of columns in the graphics output area
	GASK('HSIZE', . . .)	returns the width of the graphics output area in units of inches
	GASK('MAXDISP', . . .)	returns the dimensions of maximum display area for the device in meters and pixels
	GASK('VPOS', . . .)	returns the number of rows in the graphics output area
	GASK('VSIZE', . . .)	returns the height of the graphics output area in units of inches
Querying Catalogs		
	GASK('GRAPHLIST', . . .)	returns the names of graphs in the current catalog
	GASK('NUMGRAPH', . . .)	returns the number of graphs in the current catalog
	GASK('OPENGRAPH', . . .)	returns the name of the currently open graph
Checking System Status		
	GASK('STATE', . . .)	returns the current operating state
	GASK('WSACTIVE', . . .)	returns whether or not the workstation is active
	GASK('WSOPEN', . . .)	returns whether or not the workstation is open
Checking Transformation Definitions		
	GASK('TRANS', . . .)	returns the coordinates of the viewport and window associated with the transformation
	GASK('TRANSNO', . . .)	returns the active transformation number
Checking Viewport Definitions		

DSGI Operations	Associated Routine	Routine Description
	GASK('CLIP', . . .)	returns the status of clipping
	GASK('VIEWPORT', . . .)	returns the coordinates of the viewport assigned to the transformation number
Checking Window Definitions		
	GASK('WINDOW', . . .)	returns the coordinates of the window assigned to the transformation number

Creating Simple Graphics with DSGI

Within any DSGI program, you need to follow these basic steps:

1. Initialize DSGI.

   The function that initializes DSGI is GINIT(). GINIT() loads the graphics sublibrary, opens a workstation, and activates a workstation.

2. Open a graphics segment.

   Before you can submit graphics primitives, you must submit the GRAPH("CLEAR", . . .) function. GRAPH("CLEAR", . . .) opens a graphic segment, to allow graphics primitives to be submitted.

3. Generate graphics elements.

   DSGI can generate arcs, bars, ellipses, elliptical arcs, lines, markers, pie slices, polygons (fill areas), and text. These graphics elements are all produced with the GDRAW function using their associated operator names.

   GDRAW functions can be submitted only when a graphics segment is open. They must be submitted between the GRAPH("CLEAR", . . .) and GRAPH("UPDATE", . . .) functions.

4. Close the graphics segment.

   Once the attribute and graphics statements have been entered, you must submit statements to close the graphics segment and output the graph. The GRAPH("UPDATE", . . .) function closes the graphic segment currently open and, can display the graphics output.

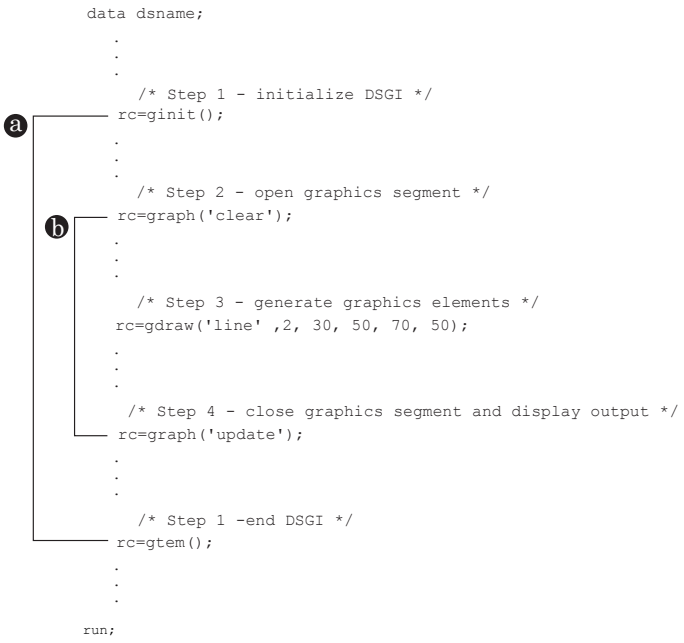
5. End DSGI.

   The GTERM() function ends DSGI by deactivating and closing the workstation, and closing the graphics sublibrary. It frees any memory allocated by DSGI.

   Note: You must execute a RUN statement at the end of the DATA step to display the output.

Figure 31.3 on page 784 outlines the basic steps and shows the functions used to initiate steps 1, 2, 4, and 5. Step 3 can consist of many types of functions. The GDRAW("LINE", . . .)function is used as an example.

Figure 31.3 Basic Steps Used in Creating DSGI Graphics Output



Notice that there are two pairs of functions that work together within a DSGI DATA step (shown by a and b in Figure 31.3 on page 784). The first pair, GINIT() and GTERM(), begin and end DSGI. Within the first pair, the second pair, GRAPH("CLEAR", . . .)and GRAPH("UPDATE", . . .)begin and end a graphics segment. You can repeat these pairs within a single DATA step to produce multiple graphics output; however, the relative positions of these functions must be maintained within a DATA step. See "Generating Multiple Graphics Output in One DATA Step" on page 795 for more information about producing multiple graphics outputs from one DATA step.

The order of these steps is controlled by DSGI operating states. Before any DSGI function or routine can be submitted, the operating state in which that function or routine can be submitted must be active. See "How Operating States Control the Order of DSGI Statements" on page 785.

Setting Attributes for Graphics Elements

The appearance of the graphics elements is determined by the settings of the attributes. Attributes control such aspects as height of text; text font; and color, size, and width of the graphics element. In addition, the HTML attribute determines if the element provides a link to another graphic or web page. Attributes are set and reset with GSET functions. GASK routines return the current setting of the attribute specified.

Each graphics primitive is associated with a particular set of attributes. Its appearance or linking capability can be altered only by that set of attributes. Table 31.3 on page 785 lists the operators used with GDRAW functions to generate graphics elements and the attributes that control them.

Table 31.3 Graphics Output Primitive Functions and Associated Attributes

Graphics Output Primitive	Functions	Associated Attributes
Arc	GDRAW('ARC', . . .)	HTML, LINCOLOR, LININDEX, LINREP, LINTYPE, LINWIDTH
Bar	GDRAW('BAR', . . .)	FILCOLOR, FILINDEX, FILREP, FILSTYLE, FILTYPE, HTML
Ellipse	GDRAW('ELLIPSE', . . .)	FILCOLOR, FILINDEX, FILREP, FILSTYLE, FILTYPE, HTML
Elliptical Arc	GDRAW('ELLARC', . . .)	HTML, LINCOLOR, LININDEX, LINREP, LINTYPE, LINWIDTH
Fill Area	GDRAW('FILL', . . .)	FILCOLOR, FILINDEX, FILREP, FILSTYLE, FILTYPE, HTML
Line	GDRAW('LINE', . . .)	HTML, LINCOLOR, LININDEX, LINREP, LINTYPE, LINWIDTH
Marker	GDRAW('MARK', . . .)	HTML, MARCOLOR, MARINDEX, MARREP, MARSIZE, MARTYPE
Pie	GDRAW('PIE', . . .)	FILCOLOR, FILINDEX, FILREP, FILSTYLE, FILTYPE, HTML
Text	GDRAW('TEXT', . . .)	HTML, TEXALIGN, TEXCOLOR, TEXFONT, TEXHEIGHT, TEXINDEX, TEXPATH, TEXREP, TEXUP

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

If you do not set an attribute before you submit a graphics primitive, DSGI uses the default value for the attribute. Refer to Chapter 32, "DATA Step Graphics Interface Dictionary," on page 813 for the default values used for each attribute.

How Operating States Control the Order of DSGI Statements

Each DSGI function and routine can be submitted only when certain operating states are active. This restriction affects the order of functions, and routines within the DATA step. Generally, the operating states within a DATA step follow this order:

GKCL ► WSAC ► SGOP ► WSAC ► GKCL

Functions That Change the Operating State

The functions described earlier in steps 1, 2, 4, and 5 actually control the changes to the operating state. For example, the GINIT() function must be submitted when the operating state is GKCL, the initial state of DSGI. GINIT() then changes the operating state to WSAC. The GRAPH("CLEAR", . . .)function must be submitted when the operating state is WSAC and before any graphics primitives are submitted. The reason it precedes graphics primitives is that it changes the operating state to SGOP, the operating state in which you can submit graphics primitives. The following list shows the change in the operating state due to specific functions:

GINIT()	GKCL ▶ WSAC
GRAPH('CLEAR', . . .)	WSAC ▶ SGOP
GRAPH('UPDATE', . . .)	SGOP ▶ WSAC
GTERM()	WSAC ▶ GKCL

Because these functions change the operating state, you must order all other functions and routines so that the change in operating state is appropriate for the functions and routines that follow. The following program statements show how the operating state changes from step to step in a typical DSGI program. They also summarize the functions and routines that can be submitted under each operating state. The functions that change the operating state are included as actual statements. Refer to "Operating States" on page 814 for the operating states from which functions and routines can be submitted.

```
data dsname;

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

      /* Step 1 - initialize DSGI            */
   rc=ginit();

      /* WSAC - workstation is active; can execute:            */
      /*   1. most GASK routines                               */
      /*   2. some catalog management functions                */
      /*      (some GRAPH functions)                           */
      /*   3. GSET functions that set attributes               */
      /*      and bundles, viewports, windows,                 */
      /*      transformations, and message logging             */

      /* Step 2 - open a graphics segment     */
   rc=graph("clear", "text");

      /* SGOP - segment open; can execute:                     */
      /* 1. any GASK routine                                   */
      /* 2. any GDRAW function                                 */
      /* 3. some catalog management functions                  */
      /*    (some GRAPH functions)                             */
```

```
      /* 4. GSET functions that set attributes                */
      /*    and bundles, viewports, windows,                  */
      /*    transformations, and message logging               */

      /* Step 3 - execute graphics primitives */
   rc = gdraw("line", 2, 30,50,50,50);

      /* Step 4 - close the graphics segment  */
   rc=graph("update");

      /* WSAC - workstation is active; can execute:           */
      /*  1. most GASK routines                               */
      /*  2. some catalog management functions                */
      /*     (some GRAPH functions)                           */
      /*  3. GSET functions that set attributes               */
      /*     and bundles, viewports, windows,                 */
      /*     transformations, and message logging             */

      /* Step 5 - end DSGI */
   rc=gterm();

      /* GKCL - initial state of DSGI         */
run;
```

Order of Functions and Routines

Functions and routines within each operating state can technically be submitted in any order; however, once an attribute is set, it remains in effect until the end of the DATA step or until you change its value. If you are producing multiple graphics output within the same DATA step, the attributes for one output affect the ones that follow. Attributes are not reset until after the GTERM() function is submitted.

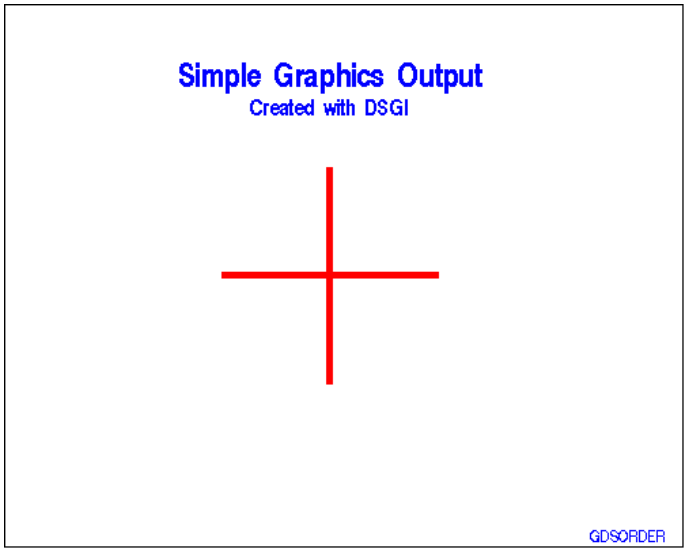
Notice that you can set attributes for the graphics primitives in several places. As long as the functions that set the attributes are executed before the graphics primitives, they affect the graphics output. If you execute them after a graphics primitive, the primitive is not affected. See “Setting Attributes for Graphics Elements” on page 784.

The following program statements illustrate a more complex DSGI program that produces Display 31.1 on page 788 when submitted. Notice that all attributes for a graphics primitive are executed before the graphics primitive. In addition, the GINIT() and GTERM() pairing and the GRAPH(“CLEAR”) and GRAPH(“UPDATE”) pairing are maintained within the DATA step. Refer to “Operating States” on page 814 for the operating states in which each function and routine can be submitted.

```
/* set the graphics environment */
goptions reset=global gunit=pct border
         hsize=7 in vsize=5 in
         targetdevice=pscolor;

   /* execute a DATA step with DSGI */
data dsname;
      /* initialize SAS/GRAPH software */
      /* to accept DSGI statements     */
   rc=ginit();
   rc=graph("clear");

      /* assign colors to color index */
```

```
   rc=gset("colrep", 1, "blue");
   rc=gset("colrep", 2, "red");

      /* define and display titles */
   rc=gset("texcolor", 1);
   rc=gset("texfont", "swissb");
   rc=gset("texheight", 6);
   rc=gdraw("text", 45, 93, "Simple Graphics Output");

      /* change the height and */
      /* display second title  */
   rc=gset("texheight", 4);
   rc=gdraw("text", 58, 85, "Created with DSGI");

      /* define and display footnotes */
      /* using same text font and     */
      /* color as defined for titles  */
   rc=gset("texheight", 3);
   rc=gdraw("text", 125, 1, "GDSORDER ");

      /* define and draw bar */
   rc=gset("lincolor", 2);
   rc=gset("linwidth", 5);
   rc=gdraw("line", 2, 72, 72, 30, 70);
   rc=gdraw("line", 2, 52, 92, 50, 50);

      /* display graph and end DSGI */
   rc=graph("update");
   rc=gterm();
run;
```

Display 31.1 Simple Graphics Output Generated with DSGI



Bundling Attributes

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

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

Attributes That Can Be Bundled for Each Graphics Primitive

Each graphics primitive has a group of attributes associated with it that can be bundled. Only the attributes in that group can be assigned to the bundle. Table 31.4 on page 789 shows the attributes that can be bundled for each graphics primitive.

Note: You do not have to use attribute bundles for all graphics primitives if you use a bundle for one. You can define bundles for some graphics primitives and set the attributes individually for others. △

However, if the other graphics primitives are associated with the same attributes you have bundled and you do not want to use the same values, you can use other bundles to set the attributes, or you can set the attributes back to "INDIVIDUAL".

Table 31.4 Attributes That Can Be Bundled for Each Graphics Primitive

Graphics Output Primitive	Associated Attributes That Can Be Bundled
GDRAW('ARC', . . .)	LINCOLOR, LINTYPE, LINWIDTH
GDRAW('BAR', . . .)	FILCOLOR, FILSTYLE, FILTYPE
GDRAW('ELLARC', . . .)	LINCOLOR, LINTYPE, LINWIDTH
GDRAW('ELLIPSE', . . .)	FILCOLOR, FILSTYLE, FILTYPE
GDRAW('FILL', . . .)	FILCOLOR, FILSTYLE, FILTYPE
GDRAW('LINE', . . .)	LINCOLOR, LINTYPE, LINWIDTH
GDRAW('MARK', . . .)	MARCOLOR, MARSIZE, MARTYPE
GDRAW('PIE', . . .)	FILCOLOR, FILSTYLE, FILTYPE
GDRAW('TEXT', . . .)	TEXCOLOR, TEXFONT

Assigning Attributes to a Bundle

To assign values of attributes to a bundle, you must

- assign the values to a numeric bundle index with the GSET("*xxx* REP", . . .)function. Each set of attributes that can be bundled uses a separate GSET("*xxx* REP", . . .)function, where *xxx* is the appropriate prefix for the set of attributes to be bundled. Valid values for *xxx* are FIL, LIN, MAR, and TEX.
- set the aspect source flag (ASF) of the attributes to "BUNDLED" before you use the bundled attributes. You can use the GSET("ASF", . . .)function to set the ASF of an attribute. You need to execute a GSET("ASF", . . .)function for each attribute in the bundle.

The following example assigns the text attributes, color, and font, to the bundle indexed by the number 1. As shown in the GSET("TEXREP", . . .)function, the color

for the bundle is green, the second color in the COLOR= graphics option. The font for the bundle is the "ZAPF" font. (See "COLREP" on page 876 for an explanation of how colors are used in DSGI.)

```
goptions colors=(red green blue);

data dsname;
.
.     /* other DATA step statements */
.
      /* associate the bundle with the index 1 */
   rc=gset("texrep", 1, 2, "zapf");
.
.     /* more statements */
.
      /* assign the text attributes to a bundle */
   rc=gset("asf", "texcolor", "bundled");
   rc=gset("asf", "texfont", "bundled");

      /* draw the text */
   rc=gdraw("text", 50, 50, "Today is the day.");
```

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

1 the current value of the attribute

2 the default value of the attribute.

Selecting a Bundle

Once you have issued the GSET("ASF", . . .)and GSET("*xxx* REP", . . .)functions, you can issue the GSET("*xxx* INDEX", . . .)function to select the bundle. The following statement selects the bundle defined in the previous example:

```
/* invoke the bundle of text attributes */
  rc=gset("texindex", 1);
```

The 1 in this example corresponds to the index number specified in the GSET("TEXREP", . . .)function.

Defining Multiple Bundles for a Graphics Primitive

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

```
/* define another attribute bundle for text */
  rc=gset("texrep", 2, 3, "swiss");
```

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

Note: When using a new bundle, you do not need to reissue the GSET("ASF", . . .) functions for the bundled attributes. Once the ASF of an attribute has been set, the setting remains in effect until it is changed. △

How DSGI Selects the Value of an Attribute to Use

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

```
goptions colors=(red green blue);
rc=gset("asf", "lincolor", "bundled");
rc=gset("asf", "linwidth", "bundled");
rc=gset("asf", "lintype", "bundled");
rc=gset("linrep", 3, 2, 5, 1);
```

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

```
/* activate the bundle */
rc=gset("linindex", 3);

  /* select other attributes for the line */
rc=gset("lincolor", 3);
rc=gset("linwidth", 10);
rc=gset("lintype", 4);

  /* draw a line from point (30,50) to (70,50) */
rc=gdraw("line", 2, 30, 70, 50, 50);
```

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

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

1. Get the index of the active line bundle.
2. Check the ASF of the LINCOLOR attribute. If the ASF is "INDIVIDUAL", the value selected with GSET("LINCOLOR", . . .) is used; otherwise, the LINCOLOR associated with the bundle index is used.
3. Check the ASF of the LINTYPE attribute. If the ASF is "INDIVIDUAL", the value selected with GSET("LINTYPE", . . .) is used; otherwise, the LINTYPE associated with the bundle index is used.
4. Check the ASF of the LINWIDTH attribute. If the ASF is "INDIVIDUAL", the value selected with GSET("LINWIDTH", . . .) is used; otherwise, the LINWIDTH associated with the bundle index is used.
5. Draw the line using the appropriate color, type, and width for the line.

Disassociating an Attribute from a Bundle

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

```
/* disassociate an attribute from a bundle */
  rc=gset("asf", "texcolor", "individual");
  rc=gset("asf", "texfont", "individual");
```

Using Viewports and Windows

In DSGI, you can define viewports and windows. Viewports enable you to subdivide the graphics output area and insert existing graphs or draw graphics elements in

smaller sections of the graphics output area. Windows define the coordinate system within a viewport and enable you to scale the graph or graphics elements drawn within the viewport.

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

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

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

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

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

You generally follow these steps when defining viewports or windows:

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

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

Defining Viewports

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

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

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

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

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

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

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

Clipping around Viewports

When you use viewports, you also might need to use the clipping feature. Even though you have defined the dimensions of your viewport, it is possible for graphics elements to display past its boundaries. If the graphics elements are too large to fit into the dimensions you have defined, portions of the graphics elements actually display outside of the viewport. To ensure that only the portions of the graphics elements that fit within the dimensions of the viewport display, turn the clipping feature on by using the GSET("CLIP", . . .)function. For details, see "CLIP" on page 875.

Defining Windows

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

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

```
/* define the window for viewport 1 */
rc=gset("window", 1, 0, 50, 20, 100);

  /* define the window for viewport 2 */
rc=gset("window", 2, 0, 40, 20, 90);

  /* define the window for viewport 3 */
rc=gset("window", 3, 10, 25, 45, 100);

  /* define the window for viewport 4 */
rc=gset("window", 4, 0, 0, 100, 100);
```

See "Scaling Graphs by Using Windows" on page 804 for an example of using windows to scale graphs.

Note: When you define a window for a viewport, the transformation numbers in the GSET("VIEWPORT", . . .)and GSET("WINDOW", . . .)functions must match in order for DSGI to activate them simultaneously. △

Activating Transformations

Once you have defined a viewport or window, you must activate the transformation in order for DSGI to use the viewport or window. To activate the transformation, use the GSET("TRANSNO",*n*, . . .)function where *n* has the same value as *n* in GSET("VIEWPORT",*n*, . . .)or GSET("WINDOW",*n*, . . .).

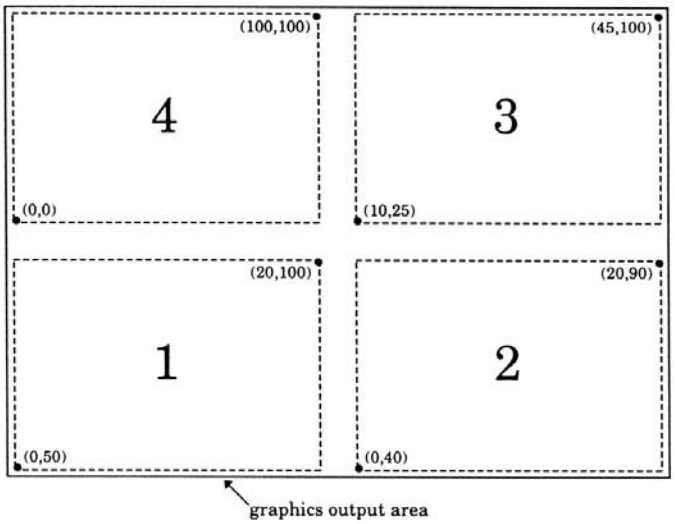
The following program statements illustrate how to activate the viewports and windows defined in the previous examples:

```
/* define the viewports */
  .
  .
  .
  /* define the windows */
  .
  .
  .
  /* activate the first transformation */
```

```
rc=gset("transno", 1);
.
. /* graphics primitive functions follow */
.
  /* activate the second transformation */
rc=gset("transno", 2);
.
. /* graphics primitive functions follow */
.
  /* activate the third transformation */
rc=gset("transno", 3);
.
. /* graphics primitive functions follow */
.
  /* activate the fourth transformation */
rc=gset("transno", 4);
.
. /* graphics primitive functions follow */
.
```

When you activate these transformations, your display is logically divided into four subareas as shown in Figure 31.4 on page 794.

Figure 31.4 Graphics Output Area Divided into Four Logical Transformations



If you want to return to the default viewport and window, execute the GSET(“TRANSNO”, 0) function.

Inserting Existing Graphs into DSGI Graphics Output

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

1 Use the GSET(“CATALOG”, . . .)function to set the output catalog to the catalog that contains the existing graph.

Note: Unless you are using the WORK library, you must have previously defined the libref in a LIBNAME statement or window when using GSET("CATALOG", . . .). △

2 Define a viewport with the dimensions and position of the place in the graphics output where you want to insert the existing graph. GSET("VIEWPORT",*n*, . . .) defines a viewport and GSET("WINDOW",*n*, . . .)defines a window.

3 Define a window as (0,0) to (100,100) so that the inserted graph is not distorted. The graph must have a square area defined to avoid the distortion. If your device does not have a square graphics output area, the window defaults to the units of the device rather than (0,0) to (100,100) and might distort the graph.

4 Activate the transformation number *n*, as defined in the viewport function, and possibly in the window function, using GSET("TRANSNO", *n*, . . .).

5 Use the GRAPH("INSERT", . . .)function with the name of the existing graph.

The following program statements provide an example of including an existing graph in the graphics output being created. The name of the existing graph is "MAP". "LOCAL" points to the library containing the catalog "MAPCTLG". The coordinates of the viewport are percentages of the graphics output area. **SAS-data-library** refers to a permanent SAS data library.

Example Code 31.1 Graphics Output Area Divided into Four Logical Transformations

```
libname local "SAS-data-library";
.
.
.
   /* select the output catalog to the */
   /* catalog that contains "map" */
  rc=gset("catalog", "local", "mapctlg");
  .
  .
  .

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

   /* set the transformation number to the one */
   /* defined in the viewport function */
  rc=gset("transno", 1);

   /* insert the existing graph */
  rc=graph("insert", "map");
```

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

Generating Multiple Graphics Output in One DATA Step

You can produce more than one graphics output within the same DATA step. All statements between the GRAPH("CLEAR", . . .)and GRAPH("UPDATE", . . .)functions produce one graphics output.

Each time the GRAPH(“UPDATE”, . . .)function is executed, a graph is displayed. After the GTERM() function is executed, no more graphs are displayed for the DATA step. The GINIT() function must be executed again to produce more graphs.

CAUTION:

Be careful using global SAS/GRAPH statements when you are producing multiple output from within the DATA step. △

If you use global SAS/GRAPH statements when producing multiple output from one DATA step, the last definition of the statements is used for all displays.

Processing DSGI Statements in Loops

You can process DSGI statements in loops to draw a graphics element multiple times in one graphics output or to produce multiple output. If you use loops, you must maintain the GRAPH(“CLEAR”, . . .)and GRAPH(“UPDATE”, . . .)pairing within the GINIT() and GTERM() pairing. (See Figure 31.3 on page 784.) The following program statements illustrate how you can use DSGI statements to produce multiple graphics output for different output devices:

```
data _null_;
  length d1-d5 $ 8;
  input d1-d5;
  array devices{5} d1-d5;
  .
  .
  .
  do j=1 to 5;
   rc=gset("device", devices{j});
   .
   .
   .
   rc=ginit();
   .
   .
   .
   do i=1 to 5;
     rc=graph("clear");
     rc=gset("filcolor", i);
     rc=gdraw("bar", 45, 45, 65, 65);
     rc=graph("update");
   end;
   .
   .
   .
   rc=gterm();
  end;
  cards;
tek4105 hp7475 ps qms800 ibm3279
;
run;
```

The inner loop produces five graphs for each device. Each graphics output produced by the inner loop consists of a bar. The bar uses a different color for each graph. The outer loop produces all of the graphs for five different devices. A total of 25 graphs is generated by these loops.

Examples

The following examples show different applications for DSGI and illustrate some of its features such as defining viewports and windows, inserting existing graphs, angling text, using GASK routines, enlarging a segment of a graph, and scaling a graph.

These examples use some additional graphics options that cannot be used in other examples in this book. Because the dimensions of the default window vary across devices, the TARGETDEVICE=, HSIZE=, and VSIZE= graphics options are used to make the programs more portable. The COLORS= graphics option provides a standard color list.

Refer to Chapter 32, "DATA Step Graphics Interface Dictionary," on page 813 for a complete description of each of the functions used in the examples.

Vertically Angling Text

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

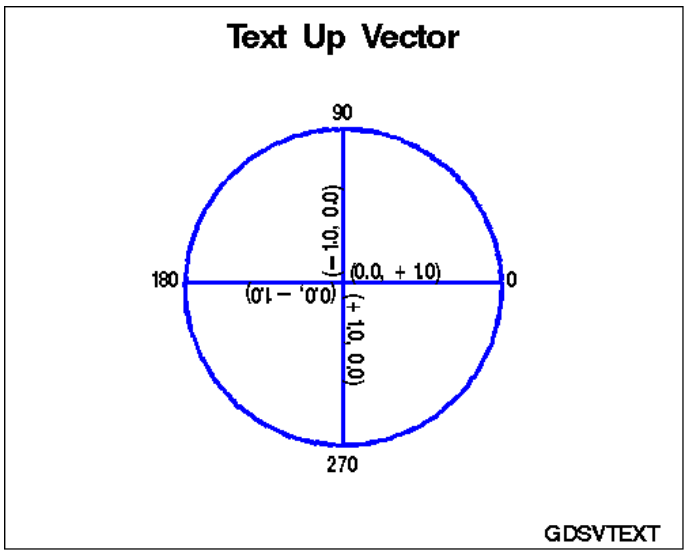
This example illustrates how global statements can be used with DSGI. In this example, FOOTNOTE and TITLE statements create the footnotes and title for the graph. The GOPTIONS statement defines general aspects of the graph. The COLORS= graphics option provides a color list from which the color referenced in GSET("*xxx* COLOR", . . .)functions are selected.

The following program statements produce Display 31.2 on page 799:

```
/* set the graphics environment */
goptions reset=global gunit=pct border
     ftext=swissb htitle=6 htext=3
     colors=(black blue green red)
     hsize=7 in vsize=5 in
     targetdevice=pscolor;

  /* define the footnote and title */
footnote1 j=r "GDSVTEXT ";
title1 "Text Up Vector";

  /* execute DATA step with DSGI */
data vector;

   /* prepare SAS/GRAPH software */
   /* to accept DSGI statements */
  rc=ginit();
  rc=graph("clear");

   /* define and display arc */
   /* with intersecting lines */
  rc=gset("lincolor", 2);
  rc=gset("linwidth", 5);
  rc=gdraw("arc", 84, 50, 35, 0, 360);
  rc=gdraw("line", 2, 49, 119, 51, 51);
  rc=gdraw("line", 2, 84, 84, 15, 85);

   /* define height of text */
```

```
  rc=gset("texheight", 5);

   /* mark 360 degrees on the arc */
   /* using default align     */
  rc=gdraw("text", 121, 50, "0");

   /* set text to align to the right and */
   /* mark 180 degrees on the arc    */
  rc=gset("texalign", "right", "normal");
  rc=gdraw("text", 47, 50, "180");

   /* set text to align to the center and */
   /* mark 90 and 270 degrees on the arc */
  rc=gset("texalign", "center", "normal");
  rc=gdraw("text", 84, 87, "90");
  rc=gdraw("text", 84, 9, "270");

   /* reset texalign to normal and     */
   /* display coordinate values or quadrant */
  rc=gset("texalign", "normal", "normal");
  rc=gdraw("text", 85, 52, "(0.0, +1.0)");

   /* rotate text using TEXUP and      */
   /* display coordinate values or quadrant */
  rc=gset("texup", 1.0, 0.0);
  rc=gdraw("text", 85, 49, "(+1.0, 0.0)");

   /* rotate text using TEXUP and      */
   /* display coordinate values or quadrant */
  rc=gset("texup", 0.0, -1.0);
  rc=gdraw("text", 83, 50, "(0.0, -1.0)");

   /* rotate text using TEXUP and      */
   /* display coordinate values or quadrant */
  rc=gset("texup", -1.0, 0.0);
  rc=gdraw("text", 83, 52, "(-1.0, 0.0)");

   /* display graph and end DSGI */
  rc=graph("update");
  rc=gterm();
run;
```

Display 31.2 Text Angled with the GSET("TEXUP", ...) Function



This example illustrates the following features:

- The COLORS= graphics option provides a color table to be used with the GSET("LINCOLOR", . . .)function.
- The HSIZE= graphics option provides a standard width for the graphics output area.
- The VSIZE= graphics option provides a standard height for the graphics output area.
- The TARGETDEVICE= graphics option selects the standard color PostScript driver to use as the target device.
- The GINIT() function begins DSGI.
- The GRAPH("CLEAR") function sets the graphics environment. Because the function does not specify a name for the catalog entry, "DSGI" is the default name.
- The GSET("TEXHEIGHT", . . .), GSET("LINCOLOR", . . .), and GSET("LINWIDTH", . . .)functions set attributes of the graphics primitives. The COLORS= graphics option provides a color table for the GSET("LINCOLOR", 2) function to reference. In this example, the color indexed by 2 is used to draw lines. Since no other color table is explicitly defined with GSET("COLREP", . . .) functions, DSGI looks at the color list and chooses the color indexed by 2 (the second color in the list) to draw the lines.
- The GDRAW("ARC", . . .)function draws an empty pie chart. The arguments of the GDRAW("ARC", . . .)function provide the coordinates of the starting point, the radius, and the beginning and ending angles of the arc.
- The GDRAW("LINE", . . .)function draws a line. It provides the type of line, the coordinates of the beginning point, and the coordinates of the ending point.
- The GDRAW("TEXT", . . .)function draws the text. It sets the coordinates of the starting point of the text string as well as the text string to be written.
- The GSET("TEXALIGN", . . .)function aligns text to the center, left, or right of the starting point specified in the GDRAW("TEXT", . . .)function.
- The GSET("TEXUP", . . .)function determines the angle at which the text is to be written.

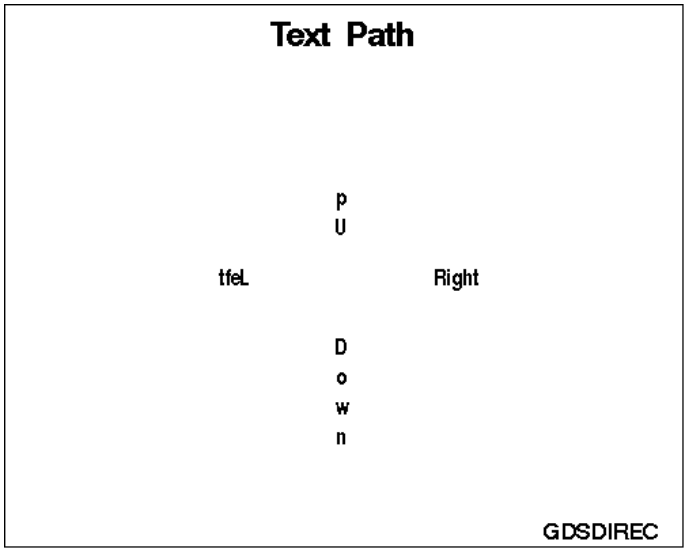
- The GRAPH("UPDATE", . . .)function closes the graphics segment.
- The GTERM() function ends DSGI.

Changing the Reading Direction of the Text

This example changes the reading direction of text. Notice that the data set name is _NULL_. No data set is created as a result of this DATA step; however, the graphics output is generated. The following program statements produce Display 31.3 on page 801:

```
  /* set the graphics environment */
goptions reset=global gunit=pct border
     ftext=swissb htitle=6 htext=3
     colors=(black blue green red)
     hsize=7 in vsize=5 in
     targetdevice=pscolor;

  /* define the footnote and title */
footnote1 j=r "GDSDIREC ";
title1 "Text Path";

  /* execute DATA step with DSGI */
data _null_;

   /* prepare SAS/GRAPH software */
   /* to accept DSGI statements */
  rc=ginit();
  rc=graph("clear");

   /* define height of text */
  rc=gset("texheight", 5);

   /* display first text */
  rc=gdraw("text", 105, 50, "Right");

   /* change text path so that text reads from */
   /* right to left and display next text   */
  rc=gset("texpath", "left");
  rc=gdraw("text", 65, 50, "Left");

   /* change text path so that text reads up */
   /* the display and display next text   */
  rc=gset("texpath", "up");
  rc=gdraw("text", 85, 60, "Up");

   /* change text path so that text reads down */
   /* the display and display next text    */
  rc=gset("texpath", "down");
  rc=gdraw("text", 85, 40, "Down");

   /* display the graph and end DSGI */
  rc=graph("update");
  rc=gterm();
run;
```

Display 31.3 Reading Direction of the Text Changed with the GSET("TEXPATH",...) Function



Features not explained earlier in "Vertically Angling Text" are described here:

- DATA _NULL_ causes the DATA step to be executed, but no data set is created.
- The GSET("TEXPATH", . . .)function changes the direction in which the text reads.

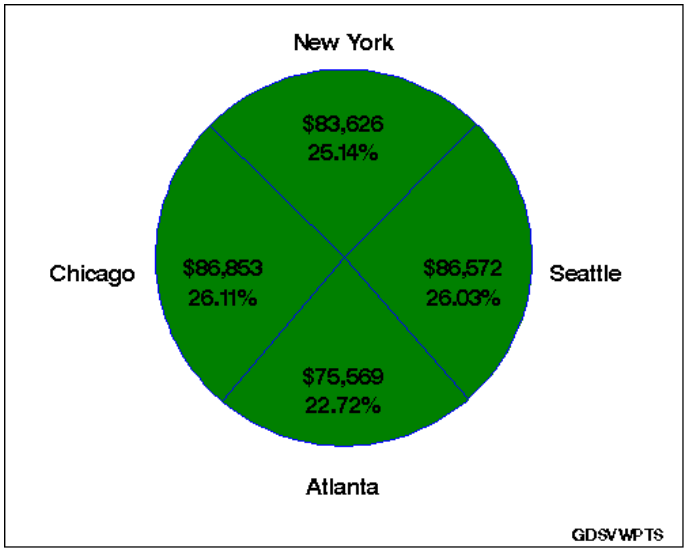
Using Viewports in DSGI

This example uses the GCHART procedure to generate a graph, defines a viewport in which to display it, and inserts the GCHART graph into the graphics output being created by DSGI. Display 31.4 on page 803 shows the pie chart created by the GCHART procedure. Display 31.5 on page 803 shows the same pie chart after it has been inserted into a DSGI graph.

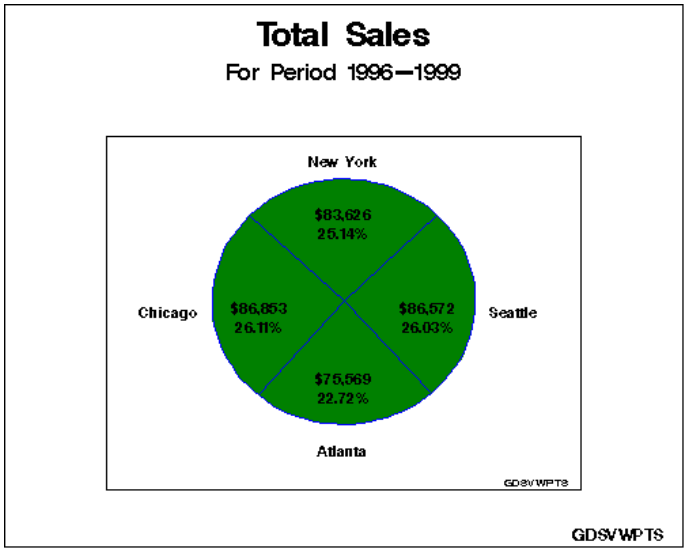
```
  /* set the graphics environment */
goptions reset=global gunit=pct border
     ftext=swissb htitle=6 htext=4
     colors=(black blue green red)
     hsize=7 in vsize=7 in
     targetdevice=pscolor;


  /* create data set TOTALS */
data totals;
  length dept $ 7 site $ 8;
  do year=1996 to 1999;
   do dept="Parts","Repairs","Tools";
     do site="New York","Atlanta","Chicago","Seattle";
      sales=ranuni(97531)*10000+2000;
      output;
     end;
   end;
  end;
```

```
run;

  /* define the footnote */
footnote1 h=3 j=r "GDSVWPTS ";

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

  /* define the titles */
title1 "Total Sales";
title2 "For Period 1996-1999";

  /* execute DATA step with DSGI */
data piein;

   /* prepare SAS/GRAPH software */
   /* to accept DSGI statements */
  rc=ginit();
  rc=graph("clear");

   /* define and activate viewport for inserted graph */
  rc=gset("viewport", 1, .15, .05, .85, .90);
  rc=gset("window", 1, 0, 0, 100, 100);
  rc=gset("transno", 1);

   /* insert graph created from GCHART procedure */
  rc=graph("insert", "GDSVWPTS");

   /* display graph and end DSGI */
  rc=graph("update");
  rc=gterm();
run;
```

Display 31.4 Pie Chart Produced with the GCHART Procedure



Display 31.5 Pie Chart Inserted into DSGI Graph by Using a Viewport



Features not explained in previous examples are described here:

- A graph can be created by another SAS/GRAPH procedure and inserted into DSGI graphics output. In this case, the NAME= option in the PIE statement of the GCHART procedure names the graph, “GDSVWPTS”, to be inserted into the DSGI graphics output.
- The GSET(“VIEWPORT”, . . .)function defines the section of the graphics output area into which GDSVWPTS is inserted. The dimensional ratio of the viewport should match that of the entire graphics output area so that the inserted graph is not distorted.
- The GSET(“WINDOW”, . . .)function defines the coordinate system to be used within the viewport. In this example, the coordinates (0,0) to (100,100) are used. These coordinates provide a square area to insert the graph and preserve the aspect ratio of the GCHART graph.

- The GSET("TRANSNO", . . .)function activates the transformation for the defined viewport and window.
- The GRAPH("INSERT", . . .)function inserts the existing graph, "GDSVWPTS", into the one being created with DSGI. If no viewport has been explicitly defined, DSGI inserts the graph into the default viewport, which is the entire graphics output area.

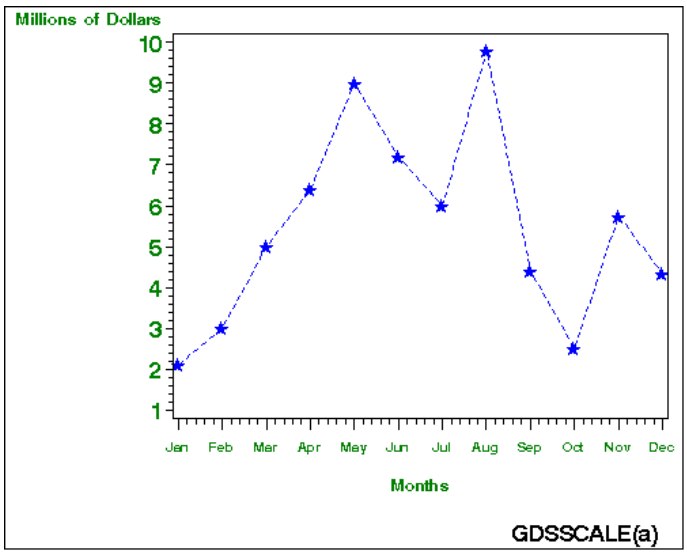
Scaling Graphs by Using Windows

This example uses the GPLOT procedure to generate a plot of AMOUNT*MONTH and store the graph in a permanent catalog. DSGI then scales the graph by defining a window in another DSGI graph and inserting the GPLOT graph into that window. Display 31.6 on page 806 shows the plot as it is displayed with the GPLOT procedure. Display 31.7 on page 806 shows how the same plot is displayed when the *x* axis is scaled from 15 to 95 and the *y* axis is scaled from 15 to 75.

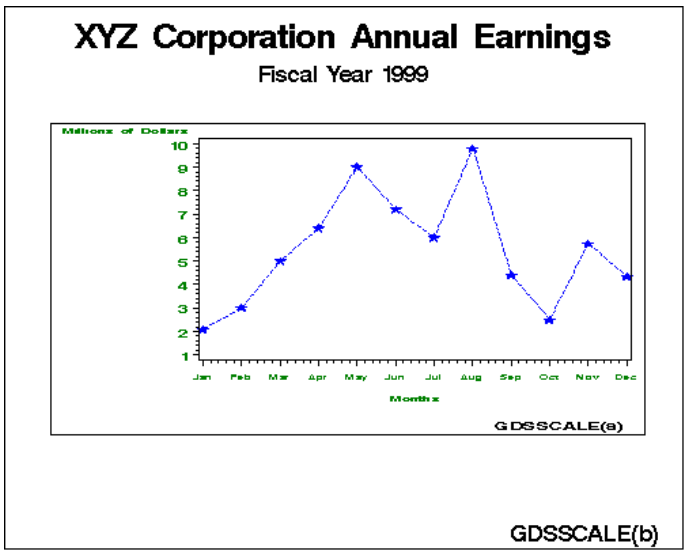
```
  /* set the graphics environment */
goptions reset=global gunit=pct border
     ftext=swissb htitle=6 htext=3
     colors=(black blue green red)
     hsize=7 in vsize=5 in
     targetdevice=pscolor;

  /* create data set EARN, which holds month */
  /* and amount of earnings for that month  */
data earn;
  input month amount;
  datalines;
1 2.1
2 3
3 5
4 6.4
5 9
6 7.2
7 6
8 9.8
9 4.4
10 2.5
11 5.75
12 4.35
;
run;

  /* define the footnote for the first graph */
footnote1 j=r "GDSSCALE(a) ";

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

```
       Tick=10 "Oct" Tick=11 "Nov" Tick=12 "Dec");

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

  /* generate a plot of AMOUNT * MONTH,     */
  /* and store in member GDSSCALE         */
proc gplot data=earn;
  plot amount*month
    / haxis=axis2
     vaxis=axis1
     name="GDSSCALE";
run;

  /* define the footnote and titles for    */
  /* second graph, scales the output */
footnote1 j=r "GDSSCALE(b) ";
title1 "XYZ Corporation Annual Earnings";
title2 h=4 "Fiscal Year 1999";

  /* execute DATA step with DSGI using */
  /* catalog entry created in previous */
  /* plot, but do not create a data set */
  /* (determined by specifying _NULL_) */
data _null_;

   /* prepare SAS/GRAPH software */
   /* to accept DSGI statements */
  rc=ginit();
  rc=graph("clear");

   /* define viewport and window for inserted graph */
  rc=gset("viewport", 1, .20, .30, .90, .75);
  rc=gset("window", 1, 15, 15, 95, 75);
  rc=gset("transno", 1);

   /* insert graph previously created */
  rc=graph("insert", "GDSSCALE");

   /* display graph and end DSGI */
  rc=graph("update");
  rc=gterm();
run;
```

Display 31.6 Plot Produced with the GPLOT Procedure



Display 31.7 Plot Scaled by Using a Window in DSGI



One feature not explained in previous examples is described here:

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

Enlarging an Area of a Graph by Using Windows

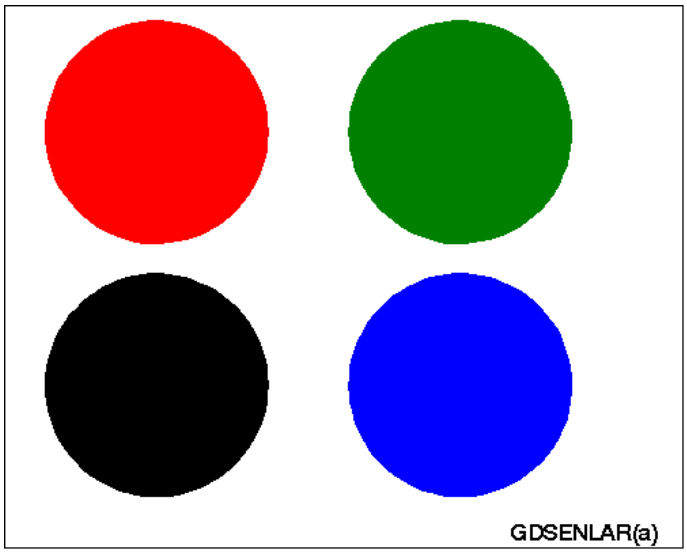
This example illustrates how you can enlarge a section of a graph by using windows. In the first DATA step, the program statements generate graphics output that contains four pie charts. The second DATA step defines a window that enlarges the bottom-left quadrant of the graphics output and inserts "GDSENLAR"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

following program statements produce Display 31.8 on page 808 from the first DATA step, and Display 31.9 on page 809 from the second DATA step:

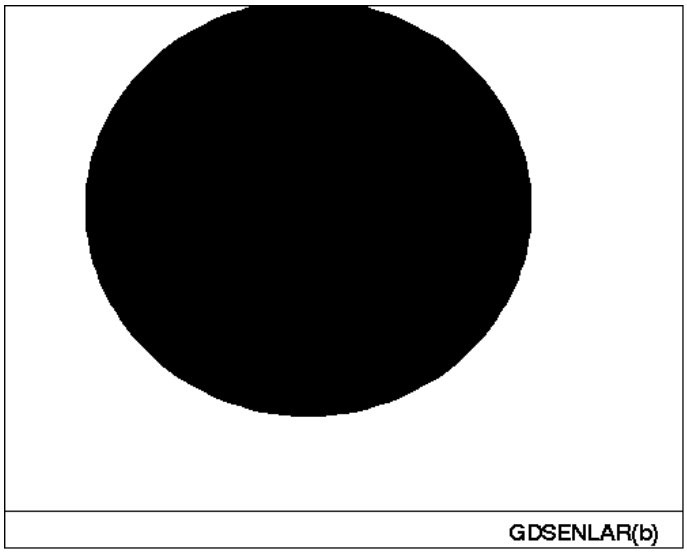
```
  /* set the graphics environment */
goptions reset=global gunit=pct border
     ftext=swissb htext=3
     colors=(black blue green red)
     hsize=7 in vsize=5 in
     targetdevice=pscolor;

  /* define the footnote for the first graph */
footnote1 j=r "GDSENLAR(a) ";

  /* execute DATA step with DSGI */
data plot;

   /* prepare SAS/GRAPH software */
   /* to accept DSGI statements */
  rc=ginit();
  rc=graph("clear", "GDSENLAR");

   /* define and draw first pie chart */
  rc=gset("filcolor", 4);
  rc=gset("filtype", "solid");
  rc=gdraw("pie", 30, 75, 22, 0, 360);

   /* define and draw second pie chart */
  rc=gset("filcolor", 1);
  rc=gset("filtype", "solid");
  rc=gdraw("pie", 30, 25, 22, 0, 360);

   /* define and draw third pie chart */
  rc=gset("filcolor", 3);
  rc=gset("filtype", "solid");
  rc=gdraw("pie", 90, 75, 22, 0, 360);

   /* define and draw fourth pie chart */
  rc=gset("filcolor", 2);
  rc=gset("filtype", "solid");
  rc=gdraw("pie", 90, 25, 22, 0, 360);

   /* display graph and end DSGI */
  rc=graph("update");
  rc=gterm();
run;

  /* define the footnote for the second graph */
footnote1 j=r "GDSENLAR(b) ";

  /* execute DATA step with DSGI  */
  /* that zooms in on a section of */
  /* the previous graph      */
data zoom;

   /* prepare SAS/GRAPH software */
```

```
   /* to accept DSGI statements */
  rc=ginit();
  rc=graph("clear");

   /* define and activate a window   */
   /* that enlarges the lower left */
   /* quadrant of the graph      */
  rc=gset("window", 1, 0, 0, 50, 50);
  rc=gset("transno", 1);

   /* insert the previous graph into */
   /* window 1            */
  rc=graph("insert", "GDSENLAR");

   /* display graph and end DSGI */
  rc=graph("update");
  rc=gterm();
run;
```

Display 31.8 Four Pie Charts Generated with DSGI



Display 31.9 Area of the Graph Enlarged by Using Windows



Features not explained in previous examples are described here:

- The GSET("WINDOW", . . .)function defines a window into which the graph is inserted. In this example, no viewport is defined, so the window coordinates map to the default viewport, which is the entire graphics output area. The result of using the default viewport is that only the portion of the graph enclosed by the coordinates of the window is displayed.
- The GRAPH("INSERT", . . .)function inserts a graph that was previously generated with DSGI. The output file to be inserted must be closed.

Using GASK Routines in DSGI

This example illustrates how to invoke GASK routines and how to display the returned values in the SAS log and write them to a data set.

This example assigns a predefined color to color index 2 and then invokes a GASK routine to get the name of the color associated with color index 2. The value returned from the GASK call is displayed in the log and written to a data set. Output 31.1 shows how the value appears in the log. Output 31.2 shows how the value appears in the data set in the OUTPUT window.

```
  /* execute DATA step with DSGI */
data routine;

   /* declare character variables used */
   /* in GASK subroutines       */
  length color $ 8;

   /* prepare SAS/GRAPH software */
   /* to accept DSGI statements */
  rc=ginit();
  rc=graph("clear");

   /* set color for color index 2 */
  rc=gset("colrep", 2, "orange");
```

```
   /* check color associated with color index 2 and */
   /* display the value in the LOG window      */
  call gask("colrep", 2, color, rc);
  put "Current FILCOLOR =" color;
  output;

   /* end DSGI */
  rc=graph("update");
  rc=gterm();
run;

  /* display the contents of ROUTINE */
proc print data=routine;
run;
```

Output 31.1 Checking the Color Associated with a Particular Color Index

```
3    /* execute DATA step with DSGI */
4  data routine;
5
6     /* declare character variables used */
7     /* in GASK subroutines       */
8    length color $ 8;
9
10     /* prepare SAS/GRAPH software */
11     /* to accept DSGI statements */
12   rc=ginit();
13   rc=graph("clear");
14
15     /* set color for color index 2 */
16   rc=gset("colrep", 2, "orange");
17
18     /* check color associated with color index 2 and */
19     /* display the value in the LOG window      */
20   call gask("colrep", 2, color, rc);
21   put "Current FILCOLOR =" color;
22   output;
23
24     /* end DSGI */
25   rc=graph("update");
26    rc=gterm();
27   run;

Current FILCOLOR =ORANGE
```

Output 31.2 Writing the Value of an Attribute to a Data Set

```
The SAS System             13:50 Tuesday, December 22, 1998   1

   Obs    color     rc

    1     ORANGE     0
```

Features not included in examples are described here:

- The GSET(“COLREP”, . . .)function assigns the predefined color “ORANGE” to the color index 2.

- GASK routines check the current value of an attribute. In this example, the GASK("COLREP", . . .)function returns the color associated with color index 2.
- A PUT statement displays the value of the COLOR argument in the log.
- An OUTPUT statement writes the value of COLOR to the ROUTINE data set.
- The GRAPH("UPDATE") function closes the graphics segment.
- The PRINT procedure displays the contents of the ROUTINE data set.

See Also

"Specifying the Catalog Name and Entry Name for Your GRSEGs" on page 100
for an explanation of graphics catalogs and catalog entries

Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327
for complete information about graphics options

"TITLE, FOOTNOTE, and NOTE Statements" on page 279
for details of using the TITLE and FOOTNOTE statements

"GOPTIONS Statement" on page 220
for details of using the GOPTIONS statement

Chapter 29, "Using Annotate Data Sets," on page 641
for an explanation of the Annotate facility

Chapter 32, "DATA Step Graphics Interface Dictionary," on page 813
for complete information on the functions and routines used with DSGI

SAS Language Reference: Dictionary
for information about additional functions and statements that can be used in the DATA step

CHAPTER

32

DATA Step Graphics Interface Dictionary

Overview

This chapter contains detailed descriptions of each command used in the DATA Step Graphics Interface (DSGI).

The following commands are associated with DSGI:

1. utility functions
   - GINIT
   - GPRINT
   - GTERM
2. GASK routines
3. GDRAW functions
4. GRAPH functions
5. GSET functions

Each routine or function is followed by an alphabetical listing of the operators used with it. For each operator, this chapter provides the statement syntax, other argument definitions, and notes about using the functions and routines, operating states, and return codes. Operating states are summarized in "Operating States" on page 775.

The syntax for all routines and functions contains the argument *return-code-variable*. This argument must be a numeric variable name and can be a different variable name for each routine.

The *return-code-variable* argument is used to debug DSGI programs. It contains the return code of the routine or function call. If the return code is any value other than 0, the routine or function did not execute properly.

Each routine and function has a different set of possible return codes. The return codes are listed in the heading for the routine or function. Refer to "Return Codes for DSGI Routines and Functions" on page 908 for an explanation of the return codes.

Operating States

This list summarizes the operating states in DSGI. For a detailed discussion of operating states, see "Operating States" on page 775.

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

Utility Functions

Utility functions enable you to initialize a session for DSGI, print error messages, and terminate the session.

GINIT

Initializes DSGI

Operating States: GKCL
Return Codes: 0, 1, 26, 301, 307
Resulting Operating State: WSAC

Syntax

return-code-variable=GINIT();

Description

The GINIT function performs three functions: it readies the library that contains SAS/GRAPH graphics routines, it opens a workstation, and it activates it. A workstation is a Graphics Kernel Standard (GKS) concept. GKS allows for multiple workstations to be open at the same time; however, for DSGI applications, you always use exactly one workstation. This function moves the operating state from GKCL to WSAC.

See Also

"GTERM" on page 815

GPRINT

Prints the specified interface error message

Operating States: All

Return Codes: 0

Syntax

return-code-variable=GPRINT(*code*);

Description

The GPRINT function displays the message that corresponds to the error code entered. You can use this routine if you have disabled automatic error logging but still want to display the message associated with a return code you have received.

Argument Definitions

code numeric constant or numeric variable name; should be the value of a return code received from some previous function.

See Also

"MESSAGE" on page 893

GTERM

Terminates DSGI

Operating States: WSAC

Return Codes: 0, 3

Resulting Operating State: GKCL

Syntax

return-code-variable=GTERM();

Description

The GTERM function performs three functions: it deactivates the workstation, closes the workstation, and closes the library that contains SAS/GRAPH routines. This function should be issued to free memory allocated by DSGI. This function moves the operating state from WSAC to GKCL.

See Also

"GINIT" on page 814

GASK Routines

When you use GASK routines, remember the following:

- All arguments are required.
- Most arguments are expressed as variable names. You can use any valid SAS variable name.
- If character arguments are expressed as character strings, they must be enclosed in quotation marks.
- All character variable names used as arguments *must* be declared in a previous LENGTH statement.
- GASK routines do not change the operating state.
- PUT statements display a value returned by a routine in the SAS log.
- OUTPUT statements write a value that is returned by a routine to a data set.

GASK routines enable you to check these current attribute settings:

ASF

ASPECT

CATALOG

CBACK

CLIP

COLINDEX

COLREP

DEVICE

FILCOLOR

FILINDEX

FILREP

FILSTYLE

FILTYPE

GRAPHLIST

HPOS

HSIZE

HTML

LINCOLOR

LININDEX

LINREP

LINTYPE

LINWIDTH

MARCOLOR

MARINDEX

MARREP
MARSIZE
MARTYPE
MAXDISP
NUMGRAPH
OPENGRAPH
PATREP
STATE
TEXALIGN
TEXCOLOR
TEXEXTENT
TEXFONT
TEXHEIGHT
TEXINDEX
TEXPATH
TEXREP
TEXUP
TRANS
TRANSNO
VIEWPORT
VPOS
VSIZE
WINDOW
WSACTIVE
WSOPEN

ASF

Finds whether an aspect source flag is bundled or separate

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8

Syntax

CALL GASK('ASF', *attribute, status, return-code-variable*);

Description

The GASK('ASF', . . .)routine returns the aspect source flag (ASF) of a particular attribute. Possible ASF values are BUNDLED (associated with a bundle index) and

INDIVIDUAL (separate from a bundle index). GASK('ASF', . . .)returns the default value INDIVIDUAL if you have not set the ASF for an attribute.

Argument Definitions

attribute character string enclosed in quotes or character variable name with one of the following values:

- □ FILCOLOR
- □ FILSTYLE
- □ FILTYPE
- □ LINCOLOR
- □ LINTYPE
- □ LINWIDTH
- □ MARCOLOR
- □ MARSIZE
- □ MARTYPE
- □ TEXCOLOR
- □ TEXFONT.

status character variable name; returns either the value BUNDLED or INDIVIDUAL.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"ASF" on page 872
"FILCOLOR" on page 877
"FILSTYLE" on page 880
"FILTYPE" on page 881
"LINCOLOR" on page 885
"LINTYPE" on page 887
"LINWIDTH" on page 888
"MARCOLOR" on page 888
"MARSIZE" on page 891
"MARTYPE" on page 891
"TEXCOLOR" on page 896
"TEXFONT" on page 897

ASPECT

Finds the aspect ratio

Operating States: All

Return Codes: 0

Syntax

CALL GASK('ASPECT', *aspect, return-code-variable*);

Description

The GASK('ASPECT', . . .)routine returns the current aspect ratio used to draw graphics output. GASK('ASPECT', . . .)searches for the current aspect ratio in the following order:

1. the aspect ratio set with the GSET('ASPECT', . . .)function
2. the ASPECT= graphics option
3. the device's default aspect ratio found in the device entry. For more information on device entries, see Chapter 38, "The GDEVICE Procedure," on page 1125.

Argument Definitions

aspect	numeric variable name; returns the aspect ratio.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

ASPECT= graphics option (see "ASPECT" on page 331)

"ASPECT" on page 873

CATALOG

Finds the catalog for the graphs

Operating States: All

Return Codes: 0

Syntax

CALL GASK('CATALOG', *libref, memname, return-code-variable*);

Description

The GASK('CATALOG', . . .)routine returns the libref and the name of the current output catalog. GASK('CATALOG', . . .)returns the default catalog, WORK.GSEG, if no other catalog has been specified with the GSET('CATALOG', . . .)function.

Argument Definitions

libref	character variable name; returns the libref of the library in which the current catalog is stored.

memname	character variable name; returns the name of the current output catalog.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"CATALOG" on page 874
"NUMGRAPH" on page 839
"OPENGRAPH" on page 839

CBACK

Finds the background color

Operating States: All
Return Codes: 0

Syntax

CALL GASK('CBACK', *cback, return-code-variable*);

Description

The GASK('CBACK', . . .)routine returns the current background color. GASK('CBACK', . . .)searches for the current background color in the following order:

1. the background color selected with the GSET('CBACK', . . .)function
2. the CBACK= graphics option
3. the default background color for the device found in the device entry. For more information about device entries, see Chapter 38, "The GDEVICE Procedure," on page 1125.

Argument Definitions

cback	character variable name; returns the background color name.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

CBACK= graphics option (see "CBACK" on page 335)
"CBACK" on page 875

CLIP

Finds whether clipping is on or off

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 55, 56

Syntax

CALL GASK('CLIP', *status*);

Description

The GASK('CLIP', . . .)routine checks whether clipping outside of viewports is enabled or disabled. One of the two following messages is displayed when this routine is called:

```
NOTE: Clipping is ON.
```

or

```
NOTE: Clipping is OFF.
```

Clipping is OFF by default.

Argument Definitions

status numeric variable name; returns the current setting, 55 (ON) or 56 (OFF), for clipping.

See Also

"CLIP" on page 875

COLINDEX

Finds the color indexes that have colors associated with them

Operating States: SGOP
Return Codes: 0, 4, 86, 87

Syntax

CALL GASK('COLINDEX', *n, index-array, return-code-variable*);

Description

The GASK('COLINDEX', . . .)routine returns the color indexes that currently have colors assigned to them.

Argument Definitions

n numeric constant or numeric variable name; tells how many color indexes you want returned. If *n* is expressed as a variable, the variable must be initialized. The variable returns the number of colors currently assigned. If *n* is expressed as a constant, this value is not returned.

index-array list of numeric variables into which the used color index numbers are returned. The list of variable names can be members of an array or OF argument lists (where the arguments are variables). If you are using an array, *index-array* must have been declared as an array. The dimension of the array is determined by the number of color indexes you want returned. Refer to the discussion of ARRAY in *SAS Language Reference: Dictionary* for more information about OF argument lists.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"COLREP" on page 822
"COLREP" on page 876

COLREP

Finds the color name associated with a color index

Operating States: SGOP
Return Codes: 0, 4, 86, 87

Syntax

CALL GASK('COLREP', *color-index, color, return-code-variable*);

Description

The GASK('COLREP', . . .)routine returns the predefined SAS color name associated with a color index. GASK('COLREP', . . .)searches for the current color assigned to a color index in the following order:

1. the color selected by the GSET('COLREP', . . .)function.
2. the COLORS= graphics option. If *color-index* is 2, the routine returns the second color from the color list of the COLORS= graphics option.
3. the device's default color list found in the device entry. If *color-index* is 2, the routine returns the second color from the default color list.

See "SAS Color Names and RGB Values in the SAS Registry" on page 175 for a list of SAS predefined color names.

Argument Definitions

color-index numeric constant; indicates the color index for which you want to check the color. Valid values are 1 to 256, inclusive.

color character variable name; returns the color name associated with *color-index*.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"COLINDEX" on page 821
"COLREP" on page 876

DEVICE

Finds the output graphics device

Operating States: All

Return Codes: 0

Syntax

CALL GASK('DEVICE', *device, return-code-variable*);

Description

The GASK('DEVICE', . . .)routine returns the current device. This routine returns the device set by one of the following methods:

- the GSET('DEVICE', . . .)function
- the DEVICE= graphics option
- the device you entered in the DEVICE prompt window
- the device you entered in the OPTIONS window.

There is no default value for a device. To use DSGI, you must specify a device. For more information about devices, see "Overriding the Default Device" on page 72.

Argument Definitions

device	character variable name; returns the name of the device driver.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

DEVICE= graphics option (see "DEVICE" on page 348)
See also: "Overriding the Default Device" on page 72

FILCOLOR

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

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('FILCOLOR', *color-index, return-code-variable*);

Description

The GASK('FILCOLOR', . . .)routine returns the current fill color. If a GSET('FILCOLOR', . . .)function has not been previously submitted, GASK('FILCOLOR', . . .)returns the default value, 1. The color index returned corresponds to a color specification in the following order:

1. the color assigned to a color name with the GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list found in the device entry.

Argument Definitions

color-index	numeric variable name; returns the color index of the fill color currently selected.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"COLREP" on page 822
"COLREP" on page 876
"FILCOLOR" on page 877

FILINDEX

Finds the bundle of fill area attributes that is active

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8

Syntax

CALL GASK('FILINDEX', *index, return-code-variable*);

Description

The GASK('FILINDEX', . . .)routine asks which fill bundle is active. If no fill bundles have been previously defined with GSET('FILREP', . . .)or activated with GSET('FILINDEX', . . .), GASK('FILINDEX', . . .)returns the default value, 1.

Argument Definitions

index numeric variable name; returns the index of the fill bundle currently selected.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"FILREP" on page 825
"FILREP" on page 879
"FILINDEX" on page 878

FILREP

Finds the fill area attributes associated with a bundle index

Operating States: GKOP, WSOP, WSAC, SGOP

Return Codes: 0, 8, 75, 76

Syntax

CALL GASK ('FILREP', *index, color-index, interior, style-index, return-code-variable*);

Description

The GASK('FILREP', . . .)routine returns the color, type of interior, and fill pattern associated with a specific fill bundle. If the bundle indicated by *index* has not been previously defined with the GSET('FILREP', . . .)function, DSGI issues the following error message:

```
ERROR: A representation for the specified fill area index has
 not been defined on this workstation.
```

Argument Definitions

index numeric constant or numeric variable name; indicates the fill bundle to check. Valid values are 1 to 20, inclusive. If *index* is expressed as a variable, the variable must be initialized to a value between 1 and 20.

color-index numeric variable name; returns the color index of the fill color associated with the bundle. The color index that is returned corresponds to a color specification in the following order:

1. a color index assigned to a color name with the GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list found in the device entry.

interior character variable name; returns the style of the interior associated with the bundle index:

- HATCH
- HOLLOW
- PATTERN
- SOLID.

style-index numeric variable name; returns the index of the fill pattern associated with the bundle. See the "FILSTYLE" on page 880 for the fill patterns represented by *style-index*.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

COLORS= graphics option (see "COLORS" on page 340)

"FILINDEX" on page 824

"COLREP" on page 876

"FILREP" on page 879

"FILSTYLE" on page 880

FILSTYLE

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

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('FILSTYLE', *style-index, return-code-variable*);

Description

The GASK('FILSTYLE', . . .)routine returns the current fill style of the interior when FILTYPE is PATTERN or HATCH. If no fill style has been previously selected with the GSET('FILSTYLE', . . .)function, GASK('FILSTYLE', . . .)returns the default value, 1.

Argument Definitions

style-index numeric variable name; returns the index of the fill pattern associated with the bundle. See the "FILSTYLE" on page 880 for the interior styles represented by *style-index*.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"FILTYPE" on page 827
"FILSTYLE" on page 880
"FILTYPE" on page 881

FILTYPE

Finds the type of the interior of the fill area

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('FILTYPE', *interior, return-code-variable*);

Description

The GASK('FILTYPE', . . .)routine returns the current fill type. If no fill type has been previously selected with the GSET('FILTYPE', . . .)function, GASK('FILTYPE', . . .)returns the default value, HOLLOW.

Argument Definitions

interior character variable name; returns the fill type that is active, that is, one of the following values:

- HATCH
- HOLLOW
- PATTERN
- SOLID.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"FILSTYLE" on page 826
"FILTYPE" on page 827

GRAPHLIST

Finds the names of segments in the current catalog

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('GRAPHLIST', *n, name-array, return-code-variable*);

Description

The GASK('GRAPHLIST', . . .)routine lists the first *n* names of the graphs that are in the current catalog. If a catalog has not been previously specified with the GRAPH('CATALOG', . . .)function, the routine returns names from the default catalog, WORK.GSEG.

The names returned are any of the following:

- those specified in the GRAPH('CLEAR', . . .)function
- if the name is omitted from the GRAPH('CLEAR' . . .)function, some form of DSGI: for example, DSGI, DSGI1, or DSGI2.
- the name specified in the NAME= option of a graphics procedure
- graphs previously created by other graphics procedures and already in the catalog.

Argument Definitions

n	numeric variable name; tells the maximum number of graph names you want returned. If you express *n* as a variable, the variable must be initialized to the maximum number of graph names you want returned.
name-array	list of character variable names into which the graph names are returned. The list of variable names can be members of an array or OF argument lists (where the arguments are variables). If you are using an array, *name-array* must be declared as an array. The dimension of the array is determined by the number of color indexes you want returned. See the discussion for ARRAY in *SAS Language Reference: Dictionary* for more information about OF argument lists.
return-code-variable	numeric variable names; returns the return code of the routine call.

See Also

"CLEAR" on page 866

HPOS

Finds the number of columns

Operating States: All

Return Codes: 0

Syntax

CALL GASK('HPOS',*hpos, return-code-variable*);

Description

The GASK('HPOS', . . .)routine returns the number of columns currently in the graphics output area. GASK('HPOS', . . .)searches for the current number of columns in the following order:

1. the value selected in the GSET('HPOS', . . .)function
2. the value of the HPOS= graphics option
3. the device's default HPOS value found in the device entry.

Argument Definitions

hpos	numeric variable name; returns the number of columns in the graphics output area.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"HSIZE" on page 829
"HPOS" on page 882
HPOS= graphics option (see "HPOS" on page 383)

HSIZE

Finds the horizontal dimension of the graphics output area

Operating States: All

Return Codes: 0

Syntax

CALL GASK('HSIZE', *hsize, return-code-variable*);

Description

The GASK('HSIZE', . . .)routine returns the current horizontal dimension, in inches, of the graphics output area. GASK('HSIZE', . . .)searches for the current horizontal dimension in the following order:

1. the value selected in the GSET('HSIZE', . . .)function
2. the value of the HSIZE= graphics option
3. the device's default HSIZE found in the device entry.

Argument Definitions

hsize	numeric variable name; the size of the graphics output area in the *x* dimension (in inches).
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"HPOS" on page 828
"HSIZE" on page 883
HSIZE= graphics option (see "HSIZE" on page 384)

HTML

Finds the HTML string that is in effect when one of the following graphic elements is drawn: bar, ellipse, fill, mark, pie, and text.

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('HTML', *string, return-code-variable*);

Description

The GASK('HTML', . . .)routine returns the current HTML string. If a GSET('HTML', . . .)function has not been previously submitted, GASK('HTML', . . .)returns the default value, null.

Argument Definitions

string	the HTML string invoked when an affected DSGI graphic element in a web page is clicked.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"BAR" on page 857
"ELLIPSE" on page 859
"FILL" on page 860
"MARK" on page 862
"PIE" on page 864
"TEXT" on page 865
"HTML" on page 884

LINCOLOR

Finds the current setting of the color to be used to draw lines

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8

Syntax

CALL GASK('LINCOLOR', *color-index, return-code-variable*);

Description

The GASK('LINCOLOR', . . .)routine returns the current line color. If a GSET('LINCOLOR', . . .)function has not been previously submitted, GASK('LINCOLOR', . . .)returns the default value, 1. The color index returned corresponds to a color specification in the following order:

1. the color specified in a GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list.

Argument Definitions

color-index	numeric variable name; returns the color index of the current line color.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"COLREP" on page 822
"COLREP" on page 876
"LINCOLOR" on page 885

LININDEX

Finds the index of the bundle of line attributes

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8

Syntax

CALL GASK('LININDEX', *index, return-code-variable*);

Description

The GASK('LININDEX', . . .)routine returns the current line bundle. If no line bundles have been previously defined with GSET('LINREP', . . .)or activated with GSET('LININDEX', . . .), GASK('LININDEX', . . .)returns the default value, 1.

Argument Definitions

index	numeric variable name; returns the index of the current line bundle.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"LINREP" on page 832
"LININDEX" on page 885
"LINREP" on page 886

LINREP

Finds the bundle of line attributes associated with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 60, 61

Syntax

CALL GASK ('LINREP', *index, color-index, width, type, return-code-variable*);

Description

The GASK('LINREP', . . .)routine returns the color, width, and line type associated with a specific line bundle. If the bundle indicated by *index* has not been previously defined with the GSET('LINREP', . . .)function, DSGI issues the following error message:

```
ERROR: A representation for the specified line type index has
 not been defined on this workstation.
```

Argument Definitions

index	numeric constant or numeric variable name; indicates the fill bundle to check. Valid values are 1 to 20, inclusive. If *index* is expressed as a variable, the variable must be initialized to a value between 1 and 20.
color-index	numeric variable name; returns the color index of the fill color associated with the bundle. The color index returned corresponds to a color specification in the following order: **1** a color index assigned with the GSET('COLREP', . . .)function **2** the *n*th color in the color list of the COLORS= graphics option **3** the *n*th color in the device's default color list.
width	numeric variable name; returns the line width (in pixels) associated with the bundle.

type numeric variable name; returns the index of the line type associated with the bundle. Refer to Figure 14.22 on page 277 for representations of the line types.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

COLORS= graphics option (see "COLORS" on page 340)

"COLREP" on page 822

"LININDEX" on page 831

"COLREP" on page 876

"LINREP" on page 886

LINTYPE

Finds the line type

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('LINTYPE', *type, return-code-variable*);

Description

The GASK('LINTYPE', . . .)routine returns the current line type. If no line type was previously selected with the GSET('LINTYPE', . . .)function, GASK('LINTYPE', . . .)returns the default value, 1.

Argument Definitions

type numeric variable name; returns the index of the line type currently selected. Refer to Figure 14.22 on page 277 for representations of the line types.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"LINTYPE" on page 887

LINWIDTH

Finds the line thickness

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('LINWIDTH', *width, return-code-variable*);

Description

The GASK('LINWIDTH', . . .)routine returns the current line width. If a line width has not been previously selected with the GSET('LINWIDTH', . . .)function, GASK('LINWIDTH', . . .)returns the default value, 1.

Argument Definitions

width	numeric variable name; returns the current line width (in units of pixels).
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"LINWIDTH" on page 888

MARCOLOR

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

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('MARCOLOR', *color-index, return-code-variable*);

Description

The GASK'MARCOLOR', . . .)routine returns the current marker color. If a GSET('MARCOLOR', . . .)function has not been previously submitted, GASK('MARCOLOR', . . .)returns the default value, 1. The color index returned corresponds to a color specification in the following order:

1 the color selected in a GSET('COLREP', . . .)function

2 the *n*th color in the color list of the COLORS= graphics option

3 the *n*th color in the device's default color list.

Argument Definitions

color-index	numeric variable name; returns the color index of the current marker color.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"COLREP" on page 822
"COLREP" on page 876
"MARCOLOR" on page 888

MARINDEX

Finds the index of the bundle of marker attributes currently selected

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('MARINDEX', *index, return-code-variable*);

Description

The GASK('MARINDEX', . . .)routine returns the current marker bundle. If no marker bundles have been previously defined with GSET('MARREP', . . .)or activated with GSET('MARINDEX', . . .), GASK('MARINDEX', . . .)returns the default value, 1.

Argument Definitions

index	numeric variable name; returns the index of the marker bundle currently selected.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"MARREP" on page 836
"MARINDEX" on page 889
"MARREP" on page 890

MARREP

Finds the bundle of marker attributes associated with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 64, 65

Syntax

CALL GASK('MARREP', *index, color-index, size, type, return-code-variable*);

Description

The GASK('MARREP' . . .)routine returns the color, size, and type of marker associated with a specific marker bundle. If the bundle indicated by *index* has not been previously defined with the GSET('MARREP', . . .)function, DSGI issues the following error message:

```
ERROR: A representation for the specified marker index has
 not been defined on this workstation.
```

Argument Definitions

index — numeric constant or numeric variable name; indicates the index of the fill bundle to check. Valid values are 1 to 20, inclusive. If *index* is expressed as a variable, the variable must be initialized to a value between 1 and 20.

color-index — numeric variable name; returns the color index of the fill color associated with the bundle. The color index returned corresponds to a color specification in the following order:

1. a color index assigned with the GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list.

size — numeric variable name; returns the marker size in units of the current window system.

type — numeric variable name; the index of the marker type associated with the bundle. See the "MARTYPE" on page 891 for an explanation of the marker indexes.

return-code-variable — numeric variable name; returns the return code of the routine call.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"COLREP" on page 822
"COLREP" on page 876
"MARINDEX" on page 889
"MARREP" on page 890

"MARTYPE" on page 891

MARSIZE

Finds the size of markers

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('MARSIZE', *size, return-code-variable*);

Description

The GASK('MARSIZE', . . .)routine returns the current marker size. If no marker size has been previously selected with the GSET('MARSIZE', . . .)function, GASK'MARSIZE', . . .)returns the default value, 1.

Argument Definitions

size	numeric variable name; returns the marker size in units of the current window system.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"MARSIZE" on page 891

MARTYPE

Finds the kind of markers

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('MARTYPE', *type, return-code-variable*);

Description

The GASK('MARTYPE', . . .)routine returns the current marker type. If no marker type has been previously selected with the GSET('MARTYPE', . . .)function, GASK('MARTYPE', . . .)returns the default value, 1.

Argument Definitions

type numeric variable name; returns the index of the marker type currently selected. See the function "MARTYPE" on page 891 for an explanation of the indexes for markers.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"MARTYPE" on page 891

MAXDISP

Finds the maximum display area size

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK ('MAXDISP', *units, x-dim, y-dim, x-pixels, y-pixels, return-code-variable*);

Description

The GASK('MAXDISP', . . .)routine returns the dimensions of the maximum display area for the device. This routine is useful when you need to know the maximum display area in order to determine the aspect ratio or to scale a graph.

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

Argument Definitions

units numeric variable name; returns a 1 to show that *x-dim* and *y-dim* are in meters.

x-dim numeric variable name; returns the dimension, in meters, in the *x* direction.

y-dim numeric variable name; returns the dimension, in meters, in the *y* direction.

x-pixels numeric variable name; returns the number of pixels in the *x* direction.

y-pixels numeric variable name; returns the number of pixels in the *y* direction.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"HSIZE" on page 829
"VSIZE" on page 852
"HSIZE" on page 883
"VSIZE" on page 906

NUMGRAPH

Finds the number of graphs in the current catalog

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('NUMGRAPH', *n, return-code-variable*);

Description

The GASK('NUMGRAPH', . . .)routine returns how many graphs are in the current catalog. The catalog checked is the catalog selected in the GSET('CATALOG', . . .)function, if specified; otherwise, it is the default catalog, WORK.GSEG.

Argument Definitions

n	numeric variable name; returns the number of graphs in the current catalog.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"CATALOG" on page 819
"CATALOG" on page 874

OPENGRAPH

Finds the name of the segment currently open

Operating States: SGOP

Return Codes: 0, 4

Syntax

CALL GASK('OPENGRAPH', *name, return-code-variable*);

Description

The GASK('OPENGRAPH', . . .)routine returns the name of the graph that is currently open.

The name returned is one of the following:

- the name specified in the GRAPH('CLEAR', . . .)function
- if the name is omitted from the GRAPH('CLEAR', . . .)function, some form of DSGI: for example, DSGI, DSGI1, and DSGI2.

Argument Definitions

name	character variable name; returns the name of the graph that is currently open.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"CLEAR" on page 866

PATREP

Finds the pattern name assigned to a style index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8. 79

Syntax

CALL GASK('PATREP', *index, pattern-name, hatch-name, return-code-variable);*

Description

The GASK('PATREP', . . .)routine returns the pattern name assigned to a style index.

Argument Definitions

index	numeric variable name; returns the index of the pattern currently selected.
pattern-name	character variable name; returns the name of the pattern at the specified index.
hatch-name	character variable name; returns the name of the hatch at the specified index.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"PATREP" on page 893

STATE

Finds the current operating state of DSGI

Operating States: All

Return Codes: 0

Syntax

CALL GASK('STATE', *status*);

Description

The GASK('STATE', . . .)routine returns the current operating state of DSGI.

Argument Definitions

status character variable name; returns one of the following values:

- GKCL
- GKOP
- SGOP
- WSAC
- WSOP.

See Also

"WSACTIVE" on page 854
"WSOPEN" on page 854

TEXALIGN

Finds the horizontal and vertical alignment of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('TEXALIGN', *halign, valign, return-code-variable*);

Description

The GASK('TEXALIGN', . . .)routine returns the current horizontal and vertical text alignment. If no values have been previously selected with the GSET('TEXALIGN', . . .)function, GASK('TEXALIGN', . . .)returns the default value NORMAL for both *halign* and *valign*.

Argument Definitions

halign — character variable name; indicates the horizontal alignment set by the GSET('TEXALIGN', . . .)function; returns one of the following values:

- CENTER
- LEFT
- NORMAL
- RIGHT.

valign — character variable name; indicates the vertical alignment set by the GSET('TEXALIGN', . . .)function; returns one of the following values:

- BASE
- BOTTOM
- HALF
- NORMAL
- TOP.

return-code-variable — numeric variable name; returns the return code of the routine call.

See Also

"TEXPATH" on page 846
"TEXUP" on page 848
"TEXALIGN" on page 894

TEXCOLOR

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

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8

Syntax

CALL GASK('TEXCOLOR', *color-index, return-code-variable*);

Description

The GASK('TEXCOLOR', . . .)routine returns the current text color. If a GSET('TEXCOLOR', . . .)function has not been previously submitted,

GASK('TEXCOLOR', . . .)returns the default value, 1. The color index returned corresponds to a color specification in the following order:

1. the color specified in a GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list.

Argument Definitions

color-index	numeric variable name; returns the color index of the color used to draw text.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"COLREP" on page 822
"COLREP" on page 876
"TEXCOLOR" on page 896

TEXEXTENT

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

Operating States: SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

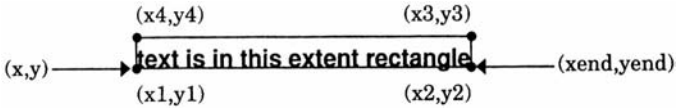
CALL GASK ('TEXEXTENT', *x, y, string, x-end, y-end, x1, x2, x3, x4, y1, y2, y3, y4, return-code-variable*);

Description

The GASK('TEXEXTENT', . . .)routine returns the text extent rectangle and text concatenation point for a specified text string. All text extent coordinates returned are in units of the current window system. If no text string is specified for *string*, GASK('TEXEXTENT', . . .)does not return values for the other arguments.

The text attributes and bundles affect the values returned by this query. See Figure 32.1 on page 843 for a diagram of the text extent rectangle (in the figure, *x,y* is always the place where the text string starts).

Figure 32.1 Text Extent Diagram



Argument Definitions

x	numeric variable name; *x* coordinates are in units based on the current window system; returns *x* coordinate after justification. The variable used to specify *x* must be initialized.
y	numeric variable name; *y* coordinates are in units based on the current window system; returns *y* coordinate after justification. The variable used to specify *y* must be initialized.
string	character string enclosed in single quotation marks or a character variable name; a set of characters for which the text extent rectangle and text concatenation point are calculated.
x-end	numeric variable name; returns the *x* coordinate of the point at which the next text string can be concatenated.
y-end	numeric variable name; returns the *y* coordinate of the point at which the next text string can be concatenated.
x1, *x2*, *x3*, *x4*, *y1*, *y2*, *y3*, *y4*	numeric variable names; return the text extent rectangles of the text strings as shown in Figure 32.1 on page 843.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"WINDOW" on page 853

"TEXT" on page 865

TEXFONT

Finds the font used to draw text strings

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('TEXFONT', *font, return-code-variable*);

Description

The GASK('TEXFONT' . . .)routine returns the current text font. GASK('TEXFONT', . . .)searches for the current font in the following order:

1. the value selected in the GSET('TEXFONT', . . .)function, if specified
2. the value of the FTEXT= graphics option, if specified
3. the device's default device-resident font if the device supports a device-resident font
4. the SIMULATE font.

Argument Definitions

font	character variable name; returns the font name.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

FTEXT= graphics options in (see "FTEXT" on page 363)
"TEXFONT" on page 897

TEXHEIGHT

Finds the character height of the text strings

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8

Syntax

CALL GASK('TEXHEIGHT', *height, return-code-variable*);

Description

The GASK('TEXHEIGHT', . . .)routine returns the current text height.
GASK('TEXHEIGHT', . . .)searches for the current text height in the following order:

1. the value selected in the GSET('TEXHEIGHT', . . .)function, if specified
2. the value of the HTEXT= graphics option, if specified
3. the default text height, 1.

Argument Definitions

height	numeric variable name; returns the character height in units of the current window system.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"TEXHEIGHT" on page 898
HTEXT= graphics options (see "HTEXT" on page 385)

TEXINDEX

Finds the index of the bundle of text attributes currently selected

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('TEXINDEX', *index, return-code-variable*);

Description

The GASK('TEXINDEX', . . .)routine returns the current text bundle. If no text bundles have been previously defined with GSET('TEXREP', . . .)or activated with GSET('TEXINDEX', . . .), GASK('TEXINDEX', . . .)returns the default value, 1.

Argument Definitions

index	numeric variable name; returns the text bundle index.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"TEXREP" on page 847
"TEXREP" on page 900
"TEXINDEX" on page 898

TEXPATH

Finds the direction of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('TEXPATH', *path, return-code-variable*);

Description

The GASK('TEXPATH', . . .)routine returns the current text path (reading direction). If TEXPATH has not been previously selected with the GSET('TEXPATH', . . .)function, GASK('TEXPATH', . . .)returns the default value, RIGHT. See the "TEXPATH" on page 899 for an illustration of text paths.

Argument Definitions

path	character variable name; returns one of the following values: □ DOWN

- □ LEFT
- □ RIGHT
- □ UP.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"TEXALIGN" on page 841
"TEXUP" on page 848
"TEXPATH" on page 899

TEXREP

Finds the attribute settings associated with a text bundle

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 68, 69

Syntax

CALL GASK('TEXREP', *index, color-index, font, return-code-variable*);

Description

The GASK('TEXREP', . . .)routine returns the color and font associated with a specific text bundle. If the bundle indicated by *index* has not been previously defined with the GSET('TEXREP', . . .)function, DSGI issues the following error message:

```
ERROR: A representation for the specified text index has
not been defined on this workstation.
```

Argument Definitions

index numeric constant or numeric variable name; indicates the fill bundle to check. Valid values are 1 to 20, inclusive. If *index* is expressed as a variable, the variable must be initialized to a value between 1 and 20.

color-index numeric variable name; returns the color index of the fill color associated with the bundle. The color index that is returned corresponds to a color specification in the following order:

1. a color index assigned with the GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list.

font character variable name; returns the text font associated with the bundle.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"COLREP" on page 822
"COLREP" on page 876
"TEXREP" on page 900

TEXUP

Finds the orientation (angle) of the text string

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8

Syntax

CALL GASK('TEXUP', *up-x, up-y, return-code-variable*);

Description

The GASK('TEXUP', . . .)routine returns the character up vector values. If TEXUP has not been previously selected with the GSET('TEXUP', . . .)function, GASK('TEXUP', . . .)returns the default values for *x* and *y*, 0 and 1. See the "TEXUP" on page 901 for an explanation of the vector values.

Argument Definitions

up-x	numeric variable name; returns the *x* component of the vector.
up-y	numeric variable name; returns the *y* component of the vector.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"TEXALIGN" on page 841
"TEXPATH" on page 846
"TEXUP" on page 901

TRANS

Finds the viewport and window coordinates associated with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 50

Syntax

CALL GASK ('TRANS', *n, vllx, vlly, vurx, vury, wllx, wlly, wurx, wury, return-code-variable*);

Description

The GASK('TRANS', . . .)routine returns the viewport and window coordinates associated with a particular transformation number. GASK('TRANS', . . .)returns the default coordinates for viewports and windows if other coordinates have not been defined for the transformation specified.

Argument Definitions

n	numeric constant or numeric variable name; indicates the number of the transformation to check. Valid values are 0 to 20, inclusive. If *n* is expressed as a variable, the variable must be initialized to a value between 0 and 20.
vllx	numeric variable name; returns the *x* coordinate of the lower-left viewport corner.
vlly	numeric variable name; returns the *y* coordinate of the lower-left viewport corner.
vurx	numeric variable name; returns the *x* coordinate of the upper-right viewport corner.
vury	numeric variable name; returns the *y* coordinate of the upper-right viewport corner.
wllx	numeric variable name; returns the *x* coordinate of the lower-left window corner.
wlly	numeric variable name; returns the *y* coordinate of the lower-left window corner.
wurx	numeric variable name; returns the *x* coordinate of the upper-right window corner.
wury	numeric variable name; returns the *y* coordinate of the upper-right window corner.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"TRANSNO" on page 850
"VIEWPORT" on page 850
"WINDOW" on page 853
"TRANSNO" on page 903
"VIEWPORT" on page 904
"WINDOW" on page 907

TRANSNO

Finds the number of the transformation to be used

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Syntax

CALL GASK('TRANSNO', *n, return-code-variable*);

Description

The GASK('TRANSNO', . . .)routine returns the current transformation. If a transformation has not been previously selected with the GSET('TRANSNO', . . .)function, GASK('TRANSNO', . . .)returns the number of the default transformation, 0.

Argument Definitions

n	numeric variable name; returns the number of the current transformation.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"TRANS" on page 848
"VIEWPORT" on page 850
"WINDOW" on page 853
"VIEWPORT" on page 904
"WINDOW" on page 907
"TRANSNO" on page 903

VIEWPORT

Finds coordinates of the viewport associated with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50

Syntax

CALL GASK('VIEWPORT', *n, llx, lly, urx, ury, return-code-variable*);

Description

The GASK('VIEWPORT', . . .)routine returns the coordinates of the viewport associated with the specified transformation. If a viewport has not been defined with the GSET('VIEWPORT', . . .)function for the specified transformation, *n*, GASK('VIEWPORT', . . .)returns the default coordinates for the viewport, (0,0) and (1,1).

Argument Definitions

n	numeric constant or numeric variable name; indicates the transformation number assigned to the viewport to check. Valid values are 0 to 20, inclusive. If *n* is expressed as a variable, the variable must be initialized to a value between 0 and 20.
llx	numeric variable name; returns the *x* coordinate of the lower-left corner.
lly	numeric variable name; returns the *y* coordinate of the lower-left corner.
urx	numeric variable name; returns the *x* coordinate of the upper-right corner.
ury	numeric variable name; returns the *y* coordinate of the upper-right corner.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"TRANS" on page 848
"TRANSNO" on page 850
"WINDOW" on page 853
"TRANSNO" on page 903
"VIEWPORT" on page 904
"WINDOW" on page 907

VPOS

Finds the number of rows

Operating States: All

Return Codes: 0

Syntax

CALL GASK('VPOS', *vpos, return-code-variable*);

Description

The GASK('VPOS', . . .)routine returns the current number of rows in the graphics output area. GASK('VPOS', . . .)searches for the current number of rows in the following order:

1. the value selected in the GSET('VPOS', . . .)function
2. the value of the VPOS= graphics option
3. the device's default VPOS value found in the device entry.

Argument Definitions

vpos	numeric variable name; returns the number of rows in the graphics output area.
return-code-variable	numeric variable name; returns the return code of the routine call.

See Also

"HPOS" on page 828
"VSIZE" on page 852
"VPOS" on page 905
VPOS= graphics option (see "VPOS" on page 430)

VSIZE

Finds the vertical dimension of the graphics output area

Operating States: All

Return Codes: 0

Syntax

CALL GASK('VSIZE', *vsize, return-code-variable*);

Description

The GASK('VSIZE', . . .)routine returns the current vertical dimension, in inches, of the graphics output area. GASK('VSIZE', . . .)searches for the current vertical dimension in the following order:

1. the value selected in the GSET('VSIZE', . . .)function
2. the value of the VSIZE= graphics option
3. the device's default VSIZE found in the device entry.

Argument Definitions

vsize	numeric variable name; returns the size of the graphics output area in the *y* dimension (in inches).

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"HSIZE" on page 829

"VPOS" on page 851

"VSIZE" on page 906

VSIZE= graphics option (see "VSIZE" on page 430)

WINDOW

Finds the coordinates of the window associated with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50

Syntax

CALL GASK('WINDOW', *n, llx, lly, urx, ury, return-code-variable*);

Description

The GASK('WINDOW', . . .)routine returns the coordinates of the window associated with the specified transformation number. If no window has been defined with the GSET('WINDOW', . . .)function for transformation *n*, GASK('WINDOW', . . .)returns the default window coordinates, which are device dependent.

Argument Definitions

n numeric constant or numeric variable name; indicates the transformation number of the window to check. Valid values are 0 to 20, inclusive. If *n* is expressed as a variable, the variable must be initialized to a value between 0 and 20.

llx numeric variable name; returns the *x* coordinate of the lower-left corner.

lly numeric variable name; returns the *y* coordinate of the lower-left corner.

urx numeric variable name; returns the *x* coordinate of the upper-right corner.

ury numeric variable name; returns the *y* coordinate of the upper-right corner.

return-code-variable numeric variable name; returns the return code of the routine call.

See Also

"TRANS" on page 848
"TRANSNO" on page 850
"VIEWPORT" on page 850
"TRANSNO" on page 903
"VIEWPORT" on page 904
"WINDOW" on page 907

WSACTIVE

Finds whether the interface is active

Operating States: All
Return Codes: 29, 30

Syntax

CALL GASK('WSACTIVE', *status*);

Description

The GASK('WSACTIVE', . . .)routine asks if the workstation is active. When the workstation is active, you can execute certain DSGI routines and functions.

Argument Definitions

status numeric variable name; returns either 29 (active) or 30 (inactive).

See Also

"STATE" on page 841
"WSOPEN" on page 854

WSOPEN

Finds whether the interface is open

Operating States: All
Return Codes: 24, 25

Syntax

CALL GASK('WSOPEN', *status*);

Description

The GASK('WSOPEN', . . .)routine asks if the workstation is open. If a workstation is open, the graphics catalog can be accessed.

Argument Definitions

status numeric variable name; returns either 24 (open) or 25 (closed).

See Also

"WSACTIVE" on page 854

GDRAW Functions

GDRAW functions create graphics elements. Each GDRAW operator is associated with a set of GSET operators that control its attributes. The color, height, and font attributes, for the GDRAW('TEXT', . . .)function are controlled by GSET('TEXCOLOR', . . .), GSET('TEXHEIGHT', . . .), and GSET('TEXFONT', . . .), respectively. For a complete list of the attributes associated with each GDRAW function, see Table 31.2 on page 781. The complete graph is displayed after the GRAPH('UPDATE', . . .)function is submitted.

When using GDRAW functions:

- all arguments must be specified
- all arguments are specified as variables or constants
- if you express an argument as a variable, the variable must be initialized
- all character arguments expressed as character strings must be enclosed in quotes
- all character variable names used as arguments *must* be declared in a LENGTH statement
- all character constants must be enclosed in single or double quotes

GDRAW functions:

- ARC
- BAR
- ELLARC
- ELLIPSE
- FILL
- IMAGE
- LINE
- MARK
- MESSAGE
- PIE
- TEXT

ARC

Draws a circular arc

Operating States: SGOP

Return Codes: 0, 4, 61, 86

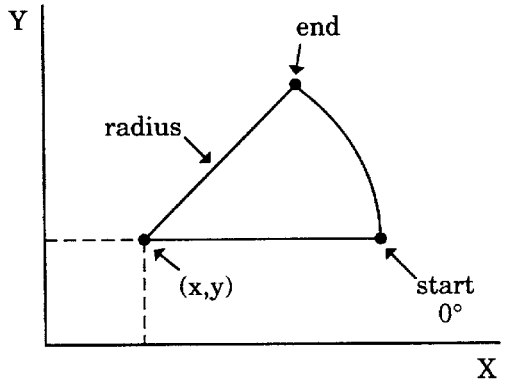
Syntax

return-code-variable=GDRAW('ARC', *x, y, radius, start, end*);

Description

The GDRAW('ARC', . . .)function draws a circular arc. The line attributes and bundles affect the appearance of this primitive. See Table 31.2 on page 781 for a list of these attributes. Figure 32.2 on page 856 illustrates the arguments used with GDRAW('ARC', . . .).

Figure 32.2 Arguments Used with the GDRAW('ARC', ...) Function



Argument Definitions

x	numeric constant or numeric variable name; specifies the *x* coordinate of the position of the arc on the display; *x* coordinates are in units based on the current window system.
y	numeric constant or numeric variable name; specifies the *y* coordinate of the position of the arc on the display; *y* coordinates are in units based on the current window system;
radius	numeric constant or numeric variable name; the arc radius size is in units based on the current window system.
start	numeric constant or numeric variable name; the starting angle of the arc is in degrees, with 0 degrees at 3 o'clock.
end	numeric constant or numeric variable name; the ending angle of the arc is in degrees, with 0 degrees at 3 o'clock.

See Also

"ELLARC" on page 858
"PIE" on page 864
"LINCOLOR" on page 885
"LININDEX" on page 885
"LINREP" on page 886
"LINTYPE" on page 887
"LINWIDTH" on page 888

BAR

Draws a rectangle

Operating States: SGOP

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

Syntax

return-code-variable=GDRAW('BAR', *x1, y1, x2, y2*);

Description

The GDRAW('BAR', . . .)function draws a rectangular bar whose sides are parallel to the sides of the display area. The fill attributes and bundles affect the appearance of this graphics element. See Table 31.2 on page 781 for a list of these attributes. Figure 32.3 on page 857 illustrates the arguments used with GDRAW('BAR', . . .).

Figure 32.3 Points that Draw a Bar

Argument Definitions

x1 numeric constant or numeric variable name; refers to the *x* coordinate of one corner of the bar.

y1 numeric constant or numeric variable name; refers to the *y* coordinate of one corner of the bar.

x2 numeric constant or numeric variable name; refers to the *x* coordinate of the corner of the bar that is diagonally opposite to the corner of (*x1,y1*).

y2 numeric constant or numeric variable name; refers to the *y* coordinate of the corner of the bar that is diagonally opposite to the corner of (*x1*,*y1*).

See Also

"FILL" on page 860
"FILCOLOR" on page 877
"FILINDEX" on page 878
"FILREP" on page 879
"FILTYPE" on page 881
"FILSTYLE" on page 880
"HTML" on page 884

ELLARC

Draws an elliptical arc

Operating States: SGOP

Return Codes: 0, 4, 61, 86

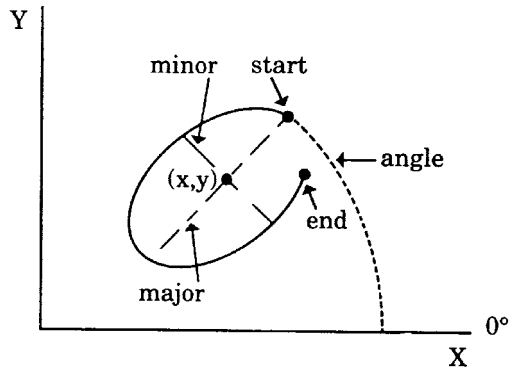
Syntax

return-code-variable =GDRAW('ELLARC', *x, y, major, minor, start, end, angle*);

Description

The GDRAW('ELLARC', . . .)function draws a hollow section of an ellipse. The line attributes and bundles affect the appearance of this primitive. See Table 31.2 on page 781 for a list of these attributes. Figure 32.4 on page 858 illustrates the arguments used with GDRAW('ELLARC', . . .)and GDRAW('ELLIPSE', . . .).

Figure 32.4 Arguments Used with GDRAW('ELLARC',...) function and GDRAW('ELLIPSE',...) function



Argument Definitions

x	numeric constant or numeric variable name; *x* coordinates are in units based on the current window system.
y	numeric constant or numeric variable name; *y* coordinates are in units based on the current window system.
major	numeric constant or numeric variable name; the major axis lengths for the elliptical arc.
minor	numeric constant or numeric variable name; the minor axis lengths for the elliptical arc.
start	numeric constant or numeric variable name; the starting angle from the major axis, in degrees, for the elliptical arc with 0 degrees beginning at the major axis.
end	numeric constant or numeric variable name; the ending angle from the major axis, in degrees, for the elliptical arc with 0 degrees at 3 o'clock.
angle	numeric constant or numeric variable name; the angle that the major axis of the elliptical arc has to 0 degrees (with 0 degrees at 3 o'clock).

See Also

"ELLIPSE" on page 859
"LINCOLOR" on page 885
"LINTYPE" on page 887
"LINWIDTH" on page 888
"LINREP" on page 886
"LININDEX" on page 885

ELLIPSE

Draws an ellipse

Operating States: SGOP

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

Syntax

return-code-variable =GDRAW('ELLIPSE', *x, y, major, minor, start, end, angle*);

Description

The GDRAW('ELLIPSE', . . .)function draws a filled section of an ellipse. The fill attributes and bundles affect the appearance of this primitive. See Table 31.2 on page 781 for a list of these attributes. Figure 32.4 on page 858 illustrates the arguments used with GDRAW('ELLARC', . . .)and GDRAW('ELLIPSE', . . .).

Argument Definitions

x	numeric constant or numeric variable name; the *x* coordinate of the position of the ellipse on the display.
y	numeric constant or numeric variable name; the *y* coordinate of the position of the ellipse on the display.
major	numeric constant or numeric variable name; the major axis length for the ellipse.
minor	numeric constant or numeric variable name; the minor axis length for the ellipse.
start	numeric constant or numeric variable name; the starting angle for the ellipse from the major axis, with 0 degrees beginning at the major axis.
end	numeric constant or numeric variable name; the ending angle for the ellipse from the major axis, with 0 degrees at 3 o'clock.
angle	numeric constant or numeric variable name; the angle that the major axis of the ellipse has to 0 degrees, with 0 degrees at 3 o'clock.

See Also

"ELLARC" on page 858
"FILCOLOR" on page 877
"FILINDEX" on page 878
"FILREP" on page 879
"FILTYPE" on page 881
"HTML" on page 884

FILL

Draws a filled area

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

Syntax

return-code-variable=GDRAW('FILL', *n, x-values, y-values*);

Description

The GDRAW('FILL' . . .)function draws a filled polygon. The fill attributes and bundles affect the appearance of this primitive. See Table 31.2 on page 781 for a list of these attributes.

Note: All of the *x* coordinates are listed in the function first, followed by the *y* coordinates. This primitive takes the first *n* values and stores them as *x* coordinates. The next *n* values are stored as *y* coordinates. △

Argument Definitions

n	numeric constant or numeric variable name; the number of vertices (*x* and *y* pairs) in the polygon. You can specify a missing value (.) for *n*. If *n* is missing, the number of vertices is computed from the number of *x* and *y* arguments.
x-values	list of numeric constants, variables, or OF arguments that describe the *x* coordinates for the vertices in units based on the current window system.
y-values	list of numeric constants, variables, or OF arguments that describe the *y* coordinates for the vertices in units based on the current window system.

See Also

"BAR" on page 857
"FILCOLOR" on page 877
"FILINDEX" on page 878
"FILREP" on page 879
"FILTYPE" on page 881
"FILSTYLE" on page 880
"HTML" on page 884

IMAGE

Displays an image

Operating State: SGOP

Return Codes: 0, 150

Syntax

return-code-variable=GDRAW('IMAGE', '*external-file*', *x1*, *y1*, *x2*, *y2*, '*style*');

Description

The GDRAW('IMAGE', . . .) function displays the specified image within opposing pairs of coordinates. The format of the external image file varies between operating environments. The (x1, y1) coordinate pair specifies one corner of the image, and the (x2, y2) coordinate pair specifies the opposite corner of the image. The *style* parameter must be either 'TILE' to copy the image as many times as necessary to fill the area; or 'FIT' to stretch one instance of the image to fill the area.

For a list of the file types that you use, see "Image File Types Supported by SAS/GRAPH" on page 181.

LINE

Draws a polyline

Operating States: SGOP

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

Syntax

return-code-variable=GDRAW('LINE', *n, x-values, y-values*);

Description

The GDRAW('LINE' . . .)function draws one line, a series of connected lines, or a dot. The line attributes and bundles affect the appearance of this primitive. See Table 31.2 on page 781 for a list of these attributes.

Note: All of the *x* coordinates are listed in the function first, followed by the *y* coordinates. This primitive takes the first *n* values and stores them as *x* coordinates and the next *n* values and stores them as *y* coordinates. △

Argument Definitions

n	numeric constant or numeric variable name; the number of vertices (*x* and *y* pairs) in the polygon. You can specify a missing value (.) for *n*. If *n* is missing, the number of vertices is computed from the number of *x* and *y* pairs.
x-values	list of numeric constants, variables, or OF arguments that describe the *x* coordinates for the vertices in units based on the current window system.
y-values	list of numeric constants, variables, or OF argument lists that describe the *y* coordinates for the vertices in units based on the current window system.

See Also

"FILCOLOR" on page 877
"LININDEX" on page 885
"LINREP" on page 886
"LINTYPE" on page 887
"LINWIDTH" on page 888

MARK

Draws a polymarker

Operating States: SGOP

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

Syntax

return-code-variable=GDRAW ('MARK', *n, x-values, y-values*);

Description

The GDRAW('MARK', . . .)function draws a series of symbols. The marker attributes and bundles affect the appearance of this primitive. See Table 31.2 on page 781 for a list of these attributes. Refer to the "MARTYPE" on page 891 for a list of symbols that you can draw with GDRAW('MARK', . . .).

Note: All of the *x* coordinates are listed in the function first, followed by the *y* coordinates. This primitive takes the first *n* values and stores them as *x* coordinates and the next *n* values and stores them as *y* coordinates. △

Argument Definitions

n	numeric constant or numeric variable name; the number of times the symbol is drawn. You can specify a missing value (.) for *n*. If *n* is missing, the number of vertices is calculated from the number of *x* and *y* pairs.
x-values	list of numeric constants, variables, or OF arguments that describe the *x* coordinates of the symbols in units based on the current window system.
y-values	list of numeric constants, variables, or OF arguments that describe the *y* coordinates of the symbols in units based on the current window system.

See Also

"TEXT" on page 865
"HTML" on page 884
"MARCOLOR" on page 888
"MARINDEX" on page 889
"MARREP" on page 890
"MARTYPE" on page 891

MESSAGE

Prints a message in the SAS log

Operating States: All

Return Codes: 0

Syntax

return-code-variable=GDRAW('MESSAGE', *message*);

Description

The GDRAW('MESSAGE', . . .)function prints a message in the SAS log. This function can be used for debugging applications or for printing custom messages for your application.

Argument Definitions

message	character string enclosed in quotes or character variable name; the text to be printed in the log.

See Also

"MESSAGE" on page 893
"GPRINT" on page 814

PIE

Draws a filled circle or section of a filled circle

Operating States: SGOP
Return Codes: 0, 4, 76, 79, 80, 86

Syntax

return-code-variable=GDRAW('PIE', *x, y, radius, start, end*);

Description

The GDRAW('PIE', . . .)function draws a filled section of a circular arc. The fill attributes and bundles affect the appearance of this primitive. See Table 31.2 on page 781 for a list of these attributes.

Argument Definitions

x	numeric constant or numeric variable name; *x* coordinates are in units based on the current window system.
y	numeric constant or numeric variable name; *y* coordinates are in units based on the current window system.
radius	numeric constant or numeric variable name; the pie radius size in units based on the current window system.
start	numeric constant or numeric variable name; the starting angle of the pie, with 0 degrees at 3 o'clock on the unit circle.
end	numeric constant or numeric variable name; the ending angle of the pie, with 0 degrees at 3 o'clock on the unit circle.

See Also

"ARC" on page 856
"FILCOLOR" on page 877
"FILINDEX" on page 878
"FILREP" on page 879
"FILTYPE" on page 881
"FILSTYLE" on page 880
"HTML" on page 884

TEXT

Draws a text string

Operating States: SGOP

Return Codes: 0, 4, 69, 86

Syntax

return-code-variable=GDRAW('TEXT', *x, y, string*);

Description

The GDRAW('TEXT', . . .)function draws a text string. The text attributes and bundles affect the appearance of this primitive. See Table 31.2 on page 781 for a list of these attributes.

Argument Definitions

x	numeric constant or numeric variable name; *x* coordinates are in units based on the current window system.
y	numeric constant or numeric variable name; *y* coordinates are in units based on the current window system.
string	character string enclosed in quotes or character variable name; a set of characters to be drawn on the output beginning at position (*x*,*y*).

See Also

"MARK" on page 862
"HTML" on page 884
"TEXCOLOR" on page 896
"TEXINDEX" on page 898
"TEXREP" on page 900
"TEXHEIGHT" on page 898

GRAPH Functions

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

When using GRAPH functions, remember the following:

- All arguments are specified as variables or constants. If you express an argument as a variable, the variable must be initialized.
- All character arguments expressed as character strings must be enclosed in quotes.
- All character variable names used as arguments *must* be declared in a LENGTH statement.
- All character constants must be enclosed in single or double quotes.

GRAPH functions:

- CLEAR
- COPY
- DELETE
- INSERT
- PLAY
- RENAME
- UPDATE

CLEAR

Opens a graphics segment for output

Operating States: WSAC
Return Codes: 0, 3, 301, 302
Resulting Operating State: SGOP

Syntax

return-code-variable=GRAPH ('CLEAR'<, *name*> <, *des*><, *byline*>);

Description

The GRAPH('CLEAR', . . .)function opens a graphics segment for output in the current catalog. The first parameter, 'CLEAR', is the only required one. The values of *name*, *des*, and *byline* are displayed in catalog listings and in catalog information in the GREPLAY procedure.

If the name specified is an existing graph, DSGI adds a suffix number to the name. If PIE is chosen for the name and it already exists, DSGI names the output PIE1; the next time the code is submitted, DSGI names the output PIE2, and so forth.

This function moves the operating state from WSAC to SGOP.

Argument Definitions

name character string enclosed in quotes or character variable name; gives a name to the graph to be opened. If *name* is not specified, DSGI

assigns the graph a name that is some form of DSGI: for example, DSGI, DSGI1, and DSGI2.

des character string enclosed in quotes or character variable name; gives a description to the graph to be opened. If *des* is not specified, DSGI assigns the following description to the catalog entry: Graph from DATA Step Graphics Interface.

BY line character string enclosed in quotes or character variable name; gives another line of description for the graph. The byline appears under the titles on the graph. DSGI does not provide a default byline.

See Also

"OPENGRAPH" on page 839
"UPDATE" on page 870

COPY

Copies a graph

Operating States: GKOP, WSOP, WSAC, SGOP

Return Codes: 0, 8, 307

Syntax

return-code-variable=GRAPH('COPY', *name, new-name)*;

Description

The GRAPH('COPY', . . .)function copies a graph to another catalog entry. The graph to be copied must be closed, and be in the current catalog. You cannot copy from one catalog to another. The new graph is also in the current catalog.

Argument Definitions

name character string enclosed in quotes or character variable name; name of the graph to be copied.

new-name character string enclosed in quotes or character variable name; name of the graph to be created.

See Also

"CATALOG" on page 819
"DELETE" on page 868
"INSERT" on page 868
"CATALOG" on page 874

DELETE

Deletes a graph

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 4, 8, 307

Syntax

return-code-variable=GRAPH('DELETE', *name*);

Description

The GRAPH('DELETE', . . .)function deletes a graph in the current catalog. The graph does not have to be closed to be deleted.

Argument Definitions

name character string enclosed in quotes or character variable name; the name of the graph to delete.

See Also

"CATALOG" on page 819
"COPY" on page 867
"CATALOG" on page 874

INSERT

Inserts a previously created segment into the currently open graph

Operating States: SGOP
Return Codes: 0, 4, 302, 307

Syntax

return-code-variable=GRAPH('INSERT', *name*);

Description

The GRAPH('INSERT', . . .)function inserts a graph into the currently open graph. The graph to be inserted must be closed and be in the current catalog.

Argument Definitions

name character string enclosed in quotes or character variable name; the name of a graph to be inserted.

See Also

"CATALOG" on page 819
"COPY" on page 867
"CATALOG" on page 874

PLAY

Displays the specified graph on your output

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 307

Syntax

return-code-variable=GRAPH('PLAY', *graph-name*);

Description

The GRAPH('PLAY', . . .)function displays the specified graph on your output.

Argument Definitions

graph-name character variable name; the name of the graph you would like to play.

See Also

"UPDATE" on page 870

RENAME

Renames a graph

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 307

Syntax

return-code-variable=GRAPH('RENAME', *name, new-name*);

Description

The GRAPH('RENAME', . . .)function renames a graph. The graph to be renamed must be in the current catalog and be closed.

Argument Definitions

name character string enclosed in quotes or character variable name; the name of the closed graph that is to be changed.

new-name character string enclosed in quotes or character variable name; the new name for the graph.

See Also

"CATALOG" on page 819
"INSERT" on page 868
"CATALOG" on page 874

UPDATE

Completes the currently open graph, or displays it, or both

Operating States: SGOP
Return Codes: 0, 4
Resulting Operating State: WSAC

Syntax

return-code-variable=GRAPH('UPDATE' <, '*show*'>);

Description

The GRAPH('UPDATE', . . .)function closes the graph currently open and displays it. DSGI operates in buffered mode, so the picture is never displayed until this function is called.

This function can be called only once for the currently open graph. Therefore, you cannot incrementally build a graph; however, you can close the currently open graph and later insert it into another graph within the same DATA step.

This function moves the operating state from SGOP to WSAC.

Argument Definitions

show character string, optional; valid values are SHOW and NOSHOW. If SHOW is specified, the graph is displayed. If NOSHOW is specified, the graph is closed and not displayed.

See Also

"CLEAR" on page 866

GSET Functions

GSET functions allow you to set attributes for the graphics elements. Some GSET functions set the attributes for a subset of graphics primitives. Attributes prefixed by

FIL control the appearance of the graphics primitives GDRAW('BAR', . . .), GDRAW('ELLIPSE', . . .), GDRAW('FILL', . . .), and GDRAW('PIE', . . .). See Table 31.2 on page 781 for a complete list of the attributes that control the appearance of the graphics primitives.

Some GSET functions affect the appearance of the entire graphics output. GSET('HPOS', . . .)and GSET('VPOS', . . .)set the number of columns and rows for the output. See each GSET function for the aspect of the graphics output it controls.

When using GSET functions, remember the following:

- All arguments must be specified.
- All arguments are specified as variables or constants. If you express an argument as a variable, the variable must be initialized.
- All character arguments that are expressed as character strings must be enclosed in quotation marks.
- All character variable names used as arguments *must* be declared in a LENGTH statement.
- All character constants must be enclosed in single or double quotation marks.

GSET functions:

ASF

ASPECT

CATALOG

CBACK

CLIP

COLREP

DEVICE

FILCOLOR

FILINDEX

FILREP

FILSTYLE

FILTYPE

HPOS

HSIZE

HTML

LINCOLOR

LININDEX

LINREP

LINTYPE

LINWIDTH

MARCOLOR

MARINDEX

MARREP

MARSIZE

MARTYPE
MESSAGE
PATREP
TEXALIGN
TEXCOLOR
TEXFONT
TEXHEIGHT
TEXINDEX
TEXPATH
TEXREP
TEXUP
TRANSNO
VIEWPORT
VPOS
VSIZE
WINDOW

ASF

Specifies an aspect source flag to bundle or separate attributes

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8
Default Value: INDIVIDUAL

Syntax

return-code-variable=GSET('ASF', *attribute, status*);

Description

The GSET('ASF', . . .)function sets an attribute's aspect source flag (ASF) so that it can be used in a bundle (BUNDLED) or individually (INDIVIDUAL).

If an attribute's ASF is set to 'BUNDLED', it cannot be used outside of a bundle. It must be defined in a GSET('*xxx*REP', . . .)function and activated with a GSET('*xxx*INDEX', . . .)function, where *xxx* can have one of the following values: FIL, LIN, MAR, TEX.

If an attribute's ASF is set to 'INDIVIDUAL', it cannot be used with a bundle. In this case, the attribute is set with a GSET('*attribute*', . . .). The values of *attribute* are listed in "Argument Definitions."

Argument Definitions

attribute character string enclosed in quotes or character variable name with one of the following values:

- □ FILCOLOR
- □ FILSTYLE
- □ FILTYPE
- □ LINCOLOR
- □ LINTYPE
- □ LINWIDTH
- □ MARCOLOR
- □ MARSIZE
- □ MARTYPE
- □ TEXCOLOR
- □ TEXFONT.

status character string enclosed in quotation marks or character variable name; accepts either the value BUNDLED or INDIVIDUAL.

See Also

"ASF" on page 817
"FILCOLOR" on page 877
"FILSTYLE" on page 880
"FILTYPE" on page 881
"LINCOLOR" on page 885
"LINTYPE" on page 887
"LINWIDTH" on page 888
"MARCOLOR" on page 888
"MARSIZE" on page 891
"MARTYPE" on page 891
"TEXCOLOR" on page 896
"TEXFONT" on page 897

ASPECT

Specifies the aspect ratio

Operating States: GKCL
Return Codes: 0, 1, 90, 307
Default Value: 0.0

Syntax

return-code-variable=GSET('ASPECT', *aspect*);

Description

The GSET('ASPECT', . . .)function sets the aspect ratio used to draw graphics output. GSET('ASPECT', . . .)affects only pies, arcs, and software text.

Argument Definitions

aspect numeric constant or numeric variable name; specifies the aspect ratio and cannot be less than 0.

See Also

ASPECT= graphics option (see "ASPECT" on page 331)

"ASPECT" on page 818

CATALOG

Specifies the catalog for the graphs

Operating States: GKCL

Return Codes: 0, 1

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

Syntax

return-code-variable=GSET('CATALOG', *libref, catalog-name*);

Description

The GSET('CATALOG', . . .)function makes the specified catalog the current catalog in which to store graphs generated with DSGI. GSET('CATALOG', . . .)creates the catalog if it does not exist.

The values of *libref* and *catalog-name* cannot exceed eight characters. The number of characters allowed for a catalog name varies across operating environments; see the SAS companion for your operating system. *Libref* should have been defined through the LIBNAME statement.

Argument Definitions

libref character string enclosed in quotation marks or character variable name; points to the library that contains the catalog.

catalog-name character string enclosed in quotation marks or character variable name; specifies the catalog name to be used.

See Also

"CATALOG" on page 819

"GRAPHLIST" on page 827

"NUMGRAPH" on page 839

CBACK

Specifies the background color

Operating States: GKCL

Return Codes: 0, 1

Default Value: 1. CBACK= graphics option, if specified; 2. device's default background color.

Syntax

return-code-variable=GSET('CBACK', *cback*);

Description

The GSET('CBACK', . . .)function sets the background color. GSET('CBACK', . . .)has the same effect as the CBACK= graphics option.

Argument Definitions

cback character string enclosed in quotation marks or character variable name; can contain any predefined SAS color name. See "SAS Color Names and RGB Values in the SAS Registry" on page 175 for a list of predefined SAS color names.

See Also

CBACK= graphics option (see "CBACK" on page 335)
"CBACK" on page 820

CLIP

Specifies whether clipping is on or off

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0

Default Value: OFF

Syntax

return-code-variable=GSET('CLIP', *status*);

Description

The GSET('CLIP', . . .)function activates or suppresses clipping around the current viewport.

Argument Definitions

status character string enclosed in quotation marks or character variable name; valid values are ON and OFF. When ON is used, the graphics elements outside of the specified viewport are not displayed. If you turn clipping OFF, the graphics elements outside of the defined viewport are displayed.

See Also

"CLIP" on page 820
"VIEWPORT" on page 850
"VIEWPORT" on page 904

COLREP

Associates a color name with a certain color index

Operating States: SGOP

Return Codes: 0, 4, 86

Default Values: 1. color list of COLORS= graphics option; 2. device's default color list

Syntax

return-code-variable=GSET('COLREP', *color-index, color*);

Description

The GSET('COLREP', . . .)function associates a predefined SAS color name with a color index. Many of the GASK routines and GSET functions use *color-index* as an argument.

If this function is not used, DSGI searches for a color specification in the following order:

1. the *n*th color in the color list of the COLORS= graphics option
2. the *n*th color in the device's default color list.

Argument Definitions

color-index numeric constant or numeric variable name; a number from 1 to 256 that identifies a color.

color character string enclosed in quotation marks or character variable name; a predefined SAS color name. See "SAS Color Names and RGB Values in the SAS Registry" on page 175 for a list of predefined SAS color names.

See Also

COLORS= graphics option (see "COLORS" on page 340)

"COLINDEX" on page 821
"COLREP" on page 822

DEVICE

Specifies the output graphics device

Operating States: GKCL

Return Codes: 0, 1

Default Value: 1. DEVICE= graphics option, if specified; 2. value entered in DEVICE prompt window; 3. value entered in OPTIONS window

Syntax

return-code-variable=GSET('DEVICE', *device*);

Description

The GSET('DEVICE', . . .)function selects the device driver.

Argument Definitions

device character string enclosed in quotation marks or character variable name; the name of the driver you are using. *Device* must match one of the device entries in the SASHELP.DEVICES catalog or one of your personal device catalogs, GDEVICE0.DEVICES through GDEVICE9.DEVICES. Refer to "Device Catalogs" on page 1126 for more information about catalogs that store device entries.

See Also

DEVICE= graphics option (see "DEVICE" on page 348)
"DEVICE" on page 823

FILCOLOR

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

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 85

Default Value: 1

Syntax

return-code-variable=GSET('FILCOLOR', *color-index*);

Description

The GSET('FILCOLOR', . . .)function selects the color index of the color used to draw fill areas. The aspect source flag (ASF) of FILCOLOR must be set to 'INDIVIDUAL' for this attribute to be used outside of a fill bundle.

DSGI searches for a color to assign to the index in the following order:

1. the color specified for the index in a GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list found in the device entry.

Argument Definitions

color-index	numeric constant or numeric variable name; indicates the index of the color to be used. Valid values are 1 to 256, inclusive.

See Also

COLORS= graphics option (see "COLORS" on page 340)

"ASF" on page 872

"COLREP" on page 876

"FILCOLOR" on page 823

"FILREP" on page 879

FILINDEX

Specifies the index of the bundle of fill area attributes

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 75

Default Value: 1

Syntax

return-code-variable=GSET('FILINDEX', *index*);

Description

The GSET('FILINDEX', . . .)function activates a particular fill bundle. To use the bundled values when the affected graphics element is drawn; the aspect source flag (ASF) for FILCOLOR, FILSTYLE, and FILTYPE must be set to 'BUNDLED'.

Argument Definitions

index	numeric constant or numeric variable name; specifies the index number of the fill bundle. Valid values are 1 to 20, inclusive.

See Also

"FILINDEX" on page 824
"ASF" on page 872
"FILREP" on page 879

FILREP

Associates a bundle of fill attributes with an index

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 75, 78, 85
Default Value: none

Syntax

return-code-variable =*GSET*('FILREP', *index, color-index, interior, style-index*);

Description

The GSET('FILREP', . . .)function assigns a color, type of interior, and style of the interior to a specific fill bundle. To use the bundled values when the affected graphics element is drawn; the aspect source flag (ASF) for FILCOLOR, FILTYPE, and FILSTYLE must be set to 'BUNDLED'.

Argument Definitions

index numeric constant or numeric variable name; indicates the index to be used with the bundle. Valid values are 1 to 20, inclusive. If *index* is expressed as a variable, the variable name must be initialized to a value between 1 and 20.

color-index numeric constant or numeric variable name; indicates the index of the color to be used. Valid values are 1 to 256, inclusive. The color index should represent one of the following:

- a color index assigned with the GSET('COLREP', . . .)function
- the *n*th color in the color list of the COLORS= graphics option
- the *n*th color in the device's default color list.

interior character string enclosed in quotation marks or character variable name; indicates the type of interior. Valid values are

- HATCH
- HOLLOW
- PATTERN
- SOLID.

style-index numeric constant or numeric variable name; indicates the index of the style to be used. Valid values are 1 to 15, inclusive, when FILTYPE is PATTERN, or 1 to 60, inclusive, when FILTYPE is

HATCH. See the GSET('FILSTYLE', . . .)function"FILSTYLE" on page 880 for a table of the patterns used for each style index. If *interior* is HOLLOW or SOLID, *style-index* is ignored.

See Also

"FILREP" on page 825
"ASF" on page 872
"COLREP" on page 876
"FILCOLOR" on page 877
"FILINDEX" on page 878
"FILSTYLE" on page 880
"FILTYPE" on page 881

FILSTYLE

Specifies the style of the interior of the fill area when the FILTYPE is PATTERN or HATCH

Operating State: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 78
Default Value: 1

Syntax

return-code-variable=GSET('FILSTYLE', *style-index*);

Description

The GSET('FILSTYLE', . . .)function activates a particular fill pattern when FILTYPE is specified as either PATTERN or HATCH. The aspect source flag (ASF) must be set to 'INDIVIDUAL' for this attribute to be used outside of a fill bundle.

Table 32.1 Style Index Table

Value	PATTERN	HATCH	Value	PATTERN	HATCH
1	X1	M1X	31		M3N045
2	X2	M1X030	32		M3N060
3	X3	M1X045	33		M3N090
4	X4	M1X060	34		M3N120
5	X5	M1N	35		M3N135
6	L1	M1N030	36		M3N150
7	L2	M1N045	37		M4X
8	L3	M1N060	38		M4X030
9	L4	M1N090	39		M4X045

Value	PATTERN	HATCH	Value	PATTERN	HATCH
10	L5	M1N120	40		M4X060
11	R1	M1N135	41		M4N
12	R2	M1N150	42		M4N030
13	R3	M2X	43		M4N045
14	R4	M2X030	44		M4N060
15	R5	M2X045	45		M4N090
16		M2X060	46		M4N120
17		M2N	47		M4N135
18		M2N030	48		M4N150
19		M2N045	49		M5X
20		M2N060	50		M5X030
21		M2N090	51		M5X045
22		M2N120	52		M5X060
23		M2N135	53		M5N
24		M2N150	54		M5N030
25		M3X	55		M5N045
26		M3X030	56		M5N060
27		M3X045	57		M5N090
28		M3X060	58		M5N120
29		M3N	59		M5N135
30		M3N030	60		M5N150

Argument Definitions

style-index numeric constant or numeric variable name. Valid values are 1 to 15, inclusive, when FILTYPE is PATTERN, or 1 to 60, inclusive, when FILTYPE is HATCH. See Table 31.1 on page 777 for value specifications.

See Also

"FILSTYLE" on page 826
"ASF" on page 872
"FILREP" on page 879
"FILTYPE" on page 881

FILTYPE

Specifies the type of the interior of the fill area

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 78
Default Value: HOLLOW

Syntax

return-code-variable=GSET('FILTYPE', *interior*);

Description

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

Argument Definitions

interior character string or character variable name; indicates the type of interior fill. Valid values are

- HATCH
- HOLLOW
- PATTERN
- SOLID.

See Also

"ASF" on page 872
"FILREP" on page 879
"FILSTYLE" on page 880

HPOS

Specifies the number of columns

Operating States: GKCL
Return Codes: 0, 1, 90, 307
Default Value: 1. HPOS= graphics option, if specified; 2. device's default HPOS setting

Syntax

return-code-variable=GSET('HPOS', *hpos*);

Description

The GSET('HPOS', . . .)function sets the number of columns in the graphics output area. GSET('HPOS', . . .)has the same effect as the HPOS= graphics option. See "HPOS" on page 383 for more information. You can reset the HPOS value by submitting one of the following statements:

```
goptions reset=goptions;
goptions reset=all;

goptions hpos=0;
```

Argument Definitions

hpos numeric constant or numeric variable name; specifies the number of horizontal columns; must be greater than 0.

See Also

"HPOS" on page 828

"HSIZE" on page 829

"VPOS" on page 851

HPOS= graphics option (see "HPOS" on page 383)

HSIZE

Specifies the horizontal dimension of the graphics output area

Operating States: GKCL

Return Codes: 0, 1, 90, 307

Default Value: 1. HSIZE= graphics option, if specified; 2. HSIZE device parameter

Syntax

return-code-variable=GSET('HSIZE', *hsize*);

Description

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

```
goptions reset=goptions;
goptions reset=all;

goptions hsize=0;
```

Argument Definitions

hsize numeric constant or numeric variable name; specifies the horizontal dimension, in inches, of the graphics output area; must be greater than 0.

See Also

"HSIZE" on page 829
"HPOS" on page 882
"VSIZE" on page 906
HSIZE= graphics option (see "HSIZE" on page 384)

HTML

Specifies the HTML string to invoke when an affected DSGI graphic element in a web page is clicked

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Default Value: null

Syntax

return-code-variable=GSET('HTML', '*string*');

Description

The GSET('HTML', . . .)function sets the HTML string to invoke when an affected DSGI graphic element in a web page is clicked. The HTML string is used with ODS processing to create a drill-down graph. The string value is used as the value for the HREF= attribute in the image map that implements the drill-down capability.

The value for *string* must be HREF= followed by a valid URL that is specified in double quotation marks, as in

```
rc = GSET("HTML", "HREF="http://www.sas.com/"");
```

The HTML string can be used by any of the following graphic element types drawn in the code: BAR, ELLIPSE, FILL, MARK, PIE, and TEXT. The string applies to all of these element types that are drawn *after* the string is set. To change the HTML string, set a new value. To turn off the HTML string, specify a null string:

```
rc = GSET("HTML", "");
```

Argument Definitions

string the HTML string. The string must be enclosed in single quotation marks and must begin with HREF= followed by a URL that is enclosed in double quotation marks.

See Also

"HTML" on page 830
"BAR" on page 857
"ELLIPSE" on page 859

"FILL" on page 860
"MARK" on page 862
"PIE" on page 864
"TEXT" on page 865

LINCOLOR

Specifies the color index of the color used to draw lines

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 85
Default Value: 1

Syntax

return-code-variable=GSET('LINCOLOR', *color-index*);

Description

The GSET('LINCOLOR', . . .)function selects the index of the color used to draw lines. The aspect source flag (ASF) for LINCOLOR must be set to 'INDIVIDUAL' for this attribute to be used outside of a line bundle.

DSGI searches for a color specification in the following order:

1. the color specified for the index in a GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list found in the device entry.

Argument Definitions

color-index numeric constant or numeric variable name; indicates the index of the color to use. Valid values are 1 to 256, inclusive.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"LINCOLOR" on page 830
"ASF" on page 872
"COLREP" on page 876
"LINREP" on page 886

LININDEX

Specifies the index of the bundle of line attributes

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 60
Default Value: 1

Syntax

return-code-variable=GSET('LININDEX', *index*);

Description

The GSET('LININDEX', . . .)function activates a particular line bundle. To use the bundled values when the affected graphics element is drawn; the aspect source flag (ASF) for LINCOLOR, LINTYPE, and LINWIDTH must be set to 'BUNDLED'.

Argument Definitions

index numeric constant or numeric variable name; indicates the index of the bundle to activate. Valid values are 1 to 20, inclusive.

See Also

"LININDEX" on page 831
"ASF" on page 872
"LINREP" on page 886

LINREP

Associates a bundle of line attributes with an index

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 60, 62, 85, 90
Default Value: none

Syntax

return-code-variable=GSET ('LINREP',*index, color-index, width, type*);

Description

The GSET('LINREP', . . .)function assigns a color, width, and line type to a specific line bundle. To use the bundled values when the affected graphics element is drawn; the aspect source flag (ASF) for LINCOLOR, LINWIDTH, and LINTYPE must be set to 'BUNDLED'.

Argument Definitions

index numeric constant or numeric variable name; indicates the number for the bundle to use as an index. Valid values are 1 and 20,

inclusive. If *index* is expressed as a variable, the variable must be initialized to a value between 1 and 20.

color-index numeric constant or numeric variable name; specifies the index of the color to use. Valid values are 1 to 256, inclusive. The color index should represent one of the following:

- a color index assigned with the GSET('COLREP', . . .)function
- the *n*th color in the color list of the COLORS= graphics option
- the *n*th color in the device's default color list.

width numeric constant or numeric variable name; indicates the width of the line; must be greater than 0.

type numeric constant or numeric variable name; indicates the type of line. Valid values are 1 to 46, inclusive. See Figure 14.22 on page 277 for representations of the different line types.

See Also

"ASF" on page 872
"COLREP" on page 876
"LINCOLOR" on page 885
"LININDEX" on page 885
"LINREP" on page 886
"LINTYPE" on page 887
"LINWIDTH" on page 888

LINTYPE

Specifies the line type

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 62
Default Value: 1

Syntax

return-code-variable=GSET('LINTYPE', *type*);

Description

The GSET('LINTYPE', . . .)function selects a line type. See Figure 14.22 on page 277 for representations of the different line types. The aspect source flag (ASF) for LINTYPE must be set to 'INDIVIDUAL' for this attribute to be used outside of a line bundle.

Argument Definitions

type numeric constant or numeric variable name; indicates the type of line to use. Valid values are 1 to 46, inclusive.

See Also

"LINTYPE" on page 833
"ASF" on page 872
"LINREP" on page 886

LINWIDTH

Specifies the thickness of the line

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 90
Default Value: 1

Syntax

return-code-variable=GSET('LINWIDTH', *width*);

Description

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

Argument Definitions

width numeric constant or numeric variable name; specifies the width of the line in pixels; must be greater than 0.

See Also

"LINWIDTH" on page 834
"ASF" on page 872
"LINREP" on page 886

MARCOLOR

Specifies the color index of the color used to draw markers

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 85
Default Value: 1

Syntax

return-code-variable=GSET('MARCOLOR', *color-index*);

Description

The GSET('MARCOLOR', . . .)function selects the color index of the color used to draw markers. The aspect source flag (ASF) of MARCOLOR must be set to 'INDIVIDUAL' for this attribute to be used outside of a marker bundle.

DSGI searches for a color specification in the following order:

1. the color specified for the index in a GSET('COLREP', . . .)function
2. the *n*th color in the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list found in the device entry.

Argument Definitions

color-index numeric constant or numeric variable name; indicates the index of the color to use. Valid values are 1 to 256, inclusive.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"MARCOLOR" on page 834
"ASF" on page 872
"COLREP" on page 876
"MARREP" on page 890

MARINDEX

Specifies the index of the bundle of marker attributes

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 64
Default Value: 1

Syntax

return-code-variable=GSET('MARINDEX', *index*);

Description

The GSET('MARINDEX', . . .)function activates the marker bundle indicated by *index*. The aspect source flag (ASF) for MARCOLOR, MARTYPE, and MARSIZE must be set to 'BUNDLED' before the GDRAW('MARK', . . .)function is executed if you want the bundled values to be used when the marker is drawn.

Argument Definitions

index numeric constant or numeric variable name; the number of the bundle to activate. Valid values are 1 to 20, inclusive.

See Also

"MARINDEX" on page 835
"ASF" on page 872
"MARREP" on page 890

MARREP

Associates a bundle of marker attributes with an index

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 64, 66, 85, 90
Default Value: none

Syntax

return-code-variable=GSET ('MARREP',*index, color-index, size, type*);

Description

The GSET('MARREP', . . .)function assigns a color, size, and type of marker to a specific marker bundle. The aspect source flag (ASF) of MARCOLOR, MARSIZE, and MARTYPE must be set to 'BUNDLED' before the GDRAW('MARK', . . .)function is executed if you want the bundled values to be used when the marker is drawn.

Argument Definitions

index — numeric constant or numeric variable name; defines the bundle index number. Valid values are 1 to 20, inclusive.

color-index — numeric constant or numeric variable name; indicates the color index of the color to use. Valid values are 1 to 256, inclusive. The color index should represent one of the following:

- a color index assigned to a color name with the GSET('COLREP', . . .)function
- the *n*th color in the color list of the COLORS= graphics option
- the *n*th color in the device's default color list.

size — numeric constant or numeric variable name; indicates the size of the marker in units of the current window system; must be greater than 0.

type — numeric constant or numeric variable name; specifies the type of marker to use; valid values are 1 to 67, inclusive. See Table 32.2 on page 892 for a table of the symbols used for each marker type.

See Also

"ASF" on page 872

"COLREP" on page 876
"MARCOLOR" on page 888
"MARINDEX" on page 889
"MARSIZE" on page 891
"MARTYPE" on page 891

MARSIZE

Selects the size of markers

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 90
Default Value: 1

Syntax

return-code-variable=GSET('MARSIZE', *size*);

Description

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

Argument Definitions

size numeric constant or numeric variable name; indicates the size of the marker in units of the current window system; must be greater than 0.

See Also

"MARSIZE" on page 837
"ASF" on page 872
"MARREP" on page 890

MARTYPE

Selects the kind of markers

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 66
Default Value: 1

Syntax

return-code-variable=GSET('MARTYPE', *type*);

Description

The GSET('MARTYPE', . . .)function determines the type of marker drawn. See Figure 14.21 on page 271 for representations of the symbols described in Table 32.2 on page 892. The aspect source flag (ASF) of MARTYPE must be set to 'INDIVIDUAL' for this attribute to be used outside of a marker bundle.

Table 32.2 Symbol Indexes Used with DSGI

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

Argument Definitions

type numeric constant or numeric variable name; indicates the index of the marker to draw. Valid values are 1 to 67, inclusive. See Table 32.2 on page 892 for value specifications.

See Also

"MARTYPE" on page 837
"ASF" on page 872
"MARREP" on page 890

MESSAGE

Specifies whether the interface error message system is enabled or disabled

Operating States: All
Return Codes: 0
Default Value: ON

Syntax

return-code-variable=GSET('MESSAGE', *status*);

Description

The GSET('MESSAGE', . . .)function activates or suppresses automatic error logging.

Argument Definitions

status character string enclosed in quotation marks or character variable name; indicates whether messages should be displayed. Valid values are ON and OFF. When ON is used, messages are automatically generated by the DSGI based on the return code from the function. If you set MESSAGE to OFF, no messages are automatically printed. You can do this to print custom messages for your application, or custom error messages.

See Also

"MESSAGE" on page 863
"GPRINT" on page 814

PATREP

Specifies the pattern name of a style index for a particular fill type.

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 79
Default value: 1

Syntax

return-code-variable=CALL GSET('PATREP', *index, pattern-name, hatch-name*);

Description

The GSET('PATREP', . . .)function sets a pattern of a style index for a particular fill type.

Argument Definitions

index	numeric variable name; indicates the index of the pattern to be used.
pattern-name	character variable name; sets the name of the pattern at the specified index.
hatch-name	character variable name; sets the name of the hatch at the specified index.

See Also

"PATREP" on page 840

TEXALIGN

Specifies the horizontal and vertical alignment of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Default values: *halign*=NORMAL, *valign*=NORMAL

Syntax

return-code-variable=GSET('TEXALIGN', *halign, valign*);

Description

The GSET('TEXALIGN', . . .)function sets a particular type of horizontal and vertical alignment for text strings. Figure 32.5 on page 895 illustrates *halign*.

Figure 32.5 Halign Values

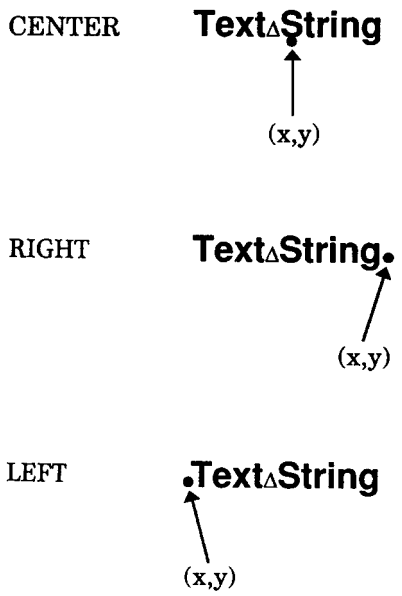


Figure 32.6 on page 895 illustrates *valign*.

Figure 32.6 Valign Values

Argument Definitions

halign character string enclosed in quotation marks or character variable name. Valid values are

CENTER

LEFT

NORMAL (the natural alignment based on the text path); alignment is chosen according to the following logic:

1. If TEXPATH is 'RIGHT', then NORMAL is 'LEFT'.
2. Otherwise, if TEXPATH is 'LEFT', then NORMAL is 'RIGHT'.
3. Otherwise, the text string is centered.

RIGHT.

valign character string enclosed in quotation marks or character variable name. Valid values are

BASE (alignment based on the baseline of the text string)

BOTTOM (alignment based on the bottom of the text string)

HALF (alignment based on the vertical midpoint of the string)

NORMAL (natural alignment based on the text path); alignment is chosen according to the following logic:

1. If TEXPATH is 'RIGHT' or TEXPATH is 'LEFT', then NORMAL is 'BASE'.
2. Otherwise, if TEXPATH is 'UP', then NORMAL is 'BOTTOM'.
3. Otherwise, if TEXPATH is 'DOWN', then NORMAL is 'TOP'.

TOP (alignment based on the top of the string).

See Also

"TEXALIGN" on page 841

"TEXT" on page 865

"TEXPATH" on page 899

"TEXUP" on page 901

TEXCOLOR

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

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 85

Default Value: 1

Syntax

return-code-variable=GSET('TEXCOLOR', *color-index*);

Description

The GSET('TEXCOLOR', . . .)function selects the color for text. The aspect source flag (ASF) of TEXCOLOR must be set to 'INDIVIDUAL' for this attribute to be used outside of a text bundle.

The value of GSET('TEXCOLOR', . . .)can be used in a text bundle. See the "TEXREP" on page 900 for information on how to define a text bundle.

DSGI searches for a color specification in the following order:

1. the color specified for the index in a GSET('COLREP', . . .)function
2. the *n*th color from the color list of the COLORS= graphics option
3. the *n*th color in the device's default color list found in the device entry.

Argument Definitions

color-index numeric constant or numeric variable name; indicates the color index of the color to be used. Valid values are 1 to 256, inclusive.

See Also

COLORS= graphics option (see "COLORS" on page 340)

"TEXCOLOR" on page 842

"ASF" on page 872

"COLREP" on page 876

"TEXREP" on page 900

TEXFONT

Specifies the font used to draw text strings

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8

Default values: 1. FTEXT= graphics option, if specified; 2. device-resident font, if possible; 3. SIMULATE font

Syntax

return-code-variable=GSET('TEXFONT', *font*);

Description

The GSET('TEXFONT', . . .)function selects a SAS/GRAPH font for the text. The aspect source flag (ASF) of TEXFONT must be set to 'INDIVIDUAL' for this attribute to be used outside of a text bundle. See "SAS/GRAPH Font Lists" on page 1644 for a list of valid SAS/GRAPH fonts. You can also use fonts you have created using the GFONT procedure.

Argument Definitions

font character string enclosed in quotation marks or character variable name; the name of a font that can be accessed by SAS/GRAPH software. If you want to use the device-resident font, submit

```
rc=gset("texfont", " ");
```

When DSGI is used with long font names, the font name must be in double quotation marks that are embedded in single quotation marks.

See Also

FTEXT= graphics options (see "FTEXT" on page 363)

"TEXFONT" on page 844

"ASF" on page 872

"TEXREP" on page 900

TEXHEIGHT

Specifies the character height of the text string

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 73

Default Value: 1. HTEXT= graphics option, if specified; 2. 1 unit

Syntax

return-code-variable=GSET('TEXHEIGHT', *height*);

Description

The GSET('TEXHEIGHT', . . .)function sets the height for text. GSET('TEXHEIGHT', . . .)affects text the same way as the HTEXT= graphics option.

Argument Definitions

height	numeric constant or numeric variable name; indicates height in units based on the current window system; must be greater than 0.

See Also

"TEXHEIGHT" on page 845

HTEXT= graphics options (see "HTEXT" on page 385)

TEXINDEX

Specifies the index of the bundle of text attributes

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 68

Default Value: 1

Syntax

return-code-variable=GSET('TEXINDEX', *index*);

Description

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

Argument Definitions

index numeric constant or numeric variable name; indicates the number of the bundle to activate. Valid values are 1 to 20, inclusive.

See Also

"TEXINDEX" on page 845
"ASF" on page 872
"TEXREP" on page 900

TEXPATH

Specifies the direction of the text string

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8
Default Value: RIGHT

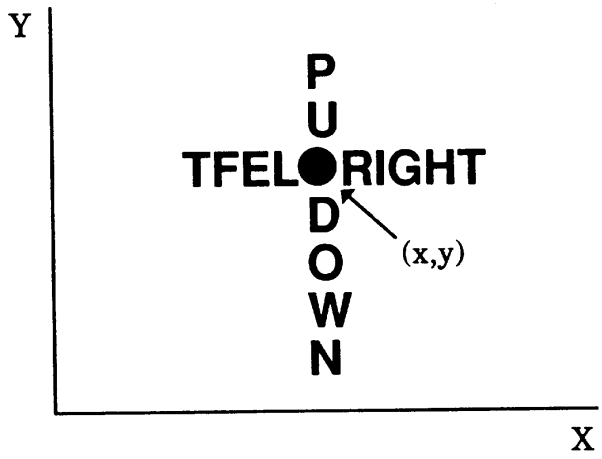
Syntax

return-code-variable=GSET('TEXPATH', *path*);

Description

The GSET('TEXPATH', . . .)function selects a particular type of text path. Text path determines the direction in which the text string reads. Figure 32.7 on page 899 illustrates the text paths that can be used with DSGI.

Figure 32.7 TEXPATH Values



Argument Definitions

path character string enclosed in quotation marks or character variable name; specifies the direction the text is read. Valid values:

- DOWN
- LEFT
- RIGHT
- UP.

See Also

"TEXPATH" on page 846
"TEXT" on page 865
"TEXALIGN" on page 894
"TEXUP" on page 901

TEXREP

Associates a bundle of text attributes with an index

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 68, 85

Default Value: none

Syntax

return-code-variable=GSET ('TEXREP',*index, color-index, font*);

Description

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

Argument Definitions

index numeric constant or numeric variable name; specifies the number to use as an index for the bundle; valid values are 1 to 20, inclusive. If *index* is expressed as a variable, the variable must be initialized to a value between 1 and 20.

color-index numeric constant or numeric variable name; indicates the color to use; valid values are 1 to 256, inclusive. The color index should represent one of the following:

- a color index assigned with the GSET('COLREP', . . .)function
- the *n*th color in the color list of the COLORS= graphics option
- the *n*th color in the device's default color list.

font character string enclosed in quotation marks or character variable name; names the font to use with the bundle. See "SAS/GRAPH

Font Lists" on page 1644 for a list of valid SAS/GRAPH fonts. You can also use fonts you have created using the GFONT procedure.

See Also

COLORS= graphics option (see "COLORS" on page 340)
"TEXREP" on page 847
"ASF" on page 872
"COLREP" on page 876
"TEXINDEX" on page 898

TEXUP

Specifies the orientation (angle) of the text string

Operating States: GKOP, SGOP, WSAC, WSOP
Return Codes: 0, 8, 74
Default Values: *upx*=0, *upy*=1

Syntax

return-code-variable=GSET('TEXUP',*upx, upy*);

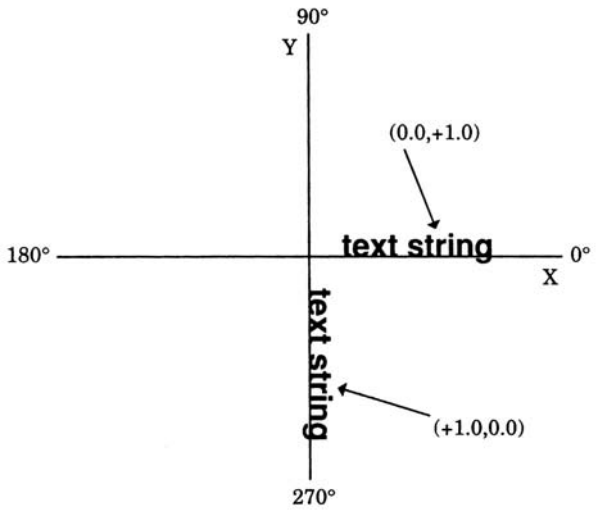
Description

The GSET('TEXUP', . . .)function sets the angle of the text string. DSGI uses the values of character up vectors to determine the angle of a text string. The character up vector has two components, *upx* and *upy*, that describe the angle at which the text string is placed. The angle is calculated with the following formula:

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

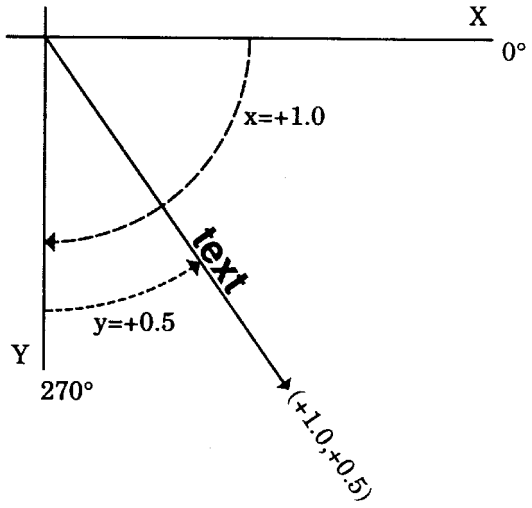
Effectively, when DSGI is calculating the angle for the text, it uses *upx* and *upy* as forces that are pushing the string toward an angle. The natural angle of text in the *upx* direction is toward the 6 o'clock position. In the *upy* direction, text naturally angles at the 3 o'clock position. If *upx* is greater than *upy*, the text is angled toward 6 o'clock. If *upy* is greater than *upx*, the text is angled toward 3 o'clock. Figure 32.8 on page 902 shows the angle of text when the values for *upx* and *upy* are (0.0, 1.0) and (1.0, 0.0).

Figure 32.8 Natural Angle of Text



As you change the values of *upx* and *upy*, the coordinate that has the highest value is taken as the angle, and the lowest value as the offset. Figure 32.9 on page 902 shows the angle of text when the character up vector values (+1.0, +0.5) are used.

Figure 32.9 Varying the Angle of Text



You can use the following macro to convert angles measured in degrees to character up vectors:

```
%macro angle(x);
 if mod(&x, 180)=90 then do;
   if mod(&x,270) = 0 then
       xup = 1.0;
    else
       xup = -1.0;
    rc = gset("texup", xup, 0.0);
    end;
```

```
 else do;
    b = mod(&x, 360);
    /* adjust y vector for 2nd and 3rd quadrants */
    if b > 90 and b lt 270 then
      yup = -1.0;
    else
      yup = 1.0;
    a=&x*1.7453292519943300e-002;
    xup = tan(-a);
    /* adjust x vector for 3rd quadrant */
    if b > 180 and b le 270 then
      xup = -xup;
    rc = gset("texup", xup, yup);
    end;
%mend angle;

data _null_;
  rc = ginit();
  rc = graph("clear", "angle");
  rc = gset("texalign", "left", "base");
  rc = gset("texheight", 5);
  rc = gset("texfont", "swissl");
  %angle(180);
  rc = gdraw("text", 50, 50, "180");
  %angle(80);
  rc = gdraw("text", 50, 50, "80");
  %angle(600);
  rc = gdraw("text", 50, 50, "600");
  rc = graph("update");
  rc = gterm();
  run;
```

Argument Definitions

upx numeric constant or numeric variable name; if *upy* is 0, *upx* cannot be 0.

upy numeric constant or numeric variable name; if *upx* is 0, *upy* cannot be 0.

See Also

"TEXUP" on page 848
"TEXT" on page 865
"TEXALIGN" on page 894
"TEXPATH" on page 899

TRANSNO

Specifies the number of the transformation to be used

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50

Default Value: 0

Syntax

return-code-variable=GSET('TRANSNO', *n*);

Description

The GSET('TRANSNO', . . .)function activates the viewport, or the window you have defined for the specified transformation number, or both. If you have not defined both a viewport and window for a transformation, the default is used for the one missing.

You can select 0 as the active transformation, but you cannot define a viewport or window for that transformation number. A transformation of 0 activates the default viewport, (0,0) to (1,1), and window, which is device dependent.

Argument Definitions

n	numeric constant or numeric variable name; indicates the viewport, or the window to activate, or both. Should correspond to the *n* used in the GSET('VIEWPORT', . . .)or GSET('WINDOW', . . .)functions, or both. Valid values are 0 to 20, inclusive.

See Also

"TRANS" on page 848
"TRANSNO" on page 850
"VIEWPORT" on page 850
"WINDOW" on page 853
"VIEWPORT" on page 904
"WINDOW" on page 907

VIEWPORT

Associates a viewport with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP

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

Default Values: *llx*=0, *lly*=0, *urx*=1, *ury*=1

Syntax

return-code-variable=GSET('VIEWPORT', *n, llx, lly, urx, ury*);

Description

The GSET('VIEWPORT', . . .)function defines a viewport and associates it with the transformation number, *n*. See the "TRANSNO" on page 903 for information on how to activate the viewport. See the "WINDOW" on page 907 for information on how to define a window to be used within the viewport.

Argument Definitions

n	numeric constant or numeric variable name; specifies the transformation number of the viewport. Valid values are 1 to 20, inclusive.
llx	numeric constant or numeric variable name; defines the *x* component of the lower-left corner of the viewport; must not exceed the value of *urx*; cannot be less than 0. Units are based on percent of the graphics output area.
lly	numeric constant or numeric variable name; defines the *y* component of the lower-left corner of the viewport; must not exceed the value of *ury*; cannot be less than 0. Units are based on percent of the graphics output area.
urx	numeric constant or numeric variable name; defines the *x* component of the upper-right corner of the viewport; cannot be greater than 1. Units are based on percent of the graphics output area.
ury	numeric constant or numeric variable name; defines the *y* component of the upper-right corner of the viewport; cannot be greater than 1. Units are based on percent of the graphics output area.

See Also

"VIEWPORT" on page 850
"WINDOW" on page 907
"TRANSNO" on page 903
"TRANSNO" on page 850
"TRANS" on page 848
"WINDOW" on page 853

VPOS

Specifies the number of rows

Operating States: GKCL

Return Codes: 0, 1, 90, 307

Default Values: 1. VPOS=graphics option, if specified; 2. device's default VPOS value

Syntax

return-code-variable=GSET('VPOS', *vpos*);

Description

The GSET('VPOS', . . .)function sets the number of rows in the graphics output area. GSET('VPOS', . . .)has the same effect on graphics output as the VPOS= graphics option.

You can reset the VPOS value by submitting one of the following statements:

```
goptions reset=goptions;
goptions reset=all;

goptions vpos=0;
```

Argument Definitions

vpos numeric constant or numeric variable name; specifies the number of rows in the graphics output area; must be greater than 0.

See Also

"VPOS" on page 851
"HPOS" on page 882
"VSIZE" on page 906
VPOS= graphics option (see "VPOS" on page 430)

VSIZE

Specifies the vertical dimension of the graphics output area

Operating States: GKCL

Return Codes: 0, 1, 90, 307

Default Values: 1. VSIZE= graphics option, if specified; 2. device's default VSIZE value

Syntax

return-code-variable=GSET('VSIZE', *vsize*);

Description

The GSET('VSIZE', . . .)function sets the vertical dimension, in inches, of the graphics output area. GSET('VSIZE', . . .)affects the dimensions of the default window.

You can reset the VSIZE value by submitting one of the following statements:

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

Argument Definitions

vsize numeric constant or numeric variable name; indicates the vertical dimension for the graph in inches; must be greater than 0.

See Also

"VSIZE" on page 852

"HSIZE" on page 883

"VPOS" on page 905

VSIZE= graphics option (see "VSIZE" on page 430)

WINDOW

Associates a window with a transformation number

Operating States: GKOP, SGOP, WSAC, WSOP

Return Codes: 0, 8, 50, 51

Default Values: *llx*=0, *lly*=0; *urx* and *ury* are device dependent

Syntax

return-code-variable=GSET ('WINDOW', *n, llx, lly, urx, ury*);

Description

The GSET('WINDOW', . . .)function defines a window and associates it with a transformation number. See the "TRANSNO" on page 903 for information on how to activate a window. See the "VIEWPORT" on page 904 for information on how to define a viewport for a window.

Argument Definitions

n	numeric constant or numeric variable name; specifies the transformation number of the window. Valid values are 1 to 20, inclusive.
llx	numeric constant or numeric variable name; defines the *x* component of the lower-left corner of the window; must not exceed the value of *urx*. Units are based on percent of the active viewport.
lly	numeric constant or numeric variable name; defines the *y* component of the lower-left corner of the window; must not exceed the value of *ury*. Units are based on percent of the active viewport.
urx	numeric constant or numeric variable name; defines the *x* component of the upper-right corner of the window. Units are based on percent of the active viewport.
ury	numeric constant or numeric variable name; defines the *y* component of the upper-right corner of the window. Units are based on percent of the active viewport.

See Also

"TRANS" on page 848
"TRANSNO" on page 850
"VIEWPORT" on page 850
"WINDOW" on page 853
"TRANSNO" on page 903
"VIEWPORT" on page 904

Return Codes for DSGI Routines and Functions

0	Function completed successfully.
1	DATA Step Graphics Interface should be in GKCL state; the statement is out of place within the DATA step.
3	DATA Step Graphics Interface should be in WSAC state; the statement is out of place within the DATA step.
4	DATA Step Graphics Interface should be in SGOP state; the statement is out of place within the DATA step.
7	DATA Step Graphics Interface should be in WSOP, WSAC, or SGOP state; the statement is out of place within the DATA step.
8	DATA Step Graphics Interface should be in GKOP, WSOP, WSAC, or SGOP state; the statement is out of place within the DATA step.
24	Workstation is open.
25	Workstation is not open.
26	Workstation cannot be opened.
29	Workstation is active.
30	Workstation is not active.
50	Invalid transformation number; transformation numbers must be in the range 0 to 20; viewports and windows cannot be defined for transformation 0.
51	Transformation is not a well-defined rectangle; transformations must have coordinates for four vertices.
52	Viewport coordinates are out of range; coordinates must be within dimensions of graphics output area for the device.
55	Clipping is on.
56	Clipping is off.
60	Bad line index; index numbers must be in the range 1 to 20.
61	No bundle defined for the line index; a GSET('LINREP', . . .)function has not been submitted for the referenced line index.
62	Line type is less than or equal to 0 or greater than 46; type must be in the range 1 to 46.

64	Invalid marker index; index numbers must be in the range 1 to 20.
65	No bundle defined for the polymarker index; a GSET('MARREP', . . .)function has not been submitted for the referenced marker index.
66	Marker type is less than or equal to 0 or greater than 67; type must be in the range 1 to 67.
68	Invalid text index; index numbers must be in the range 1 to 20.
69	No bundle defined for the text index; a GSET('TEXREP', . . .)function has not been submitted for the referenced text index.
73	Character height is less than or equal to 0; height must be greater than 0.
74	Both components of the character up vector are 0; both X and Y of a character up vector cannot be 0.
75	Invalid fill index; index numbers must be in the range 1 to 20.
76	No bundle defined for the fill index; a GSET('FILREP', . . .)function has not been submitted for the referenced fill index.
78	Style index is less than or equal to 0 or greater than 60; style indexes must be in the range of 1 to 60.
79	Invalid pattern index.
86	Invalid color index; color index is out of the range 1 to 256 or is not numeric.
87	No color name defined for the color index
90	Value is less than 0; value must be greater than or equal to 0.
150	External image file cannot be accessed. The image file either cannot be accessed, or the image file is in an unsupported format, or the image data is incomplete or otherwise corrupt.
301	Out of memory; your workstation does not have enough memory to generate the graph.
302	Out of room for graph; your device cannot display the size of the graph.
307	Error occurred in program library management; a GRAPH function did not execute properly.

See Also

Chapter 6, "Using Graphics Devices," on page 67
for information about specifying device drivers.

Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327
for descriptions of graphics options and device parameters

Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155
for information about the fonts available in SAS/GRAPH software

Chapter 12, "SAS/GRAPH Colors and Images," on page 167

for information about specifying colors in SAS/GRAPH programs

"GOPTIONS Statement" on page 220
for an explanation of setting graphics options with the GOPTIONS statement

"PATTERN Statement" on page 240
for information about specifying patterns with DSGI

"SYMBOL Statement" on page 252
for representations of the markers that can be used with DSGI

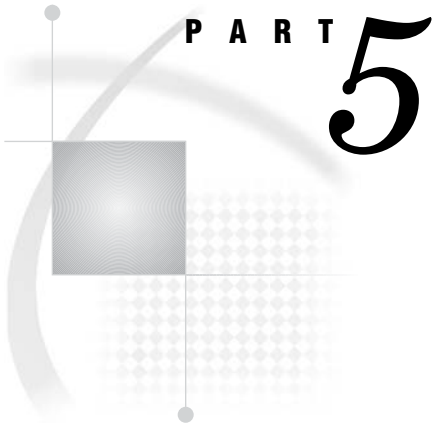
Chapter 31, "The DATA Step Graphics Interface," on page 769
for a complete explanation of using DSGI statements to produce graphs

Chapter 38, "The GDEVICE Procedure," on page 1125
for information about device entries

The discussion for ARRAY in *SAS Language Reference: Dictionary*
for an explanation of argument lists

References

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



PART 5

SAS/GRAPH Procedures

CHAPTER

33

The GANNO Procedure

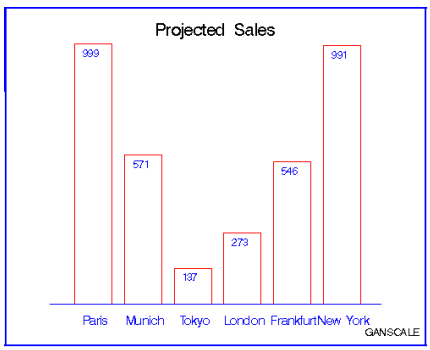
Overview

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 29, "Using Annotate Data Sets," on page 641.

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 will cause the procedure to scale the output to fit the available space. The GSLIDE procedure does not have this capability.

Figure 33.1 on page 913 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 on page 916.

Procedure Syntax

Requirements: An input Annotate data set is required.

Supports: Output Delivery System (ODS)

PROC GANNO ANNOTATE=*Annotate-data-set*
<DATASYS>
<DESCRIPTION='*description*'>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>
<NAME= '*entry-name*' | *variable-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='*entry-name*' | *variable-name*>;

Required Arguments

ANNOTATE=*Annotate-data-set*
ANNO=*Annotate-data-set*
specifies a data set that includes Annotate variables that identify graphics commands and parameters.

See also: Chapter 29, "Using Annotate Data Sets," on page 641

Options

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 display.

See also: "Using the DATASYS Option to Scale Graphs" on page 916

Featured in: Example 1 on page 916

DESCRIPTION='*description*'
specifies the description of the catalog entry for the chart. The maximum length is 256 characters. The description does not appear on the chart. By default, the GANNO procedure assigns the description OUTPUT FROM PROC GANNO.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window
- the catalog-entry properties that you can view from the Explorer window
- the Table of Contents that is generated when you use CONTENTS= on an ODS HTML statement, assuming that the procedure output is generated while the contents page is open
- the Description field of the PROC GREPLAY window
- the chart description for Web output (depending on the device driver). For more information, see "PROC GANNO Statement" on page 914.

Alias: DES=

Featured in: Example 2 on page 919

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 also:

Featured in: Example 2 on page 919

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map in an HTML output file. The information in the image map data set includes the shape and coordinates of the elements in the graph and drill-down URLs that have been associated with those elements. The drill-down URLs are provided by one or two variables in the input data set. These variables are identified to the GANNO procedure with the HTML= and/or HTML_LEGEND= options.

The %IMAGEMAP macro generates the image map in the HTML output file. The macro takes two arguments, the name of the image map data set and the name or fileref of the HTML output file, as shown in the following example:

```
%imagemap(imgmapds, myimgmap.html);
```

See also: Chapter 30, "Annotate Dictionary," on page 667 and "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

Featured in: Example 4 on page 925

NAME='*entry-name*' | *variable-name*
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to

lowercase, and periods are converted to underscores. The default GRSEG name is GANNO. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, GANNO1.

See also: "About Filename Indexing" on page 99

Featured in: Example 2 on page 919 Example 3 on page 921

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 on page 916.

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

Example 1: Scaling Data-Dependent Output

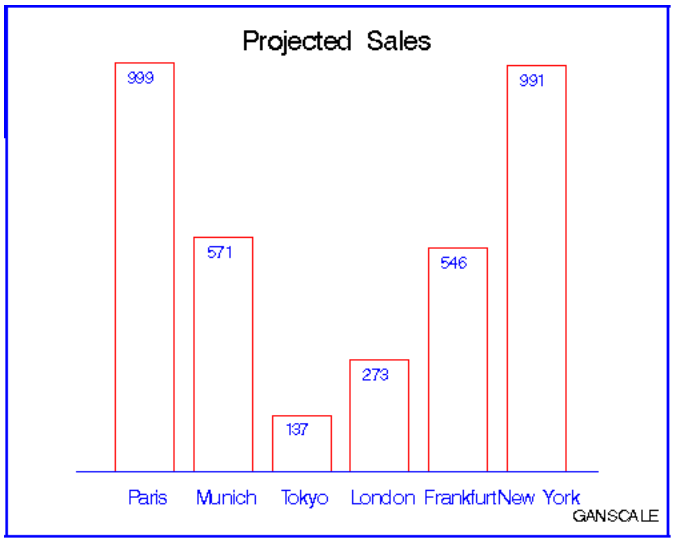
Procedure features:

PROC GANNO statement options:

ANNOTATE=
DATASYS

Sample library member: GANSCALE

Figure 33.2 Scaled GANNO Output



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.

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="swiss";
      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="swiss"; 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="swissb";
         size=5; color=" "; output;
         x=92; y=5; size=3; style="swiss"; text="GANSCALE"; output;
      function="frame"; color="blue"; when="b";
         style="empty"; output;
   end;
run;
```

Display the annotate graphics. The ANNOTATE= 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 display on the graph.

```
proc ganno annotate=wrldanno
   datasys;
run;
quit;
```

Example 2: Storing Annotate Graphics

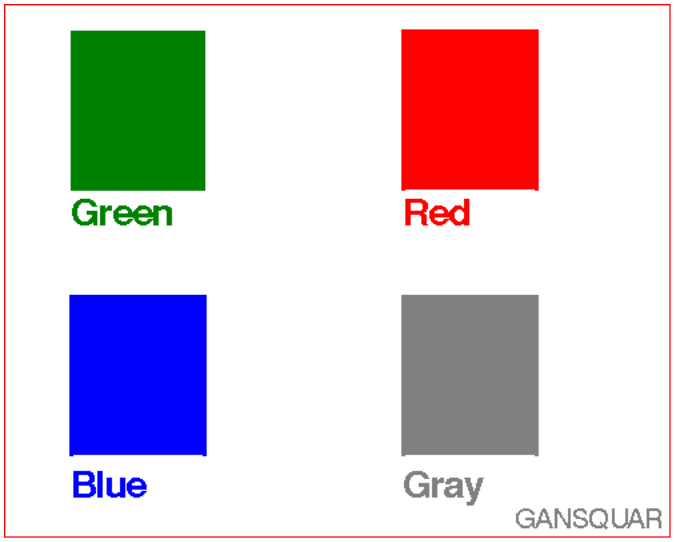
Procedure features:

PROC GANNO statement options:

DESCRIPTION=
GOUT=
NAME=

Sample library member: GANSQUAR

Figure 33.3 Four Squares



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.

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

Procedure features:

PROC GANNO statement option:

NAME=

Sample library member: GANMULTI

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 Figure 33.4 on page 923, Figure 33.5 on page 924, Figure 33.4 on page 923 and Figure 33.6 on page 924.

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 on page 919. 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.

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;
```

Figure 33.4 Output for COLOR Value BLUE (WORK.EXCAT.BLUE.GRSEG)

Figure 33.5 Output for COLOR Value GRAY (WORK.EXCAT.GRAY.GRSEG)

Figure 33.6 Output for COLOR Value GREEN (WORK.EXCAT.GREEN.GRSEG)

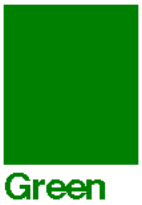
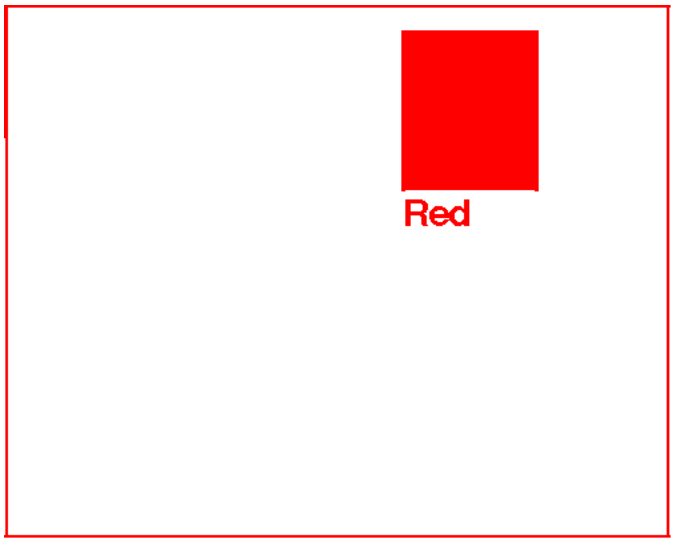


Figure 33.7 Output for COLOR Value RED (WORK.EXCAT.RED.GRSEG)



Example 4: Using Annotate Graphics in a Drill-Down Graph

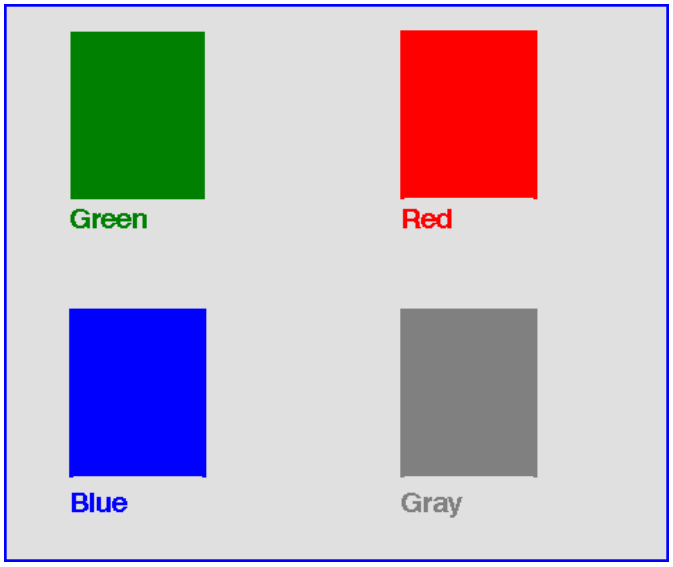
Procedure features:

PROC GANNO statement option:

IMAGEMAP=

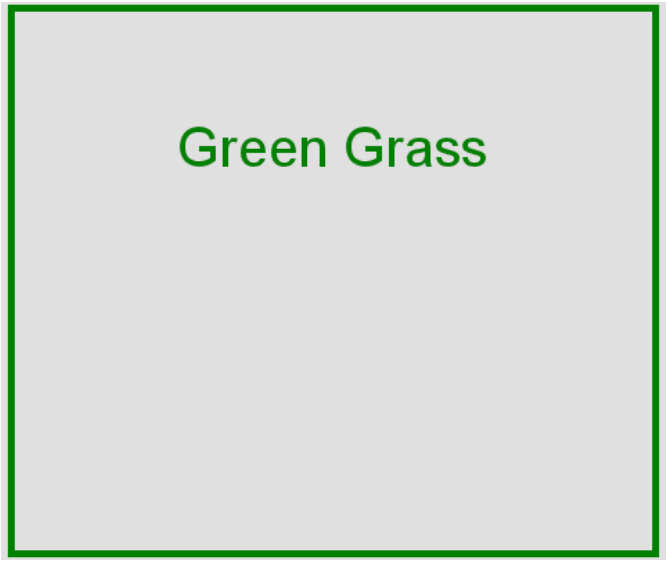
Sample library member: GANDRILL

This example creates essentially the same Annotate data set used in Example 2 on page 919. It draws four colored squares and displays the data set as a single graphics output.



However, this time the example shows you how to use Annotate graphics to generate a drill-down graph. The example uses the HTML variable in the Annotate data set to

specify linking information that defines each of the four squares as a hot zone. When the graph is viewed in a browser, you can click on a square to drill down to a related graph. For example, if you click on the green square, it drills down to a graph that confirms that you selected the green square.



The example uses the ODS HTML destination to generate the drill-down graph. To implement the drill-down capability, the Annotate data set uses the HTML variable to provide the linking information (see "HTML Variable" on page 709). The presence of the HTML variable in the Annotate data set and the IMAGEMAP= option on the GANNO procedure causes the ODS HTML destination to generate an image map for the graph in the HTML output.

The example runs four GSLIDE procedures to generate the target output. Each GSLIDE procedure uses the NAME= option to name the graph it produces, ensuring that the GIF driver creates files named green.gif, blue.gif, red.gif, and gray.gif. These are the files that are referenced as targets by the strings that are specified for the Annotate data set's HTML variable.

Allocate a storage location for all the output files, and set the graphics environment.DEV= specifies the GIF device. GOUTMODE=REPLACE specifies that the output graphics use the same filenames each time you run the program, instead of appending numbers to the filenames.

```
/* set the graphics environment */
goptions reset=all dev=gif gunit=pct goutmode=replace border;
```

Create the Annotate data set. The HTML variable is used to define the linking information for each square. Because the GSLIDE procedures that generate the target output use NAME= to ensure the output files are named green.gif, red.gif, blue.gif, and gray.gif, strings that reference those names are assigned to the HTML variable for the appropriate observation in the data. For the final observation, the HTML variable's value is set to a null string; otherwise it would retain the last assigned value, which is **href=gray.gif**. In that case, the graph's background area would be defined as a hot zone that links to file gray.gif. For a description of the other functions and variables used in the Annotate data set, see Example 2 on page 919.

```
/* create the Annotate data set */
data squares;
     length function style color $ 8
            html text $ 15;
     xsys="3"; ysys="3";

        /* draw the green square */
     color="green";
     function="move"; x=10; y=65; output;
     function="bar";  x=30; y=95; style="solid";
        html="href=green.gif"; output;

        /* label green square */
     function="label"; x=10; y=63; position="6";
        style="swissb"; size=2; text="Green"; output;

        /* draw the red square */
     color="red";
     function="move"; x=60; y=65; output;
     function="bar";  x=80; y=95;
        html="href=red.gif"; output;

        /* label red square */
     function="label"; x=60; y=63; position="6";
        style="swissb"; size=2; text="Red"; output;

        /* draw the blue square */
     color="blue";
     function="move"; x=10; y=15; output;
     function="bar";  x=30; y=45;
        html="href=blue.gif"; output;

        /* label blue square */
     function="label"; x=10; y=12; position="6";
        style="swissb"; size=2; text="Blue"; output;

        /* draw the gray square */
     color="gray";
     function="move"; x=60; y=15; output;
     function="bar";  x=80; y=45;
        html="href=gray.gif"; output;

        /* label gray square and add a footnote */
     function="label"; x=60; y=12; position="6";
        style="swissb"; size=2; text="Gray"; output;
```

```
        /* draw a blue frame */
     function="frame"; color="blue"; style="empty";
        /* set null link for background area in frame */
        html=""; output;
run;
```

Open the ODS HTML destination. BODY= specifies the filename for the HTML output. PATH specifies the path where the output graphics files and HTML files are created.

```
/* open the ODS HTML destination */
ods html body="gandrill.htm" path=".";
```

Generate the drill-down graph. IMAGEMAP= specifies ANNOMAP as the name for the Imagemap data set.

```
/* generate annotate graphics */
proc ganno annotate=squares
    imagemap=annomap
    description="Four squares";
run;
```

Generate the target output.PROC GSLIDE is run four times to generate the four graphs that will serve as target output for the links that are defined in the drill-down graph.

```
/* generate the target output */
proc gslide wframe=4
   cframe=green name="green";
   note height=20;
   note height=10
        justify=center color=green
       "Green Grass";
run;

proc gslide wframe=4
   cframe=blue name="blue";
   note height=20;
   note height=10
        justify=center color=blue
       "Blue Sky";
run;

proc gslide wframe=4
   cframe=red name="red";
   note height=20;
   note height=10
        justify=center color=red
       "Red Wine";
run;

proc gslide wframe=4
   cframe=gray name="gray";
```

```
   note height=20;
   note height=10
        justify=center color=gray
       "Gray Mare";
run;
quit;
goptions goutmode=append;
run;
```

CHAPTER

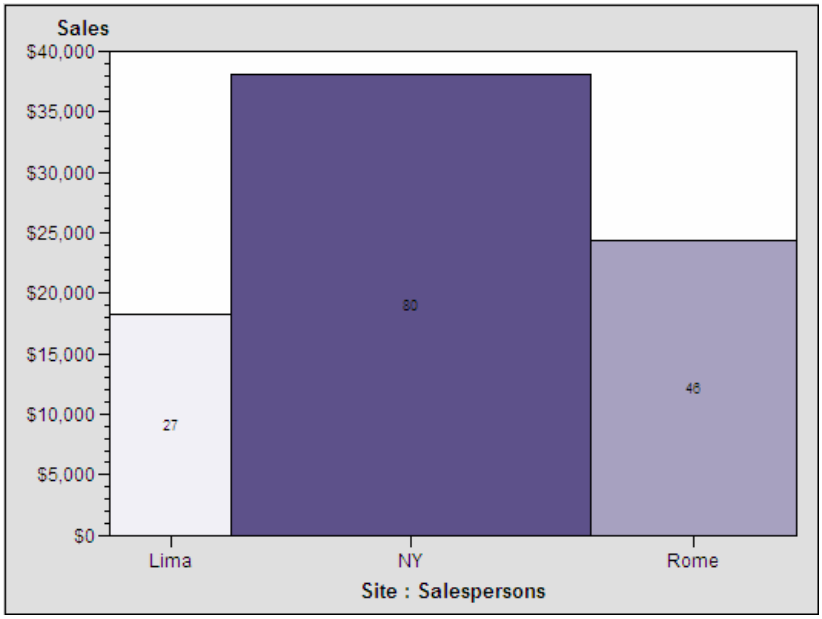
34

The GAREABAR Procedure

Overview

The GAREABAR procedure produces an area bar chart displaying two statistics for each category of data. 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 sales persons generating revenue for that site.

Display 34.1 Number of Sales Persons and Total Sales for Each Site



Concepts

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.

Examples using the SUBGROUP option are shown in “Examples” on page 937 and Example 4 on page 943.

Procedure Syntax

Requirements:

- a GOPTIONS statement with DEV=ACTIVEX or DEV=ACTXIMG
- an ODS statement to close the listing destination
- an ODS statement to open the output destination
- at least one HBAR, HBAR3D, VBAR, or VBAR3D statement
- a SUMVAR= option
- an ODS statement to close the output destination
- an ODS statement to open the listing destination.

Global statements: FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

Reminder: BY, FORMAT, LABEL, WHERE

Supports: Run-group processing, Activex, Actximg

PROC GAREABAR <DATA=*input-data-set*;>
HBAR | HBAR3D | VBAR | VBAR3D *chart-variable*width-variable* /
SUMVAR=*numeric-variable*<*(option(s)*> ;

PROC GAREABAR Statement

Identifies the data set containing the chart variables.

Requirements: An input data set is required.

Syntax

PROC GAREABAR<DATA=*input-data-set*>;

Options

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 Statements

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.

Requirements: One category variable, one width variable, and the SUMVAR= option variable.

Global statements: FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

Description

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.

Syntax

HBAR | HBAR3D | VBAR | VBAR3D *chart-variable*width-variable*
SUMVAR=*numeric-variable*</ *option(s)*>;

*option(s)*can be one or more options from any or all of the following categories:

- appearance options:
  - CFRAME=*background-color*
  - CTEXT=*text-color*
  - FRAME | NOFRAME
  - LEGEND=LEGEND<1...99>
  - NOLEGEND
- statistic options:
  - RESPONSESTAT=*statistic*
  - WIDTHSTAT=*statistic*
- midpoint options:
  - CONTINUOUS
  - DISCRETE
  - SUBGROUP=*subgroup-variable*

□ description options

DESCRIPTION=*"description"*

NAME=*"name"*

Required Arguments

The options in an HBAR, HBAR3D, VBAR, and 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.

category-variable

specifies the variable that defines the categories of data to chart. The CATEGORY variable can be either character or numeric. Each unique value of the category variable results in a separate bar.

sumvar=variable

specifies the variable that defines the height of each vertical bar or length of each horizontal bar. The SUMVAR= option variable is always numeric. The default statistic is sum.

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.

Options

The options in an HBAR, HBAR3D, VBAR, and VBAR3D statement affect all graphs 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 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, and so might not be measured accurately. The default is discrete.

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 on 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 attributes 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 descriptive text is displayed as follows:

- the description in the Results window
- the properties that you view from the Explorer window
- the Table of Contents that is generated when you use CONTENTS= on an ODS HTML statement, assuming the output is generated while the contents page is open
- 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: GAREABAR of*category 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.

Restriction: The LEGEND statement options are partially supported by ActiveX.

See also: "LEGEND Statement" on page 225

Restriction: Not supported by Java.

NAME=*"name"*
specifies the name of the graphics output file. The name can be up to 256 characters long. Uppercase characters are converted to lowercase. The default name is **`graph.png`**. If the name duplicates and existing name, then SAS/GRAPH adds a number to the name to create a unique name, for example, **`graph1.png`**.

Restriction: Not supported by Java.

See also: "About Filename Indexing" on page 99

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.

RESPONSESTAT= *SUM* |*PCT*| *PERCENT*
specifies the statistic for subgroups. The default is sum.

If the SUBGROUP= option is not specified, then the RESPONSESTAT= option is ignored.

Alias: RESPSTAT= or RSTAT=

Restriction: Not supported by Java.

SUBGROUP=*subgroup-variable*
divides the bars into segments according to the values of the *subgroup-variable*. The *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 each unique value of the subgroup variable.

Restriction: Not supported by Java.

SUMVAR=*summary-variable*
specifies the numeric variable for the sum calculation. The GAREABAR procedure calculates the sum of for each category to determine the length or height of each bar.

Restriction: Not supported by Java.

WIDTHSTAT= SUM | PCT | PERCENT
specifies whether the WIDTH= option statistic is a percent or a sum. The default statistic is sum.

Alias: WSTAT=SUM | PCT | PERCENT

Restriction: Not supported by Java.

Examples

Note: 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 HTML;` *`procedure-code`*`; ODS HTML CLOSE;`**). See "RUN-Group Processing" on page 56 for more information. △

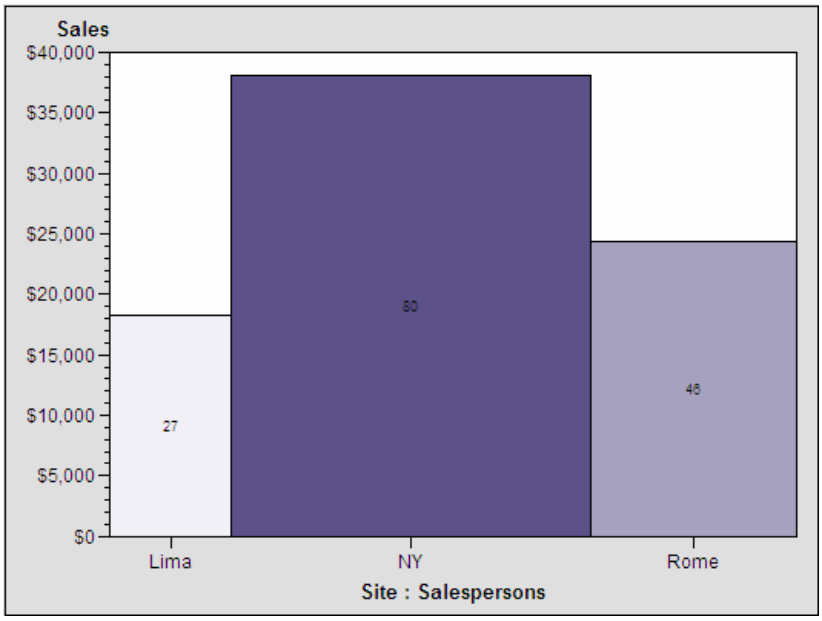
Example 1: Generating an Area Bar Chart

Procedure features: VBAR Statement

Data set: WORK.TOTALS

Sample library member: GABSUMVR

Figure 34.1 Area Bar Chart



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 sales persons.

Reset the graphics options. Set the device to output Activex..

```
goptions reset=all dev=activex;
```

Create the 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 listing destination..

```
ods listing close;
```

Open the HTML output destination..

```
ods html;
```

Run PROC GAREABAR with VBAR statement. The VBAR site statement creates a vertical bar for each value of site. *SALESPERSONS sets the width variable for bars. SUMVAR=SALES sets the height variable for each of the bars.

```
proc gareabar data=totals;
   vbar site*salespersons /
      sumvar=sales;
run;
quit;
```

Close HTML destination..

```
ods html close;
```

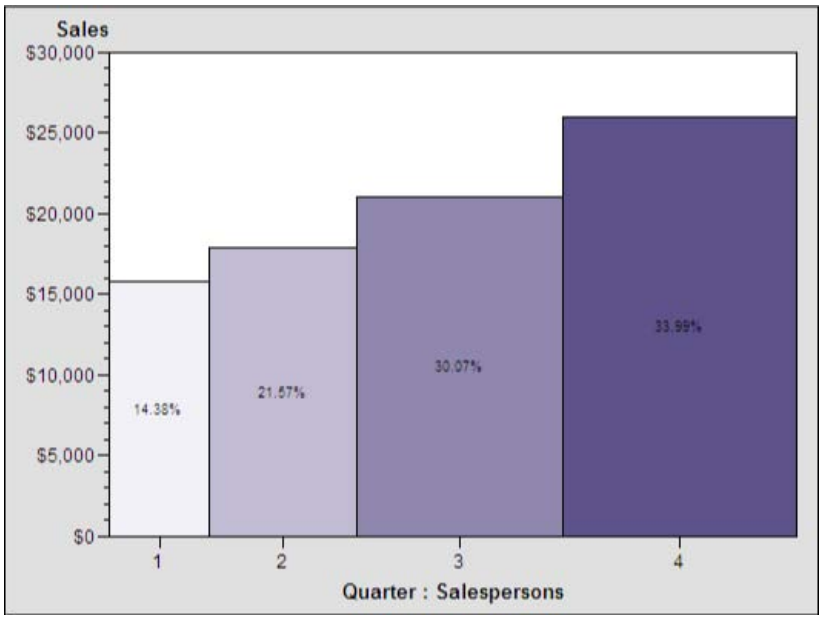
Open the listing destination..

```
ods listing;
```

Example 2: Generating an Area Bar Chart with a Numeric Chart Variable

Procedure features: VBAR Statement, SUMVAR=, WSTAT=
Data set: WORK.TOTALS
Sample library member: GABNUMVR

Figure 34.2 Area Bar Chart with Numeric Chart Variable (gabnumvr)



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 is 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.

Reset the graphics options. Set device to output ActiveX..

```
goptions reset=all dev=activex;
```

Create the 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 listing destination.

```
ods listing close;
```

Open the HTML output destination.

```
ods html;
```

Run PROC GAREABAR with VBAR statement. The VBAR=SITE option creates a vertical bar for each value of quarter. *SALESPERSONS sets the width of each of the bars. The SUMVAR=SALES option sets the height of each of the bars. WSTAT=PCT option sets the number of salespersons as a percentage of the whole.

```
proc gareabar data=totals;
   vbar quarter*salespersons/
      sumvar=sales
      wstat=pct;
run;
quit;
```

Close the HTML destination.

```
ods html close;
```

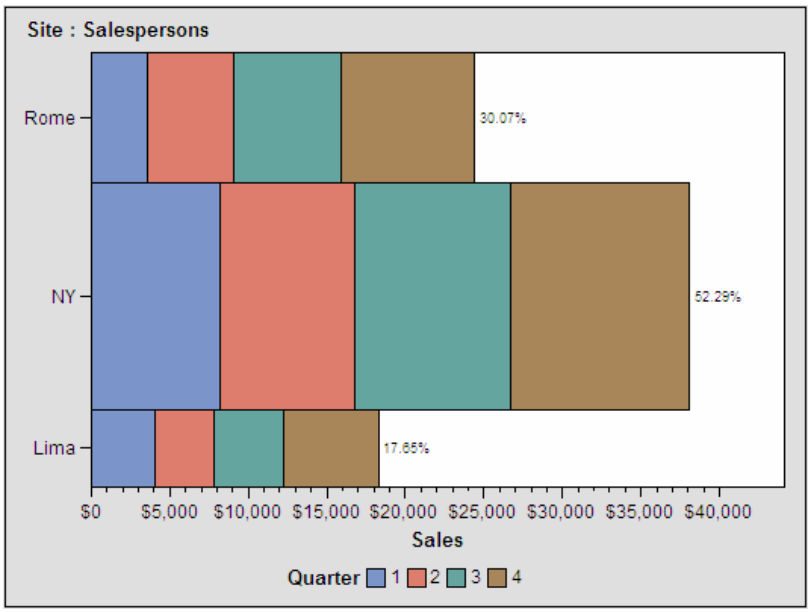
Open the listing destination.

```
ods listing;
```

Example 3: Generating an Area Bar Chart with Subgroups

Procedure features: HBAR Statement, SUBGROUP=, SUMVAR=, RSTAT=, WSTAT=
Data set: WORK.TOTALS
Sample library member: GABSUBGR

Figure 34.3 Area Bar Chart with Subgroups (gabsubgr)



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.

Reset the graphics options. Set the device to output ActiveX.

```
goptions reset=all dev=activex;
```

Create the 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 listing destination.

```
ods listing close;
```

Open the HTML output destination.

```
ods html;
```

Run PROC GAREABAR with an HBAR statement. The SUMVAR=SALES option sets the length of the bar.

The HBAR SITE*SALESPERSONS creates a horizontal bar for each site. SALESPERSONS is represented by the width of each bar. The WSTAT=PERCENT option sets the statistic to percentage 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 the HTML destination..

```
ods html close;
```

Open the listing destination..

```
ods listing;
```

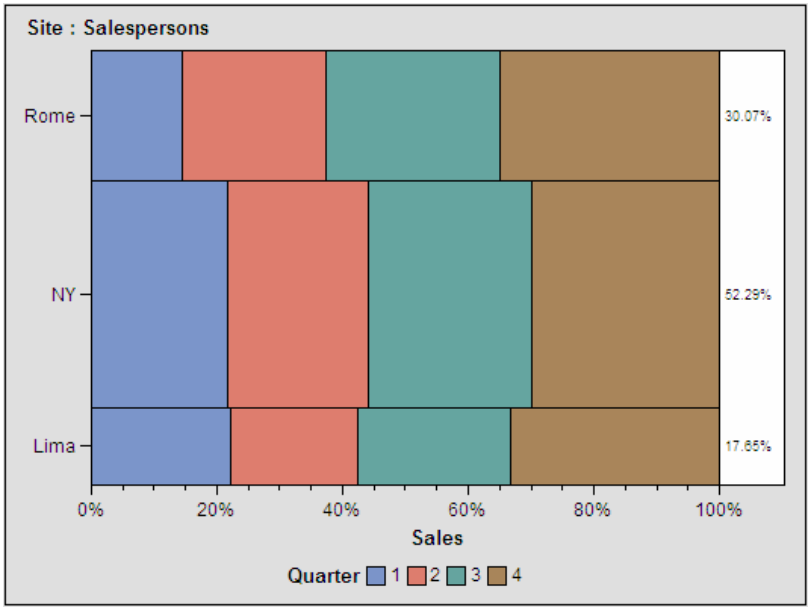
Example 4: Area Bar Chart with Subgroups; Using the RSTAT= option and the WSTAT= option to Calculate Statistics as Percentages

Procedure features: HBAR Statement, SUBGROUP=, RSTAT=, WSTAT=

Data set: WORK.TOTALS

Sample library member: GABWSTAT

Figure 34.4 Area Bar Chart with Subgroups and Percentage Statistics (gabwstat)



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.

Reset the graphics options. Set the device to output Activex.

```
goptions reset=all dev=activex;
```

Create the 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    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 listing destination.

```
ods listing close;
```

Open the HTML output destination destination.

```
ods html;
```

Run PROC GAREABAR with an HBAR statement. Because SITE*SALESPERSONS and WSTAT=PERCENT, 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 sales for each quarter is displayed as percentages along the horizontal axis.

```
proc gareabar data=totals;
   hbar site*salespersons /
      sumvar=sales
      subgroup=quarter
      rstat=PCT
      wstat=PCT;
run;
quit;
```

Close the HTML destination..

```
ods html close;
```

Open the listing destination..

```
ods listing;
```

CHAPTER

35

The GBARLINE Procedure

Overview

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, showing trends and patterns.

In conjunction with the SYMBOL statement, the GBARLINE procedure can produce needle plot overlays, and overlay plots with stepped interpolation.

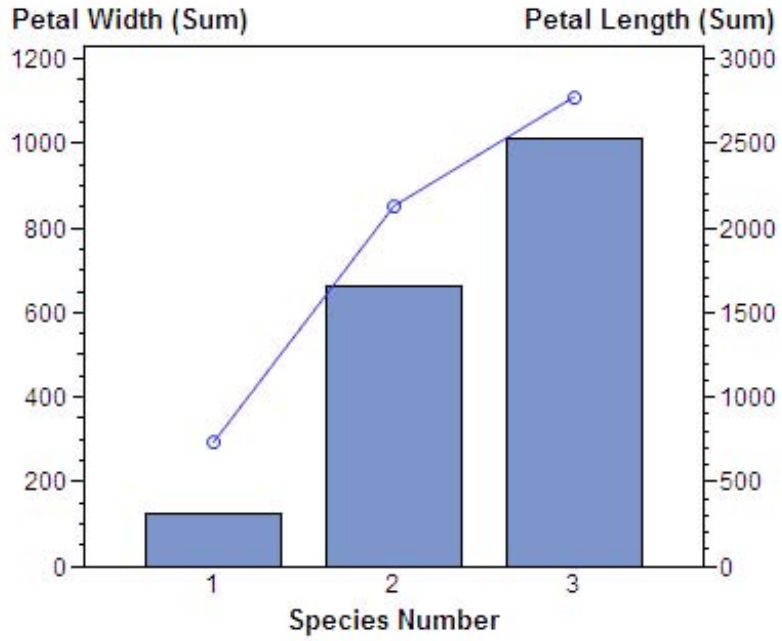
Note: PROC GBARLINE is not supported by Java. △

About Bar-Line Charts

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 35.1 on page 948 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

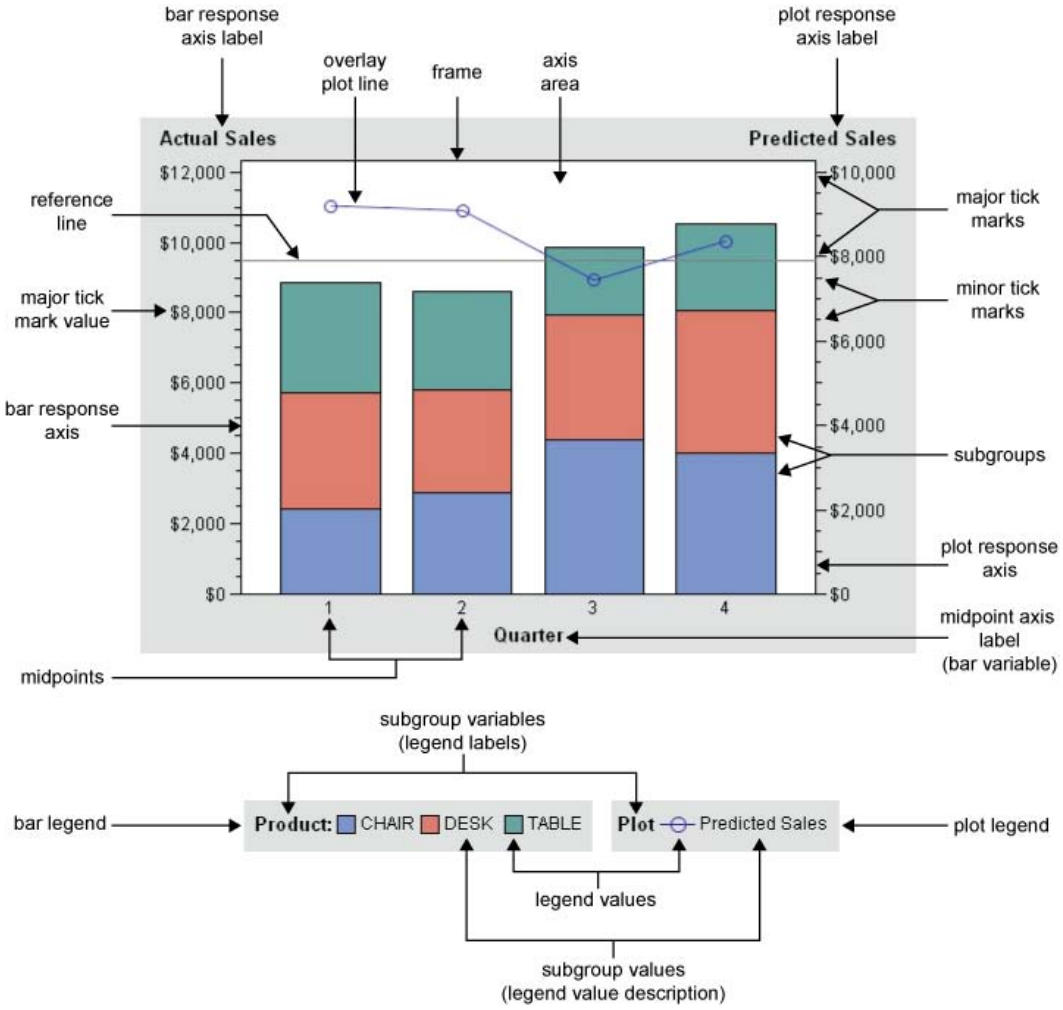


Concepts

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 chart uses the same chart variable and has an optional response variable (SUMVAR= option). A computed statistic can be set with the TYPE= option.

Figure 35.2 on page 949 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 contribute.

A character chart variable is always discrete.

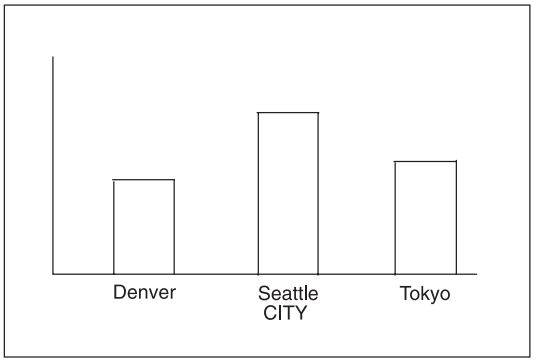
About Midpoints

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. 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

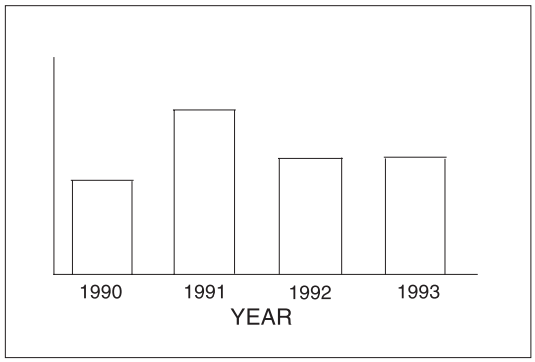


By default, character midpoints are arranged in alphabetic order. If a character variable has an associated format, then the values are arranged in order of the formatted values.

Discrete Numeric Values

A numeric chart variable used with the DISCRETE option generates a midpoint for each unique value of the chart variable. In the following example, the numeric variable YEAR used with the DISCRETE option produces one midpoint for each year:

Figure 35.4 Discrete Numeric Midpoints



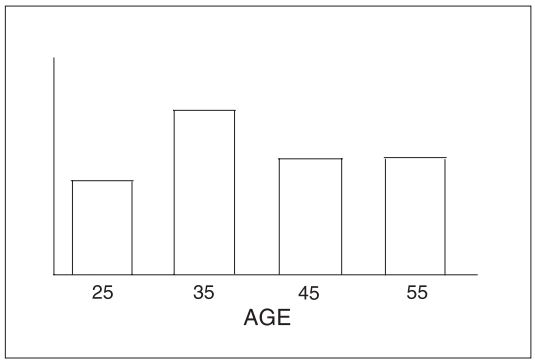
By default, numeric midpoints are arranged in ascending order of the chart variable. If the numeric variable has an associated format, then each formatted value generates a separate midpoint. Formatted numeric variables are arranged in ascending order according to their unformatted numeric values.

Continuous Numeric Values

A continuous numeric variable generates midpoints that represent ranges of values. By default, the 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 35.5 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 35.3 on page 950, 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, to control the range of values each midpoint represents, or to 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 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 description of the LEVELS= and MIDPOINTS= options.

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, with SUM being 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 953. See also the descriptions of the SUMVAR= and TYPE= options for the PLOT statement.

About 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 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:

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 graph a percentage or cumulative percentage based on a sum, then 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 TYPE=, SUMVAR=, and FREQ= options.

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, and, as a result, 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 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 252 for details.

Controlling 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 you specify with the ODS statement. You can turn off styles by specifying the NOGSTYLE system option, or 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 240 and the "SYMBOL Statement" on page 252.

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.

Because the system option—GSTYLE—is in effect by default, the procedure uses the style's default bar fill colors, plot line colors, widths, symbols, patterns, and outline colors when producing output. 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 second default symbol for the next PLOT statement, the third default symbol for the next PLOT statement, and so on, continuing 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.) If no SYMBOL or PATTERN statements are in effect and the COLORS= option is not used in the GOPTIONS statement, then 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.

Changing any of the following conditions might 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, then the procedure rotates the default solid pattern through that list, using 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 these graphics options, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

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 the VALUE= option in the PATTERN statement on page 242.

When you use PATTERN statements, the procedure uses the specified patterns until all of the PATTERN definitions 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= on page 963.) 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 252. 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 185.

You can use the IBACK= graphics option to specify image files that fill the background area. For additional information, including a listing of recognized image file types, see "Image File Types Supported by SAS/GRAPH" on page 181 and "Displaying an Image in a Graph Background" on page 182.

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 the PATTERNID= option on page 970 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.

To change the color of...	Use this option...
the axis text	CTEXT=
the axis lines	CAXIS=
the area within the frame	CFRAME=

Procedure Syntax

Requirements: One BAR statement

Global statements: AXIS, FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

Reminder: The procedure can also include the BY, FORMAT, LABEL, and WHERE statements.

Restriction: Not supported by Java and Javaimg

PROC GBARLINE <DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<IMAGEMAP=*output-data-set*>;

BAR *bar-variable* <*/option(s)*>;

<**PLOT** <*/option(s)*>;>...

<**PLOT** <*/option(s)*>;>

PROC GBARLINE Statement

Identifies the data set containing the chart and response variables. Can specify an annotate data set.

Requirements: An input data set is required.

Restriction: Not supported by Java and Javaimg

Syntax

PROC GBARLINE <DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<IMAGEMAP=*output-data-set*>;

Options

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 also: Chapter 29, "Using Annotate Data Sets," on page 641

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.

See also: "SAS Data Sets" on page 54 and "About the Chart Variable" on page 950

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map in an HTML output file. The information in the image map data set includes the shape and coordinates of the elements in the graph and drill-down URLs that have been associated with those elements. The drill-down URLs are provided by one or two variables in the input data set. These variables are identified to the GBARLINE procedure with the HTML= option.

The %IMAGEMAP macro generates the image map in the HTML output file. The macro takes two arguments, the name of the image map data set and the name or fileref of the HTML output file, as shown in the following example:

```
%imagemap(imgmapds, myimgmap.html);
```

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.

Requirements: One bar variable is required.

Global statements: AXIS, FOOTNOTE, LEGEND, PATTERN, TITLE

Supports: Drill-down functionality

Restriction: Not supported by Java and Javaimg

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 select or order the midpoints (bars), to control the tick marks on the response axis, to change the type of chart statistic, to display specific statistics, and to 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 Chapter 14, “SAS/GRAPH Statements,” on page 197 for more information.
- add titles and footnotes to the chart. See “TITLE, FOOTNOTE, and NOTE Statements” on page 279 for more information.
- use an Annotate data set to enhance the chart. See Chapter 29, “Using Annotate Data Sets,” on page 641 for more information.

- display an image in the background of the chart. See “IBACK” on page 386 for more information.
- display images in the bars of the chart. See the IMAGE= option on page 241 for the PATTERN statement.

Syntax

BAR*chart-variable* </*option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options
  - ANNOTATE=*Annotate-data-set*
  - CAUTOREF=*reference-line-color*
  - CAXIS=*axis-color*
  - CERROR=*error-bar-color*
  - CFRAME=*background-color*
  - COUTLINE=*bar-outline-color* | SAME
  - CREF=*reference-line-color*|(*reference-line-color*)|*reference-line-color-list*
  - CTEXT=*text-color*
  - FRAME | NOFRAME
  - LAUTOREF=*reference-line-type*
  - LEGEND=LEGEND<1...99>
  - LREF=*reference-line-type*|(*reference-line-type*)|*reference-line-type-list*
  - NOLEGEND
  - PATTERNID=BY|MIDPOINT
  - SPACE=*bar-spacing*
  - WAUTOREF=*reference-line-width*
  - WIDTH=*bar-width*
  - WOUTLINE=*bar-outline-width*
  - WREF=*reference-line-width*|(*reference-line-width*)|*reference-line-width-list*
- statistic options
  - CFREQ
  - CLM=*confidence-level*
  - CPERCENT
  - ERRORBAR=BARS | BOTH | TOP
  - FREQ
  - FREQ=*numeric-variable*
  - INSIDE=*statistic*
  - MEAN
  - OUTSIDE=*statistic*
  - PERCENT
  - SUM
  - SUMVAR=*summary-variable*
  - TYPE=*statistic*
- midpoint options
  - DISCRETE
  - LEVELS=*number-of-midpoints*

MIDPOINTS=*value-list*
MIDPOINTS=OLD
MISSING
SUBGROUP=*subgroup-variable*

- axes options
  ASCENDING
  AUTOREF
  AXIS=AXIS<1...99>
  CLIPREF
  DESCENDING
  MAXIS=AXIS<1...99>
  MINOR=*number-of-minor-ticks*
  NOAXIS
  NOBASEREF
  NOZERO
  RANGE
  RAXIS=*value-list* | AXIS<1...99>
  REF=*value-list*
- catalog entry description options
  DESCRIPTION='*entry-description*'
  NAME='*entry-name*'
- ODS options
  HTML=*variable*
  HTML_LEGEND=*variable*

Required Arguments

chart-variable
specifies the variable that defines the categories of data to chart. The variable must be in the input data set.

See also: "About the Chart Variable" on page 950

Options

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 on specifying colors, see Chapter 12, "SAS/GRAPH Colors and Images," on page 167. For details on specifying images, see "Specifying Images in SAS/GRAPH Programs" on page 181. For a complete description of the graphics options, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate charts produced by the BAR statement.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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>
See RAXIS= on page 971.

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.

Note: 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. △

Style reference: Color attribute of the GraphWalls element.

CFREQ
displays the cumulative frequency statistic above the bars. A maximum of two statistics can be printed if the INSIDE= option is also used. This option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ option is specified.

See also: "About Chart Statistics" on page 953 and "Displaying Statistics In Bar-Line Charts" on page 973

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= for details on 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.

Style reference: Color attribute of the GraphOutlines element.

See also: "Controlling Patterns, Outlines, Colors, and Images" on page 955

CPERCENT
CPCT
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.

See also: "About Chart Statistics" on page 953 and "Displaying Statistics In Bar-Line Charts" on page 973

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 is of the form (*color1* color2 ...*colorN*) or (*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.

Style reference: LineStyle attribute of the GraphGridLines element.

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 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 or, if the NOGSTYLE system option is specified, 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, and 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.

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. △

Style reference: GraphLabelText, GraphValueText

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.

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=*'entry-description'*

specifies the description of the catalog entry for the chart. The maximum length for the *entry-description* is 256 characters. The description does not appear on the chart. By default, the GBARLINE procedure assigns a description of the form BAR CHART OF *variable*, where *variable* is the name of the chart variable.

The *entry-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 294. The 256-character limit rule is applied 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 first, and then the substitution is performed.

The descriptive text is shown in each of the following:

- the description portion of the Results window
- the catalog entry properties that you can view from the Explorer window
- the Description field of the PROC GREPLAY window
- the data tip text for web output (depending on the device driver you are using).

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.

The LEVELS= option is ignored when you use DISCRETE. The MIDPOINTS= option overrides DISCRETE. 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:

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 percent. 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 non-missing 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 follows:

```
LCLM = MEAN - STDERR * TINV( 1-(1-LEVEL/100)/2, N-1);
```

If you want the error bars to represent a given number, C, of standard errors instead of a confidence interval, and 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, if you want error bars that represent one standard error (C=1) with a sample size of N, then 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 NOFRAME option overrides the CFRAME= option and the IBACK= graphics option.

The color of the frame or backplane outline is the color of the midpoint axis, which is determined by the CAXIS= 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. This option overrides the CFREQ, PERCENT, CPERCENT, SUM, and MEAN options.

See also: "About Chart Statistics" on page 953 and "Displaying Statistics In Bar-Line Charts" on page 973

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.

Because 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 also: "Calculating Weighted Statistics" on page 954

HTML=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS statement. These links are associated with the bars and point to the data or graph you want to display when the user drills down on the bar.

Featured in: Example 3 on page 985

See also: "Data Tips for Web Presentations" on page 598 and "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file that is created by the ODS statement. These links are associated with a legend value and point to the data or graph that you want to display when the user drills down on the value. The values of *variable* can be up to 1024 characters long. Characters after the 1024-character limit (including any closing quotes) are truncated.

Restriction: Not supported by ActiveX

See also: "Data Tips for Web Presentations" on page 598 and "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

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

To display statistics with INSIDE=SUM or INSIDE=MEAN, you must also specify the SUMVAR= option.

See also: "About Chart Statistics" on page 953 and "Displaying Statistics In Bar-Line Charts" on page 973

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 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. LEGEND= 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 LEGEND= is specified for both the BAR and the POSITION= options on the LEGEND statements are the same location, a single combined legend will be drawn. However, ActiveX output will display separate but adjacent legends.

The ActiveX device does not support all LEGEND statement options. See "LEGEND Statement" on page 225 for more information.

Featured in: Example 2 on page 1067

Restriction: Partially supported by ActiveX

See also: SUBGROUP= on page 971 and "LEGEND Statement" on page 225

LEVELS=*number-of-midpoints* | ALL
specifies the number of midpoints for the numeric chart variable. The range for each midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985). If your data contains a large number of unique midpoint values (over 200), then you can use the XPIXELS and YPIXELS GOPTIONS to allow the device driver to render a larger (and more readable) graph. The LEVELS= option is ignored if any of these statements are 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 is of the form (*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 also: "AXIS Statement" on page 198 and "About Midpoints" on page 950

MEAN
displays the mean statistic above the bars. 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 or if the FREQ, CFREQ, PERCENT, CPERCENT, or SUM option is specified. MEAN is ignored unless you also use the SUMVAR= option.

See also: "About Chart Statistics" on page 953 and "Displaying Statistics In Bar-Line Charts" on page 973

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 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.

For a complete description of *value-list*, see ORDER= on page 205.

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 either adjusting the size of the text with the HTEXT= graphics option or by increasing the number of cells in your graphics display 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.

See also: "About Midpoints" on page 950

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 bar response axis.

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='*entry-name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default name is gbarline. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, gbarline1.

See also: "About Filename Indexing" on page 99

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, but

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 option. 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: If you assign bar label names to each bar with the VALUE= option in an AXIS statement, and a bar is omitted from your graph, then the label names might be inadvertantly 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.

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 also: "About Chart Statistics" on page 953 and "Displaying Statistics In Bar-Line Charts" on page 973

PATTERNID=BY | MIDPOINT | SUBGROUP
specifies the way 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. This option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ or CFREQ option is specified.

See also: "About Chart Statistics" on page 953 and "Displaying Statistics In Bar-Line Charts" on page 973

RANGE
displays on the axis of the chart the range of numeric values represented by each bar. In the graphics output, the less-than symbol (<) and the less-than-or-equal-to symbol (<=) are used to accurately specify the starting and ending values of each 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= option on page 968. If specified, the DISCRETE option overrides the RANGE option.

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= option on page 968 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 also: "AXIS Statement" on page 198

REF=*value-list*
draws reference lines at the specified points on the chart response axis. See the MIDPOINTS= option on page 968 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 non-negative 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 number of bars on the chart. Use SPACE=0 to leave no space between adjacent bars.

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.

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.

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.

Featured in: Example 3 on page 985

See also: "LEGEND Statement" on page 225

SUM
displays the sum statistic above the bars. 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 or if the FREQ, CFREQ, PERCENT, or CPERCENT option is specified. SUM is ignored unless you also use the SUMVAR= option.

See also: "About Chart Statistics" on page 953 and "Displaying Statistics In Bar-Line Charts" on page 973

SUMVAR=*summary-variable*
specifies a numeric variable for sum or mean calculations. The GBARLINE procedure calculates the sum or, if requested, the mean of *summary-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.

Featured in: Example 1 on page 981

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:

  SUM
  sum (default)

  MEAN
  mean

Because you cannot use TYPE=FREQ, TYPE=CFREQ, TYPE=PERCENT, or TYPE=CPERCENT 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 954.

See also: "About Chart Statistics" on page 953 for a complete description of statistic types

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;
```

If the requested bar width results in a chart that is too large to fit in the space available for the midpoint axis, then 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 the 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 is of the form (*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.

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 Chapter 14, "SAS/GRAPH Statements," on page 197 for a complete discussion.

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, but it does not change the range of values for the chart as a whole. For details, see “About Midpoints” on page 950.

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 description of the ORDER= option on page 205.

PLOT Statement

Creates one or more plot overlays on top of the bar-line chart.

Requirements: If specified, PLOT statements must be specified after the BAR statement.

Global statements: AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE

Supports: Data tips and drill-down functionality

Restriction: Not supported by Java

Description

The PLOT statement specifies one plot request. You can use multiple PLOT statements to generate multiple plots. The PLOT statement automatically

- 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 interpolations. For more information on the SYMBOL statement, see "SYMBOL Statement" on page 252.

In addition, you can use global statements to add a legend, modify the axis, or add titles, footnotes, and notes to the plot.

Syntax

PLOT </*options(s)*>;

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 952 and the description of the SUMVAR= option.

Option(s) can be one or more options from any or all of the following categories:

- appearance options:
  - ANNOTATE=*Annotate-data-set*
  - ASCENDING
  - CAUTOREF=*reference-line-color*
  - CAXIS=*axis-color*
  - CREF=*reference-line-color*|(*reference-line-color*)|*reference-line-color-list*
  - CTEXT=*text-color*
  - DESCENDING
  - LAUTOREF=*reference-line-type*
  - LEGEND=LEGEND<1...99>
  - LREF=*reference-line-type*|(*reference-line-type*)|*reference-line-type-list*
  - NOLINE
  - NOMARKER
  - WAUTOREF=*reference-line-width*
  - WREF=*reference-line-width*|(*reference-line-width*)|*reference-line-width-list*
- statistic options:
  - CFREQ
  - CPERCENT
  - FREQ
  - FREQ=*numeric-variable*
  - MEAN
  - PERCENT
  - SUM
  - SUMVAR=*plot-variable*
  - TYPE=*statistic*
- axes options:
  - AUTOREF

AXIS=AXIS<1...99>
CLIPREF
MINOR=*number-of-minor-ticks*
NOAXIS
RAXIS=*value-list* | AXIS<1...99>
REF=*value-list*

□ ODS options:

HTML=*variable*
HTML_LEGEND=*variable*

Options

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>
See RAXIS= on page 979.

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.

CLIPREF
clips the reference lines at the bars. Using this option makes the reference lines appear to be behind the bars.

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 is of the form (*color1* color2 ...*colorN*) or (*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, including 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 or, if the NOGSTYLE system option is specified, then the default color is 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, and 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.

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. △

Style reference: GraphLabelText, GraphValueText

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=*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.

Because 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 in the HTML file created by the ODS statement. These links are associated with the plot points and bars. The links point to the data or graph that you want to display when the user drills down on the plot point or bar area. This option is featured in Example 3 on page 985.

See also: "Data Tips for Web Presentations" on page 598 and "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file that is created by the ODS statement. These links are associated with a legend value and point to the data or graph that you want to display when the user drills down on the value. The values of *variable* can be up to 1024 characters long. Characters after the 1024-character limit (including any closing quotes) are truncated.

Restriction: Not supported by Java and ActiveX

See also: "Data Tips for Web Presentations" on page 598 and "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601.

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. LEGEND= 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.

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.

You can specify both a BAR and PLOT legend on the same graph. If LEGEND= is specified for both the BAR and the POSITION= options on the LEGEND statements are the same location, a single combined legend will be drawn. However, ActiveX output will display separate but adjacent legends.

The ActiveX device does not support all LEGEND statement options. See "LEGEND Statement" on page 225 for more information.

Featured in: Example 3 on page 985

Restriction: Not supported by Java. Partially supported by ActiveX.

See also: "LEGEND Statement" on page 225

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 is of the form (*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

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, but 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.

NOLINE
suppresses the line connecting the PLOT symbols, regardless of what is specified in the SYMBOL statement.

NOMARKER
suppresses drawing the marker symbol, regardless of what is specified in the SYMBOL statement.

RAXIS=*value-list* | AXIS<1...99>
AXIS=*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 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*>

Any response values that exceed the highest tick mark value are not plotted. The overlay plot line connects only the visible plot response values.

REF=*value-list*
draws reference lines at the specified points using the chart response axis. See the MIDPOINTS= option on page 968 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.

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.

Featured in: Example 1 on page 981

See also: "About Response Variables" on page 952

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 FREQ= to calculate percentages or frequencies based on a sum.

See also: "About Chart Statistics" on page 953 and "Calculating Weighted Statistics" on page 954

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 of the form (*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.

About 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 "SYMBOL Statement" on page 252 for a complete discussion of the features of the SYMBOL statement.

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.

"SYMBOL Statement" on page 252 describes the JOIN, STEP, and NONE interpolation methods.

Examples

Example 1: Producing a Basic Bar-Line Chart

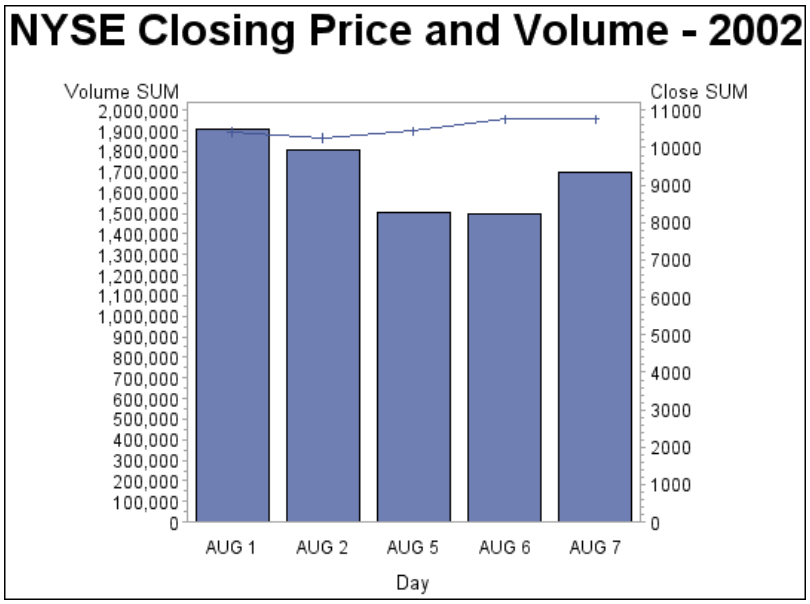
Procedure Features:

BAR statement options:

SUMVAR=

PLOT statement options:

SUMVAR=



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.

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 and footnote.

```
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

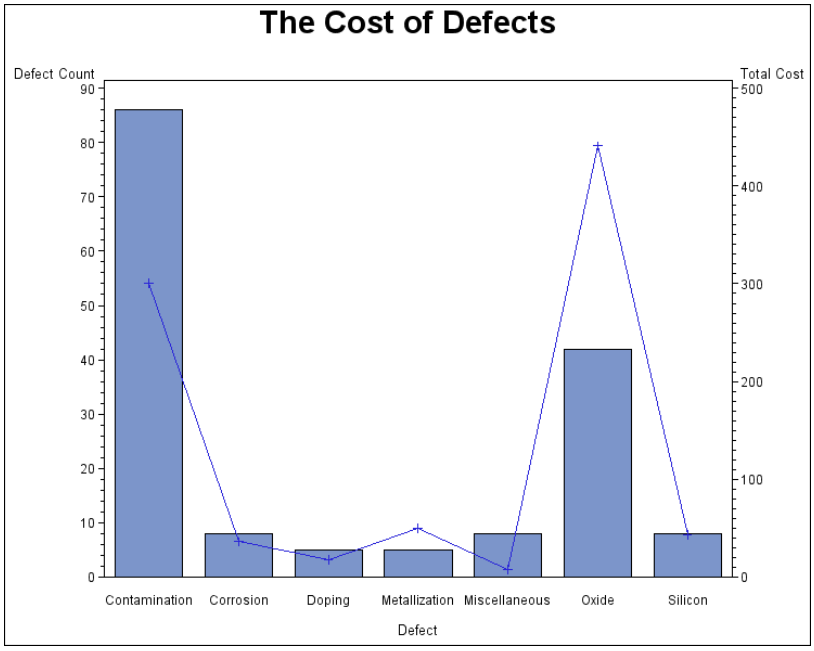
Procedure Features:

BAR statement options:
- AXIS=
- SUMVAR=

PLOT statement options:
- AXIS=
- FREQ=
- SUMVAR=

Other Features:

AXIS statement



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.

Set the graphics environment.

```
goptions reset=all border;
```

Create the data set FAILURE. Each observation of the FAILURE data set contains the type of manufacturing defect, a count of how many times it occurred, and a cost associated with the defect.

```
data failure;
   length Defect $15;
   input  Defect Count @@;
   select (Defect) ;
      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;

datalines;
Contamination 15   Corrosion 2      Doping 1         Metallization 2
Miscellaneous 3    Oxide 8          Silicon 1        Contamination 16
Corrosion 3        Doping 1         Metallization 3  Miscellaneous 1
Oxide 9            Silicon 2        Contamination 20 Corrosion 1
Doping 1           Metallization 0  Miscellaneous 3  Oxide 7
Silicon 2          Contamination 12 Corrosion 1      Doping 1
Metallization 0    Miscellaneous 0  Oxide 10         Silicon 1
Contamination 23   Corrosion 1      Doping 1         Metallization 0
Miscellaneous 1    Oxide 8          Silicon 2
;
run;
```

Define the title and footnote.

```
title1 "The Cost of Defects";
footnote1 j=r "GBLWTSTA";
```

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. GBARLINE multiplies the value of the FREQ= variable by the value of the COUNT variable, and uses the result to determine the plot points.

```
proc gbarline data=failure;
   bar Defect/ 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

Procedure Features:

BAR statement options:

- DISCRETE
- HTML=
- LEGEND=
- MAXIS=
- RAXIS=
- SUBGROUP=
- SUMVAR=

PLOT statement options:

- AXIS=
- HTML=
- LEGEND=

Multiple PLOT statements:

- SUMVAR=

Other Features:

STYLE= option in the ODS statement

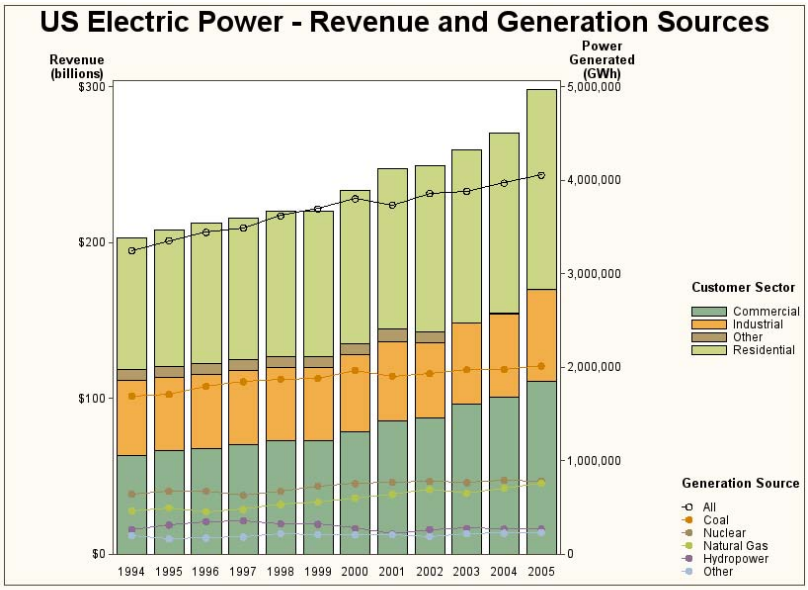
AXIS statements

LEGEND statements

ODS HTML statement

SYMBOL statement

Data set: SASHELP.ELECTRIC



Link to Bar Data: USEIA Energy Customer Sectors
Link to Line Data: USEIA Energy Generation Sources

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.

Set the graphics environment.

```
goptions reset=all border;
```

Open the HTML destination. 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. You can also use the ODS PATH= and FILE= options to specify a location for the output files.

```
ods listing close;
ods 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=r "GBLPOWER";

footnote2  j=l italic
           link="http://www.eia.doe.gov/cneaf/electricity/epa/epat7p3.html"
           "Link to Bar Data: USEIA Energy Customer Sectors";

footnote3 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. Using different LEGEND statements and positioning the legends in different places for the bar chart and the overlay plots 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 the ODS HTML destination. You must close the HTML destination before you can view the output with a browser.

```
ods HTML close;
ods listing;
```

CHAPTER

36

The GCHART Procedure

Overview

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

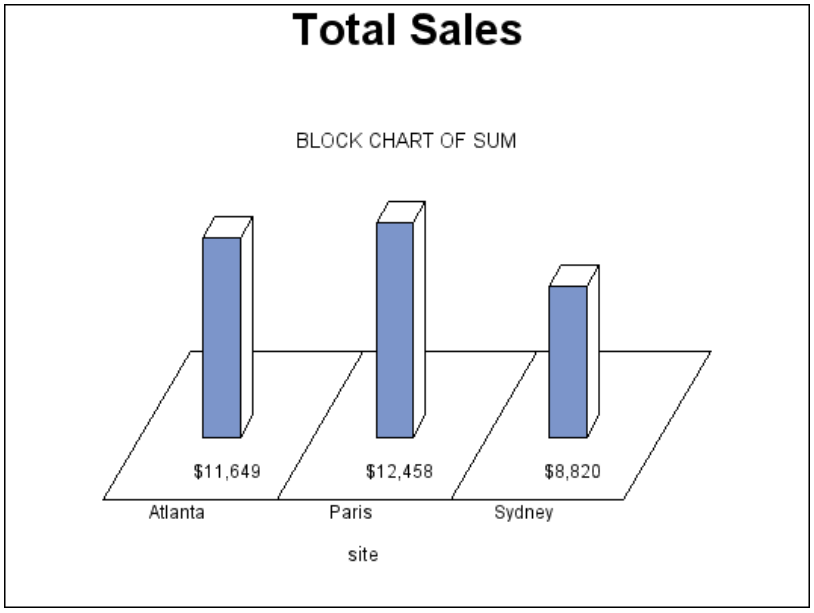
About Block Charts

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 36.1 on page 991 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. Midpoint values are, by default, 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 on page 1066. For more information on producing block charts, see "BLOCK Statement" on page 1005.

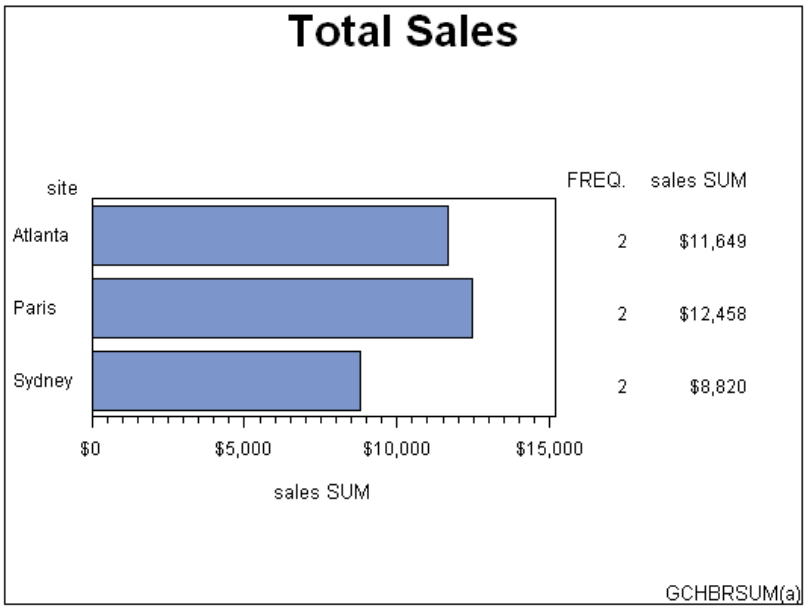
About Bar Charts

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 the procedure you choose.

Figure 36.2 on page 992 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. Midpoint values are, by default, 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.

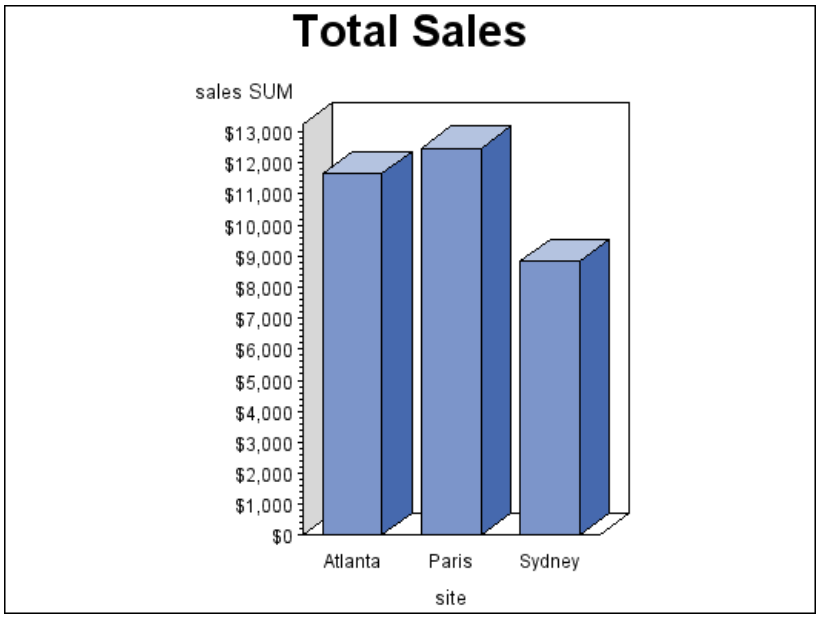
Figure 36.2 Horizontal Bar Chart (GCHBRSUM (a))



The program for this chart is Example 3 on page 1070.

Figure 36.3 on page 992 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. Horizontal bar charts by default 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 of each bar.

Figure 36.3 Vertical (Three-Dimensional) Bar Chart (GCHBRSUM(b))



The program for this chart is Example 3 on page 1070. For more information on producing horizontal and vertical bar charts, see "HBAR, HBAR3D, VBAR, and VBAR3D Statements" on page 1015.

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 36.4 on page 993 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))

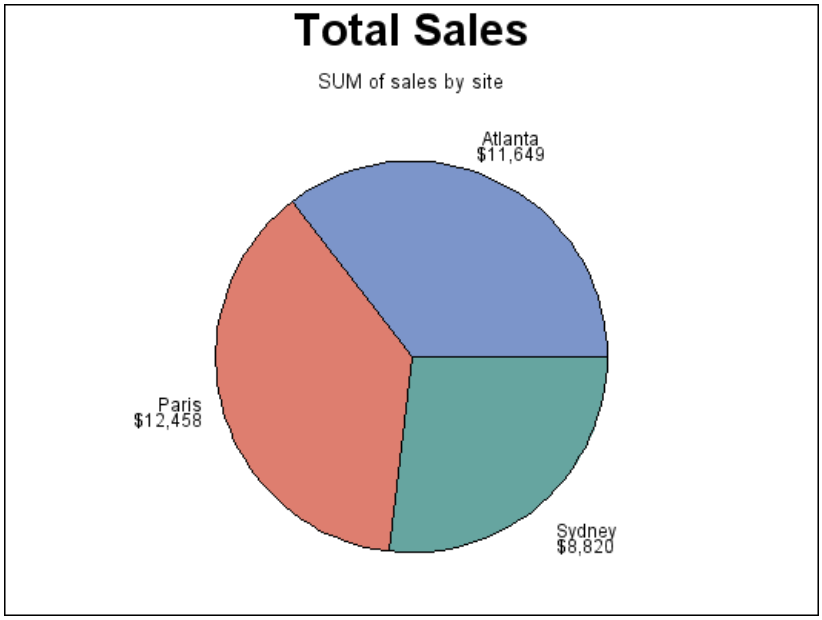


Figure 36.5 on page 994 shows the three-dimensional version of the same pie chart. This version features the exploded slice.

Figure 36.5 Three-Dimensional Pie Chart (GCHPISUM(b))

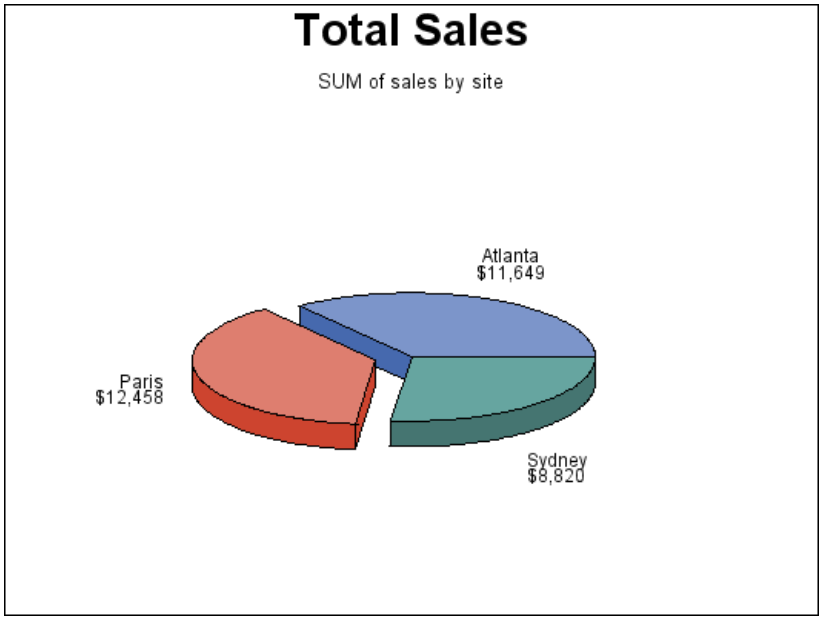
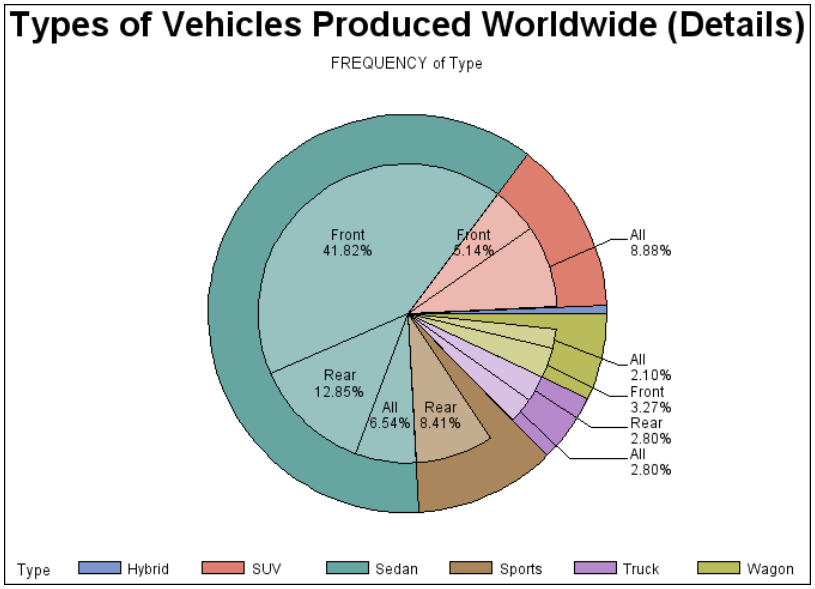


Figure 36.6 on page 994 shows a detail pie chart generated from the same data.

Figure 36.6 Detail Pie Chart (GCHDTPIE)



The programs for these charts are in Example 7 on page 1080 and Example 13 on page 1092. For more information on producing pie or donut charts, see "PIE, PIE3D, and DONUT Statements" on page 1038.

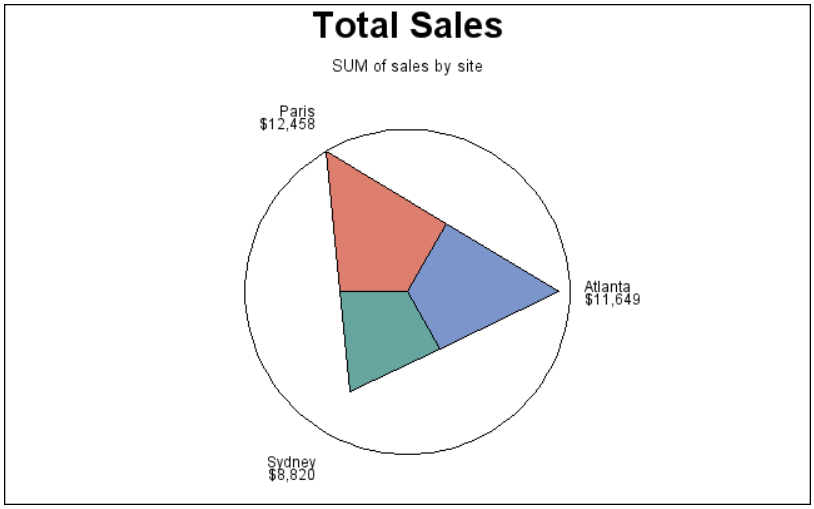
About Star Charts

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 36.7 on page 995 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)



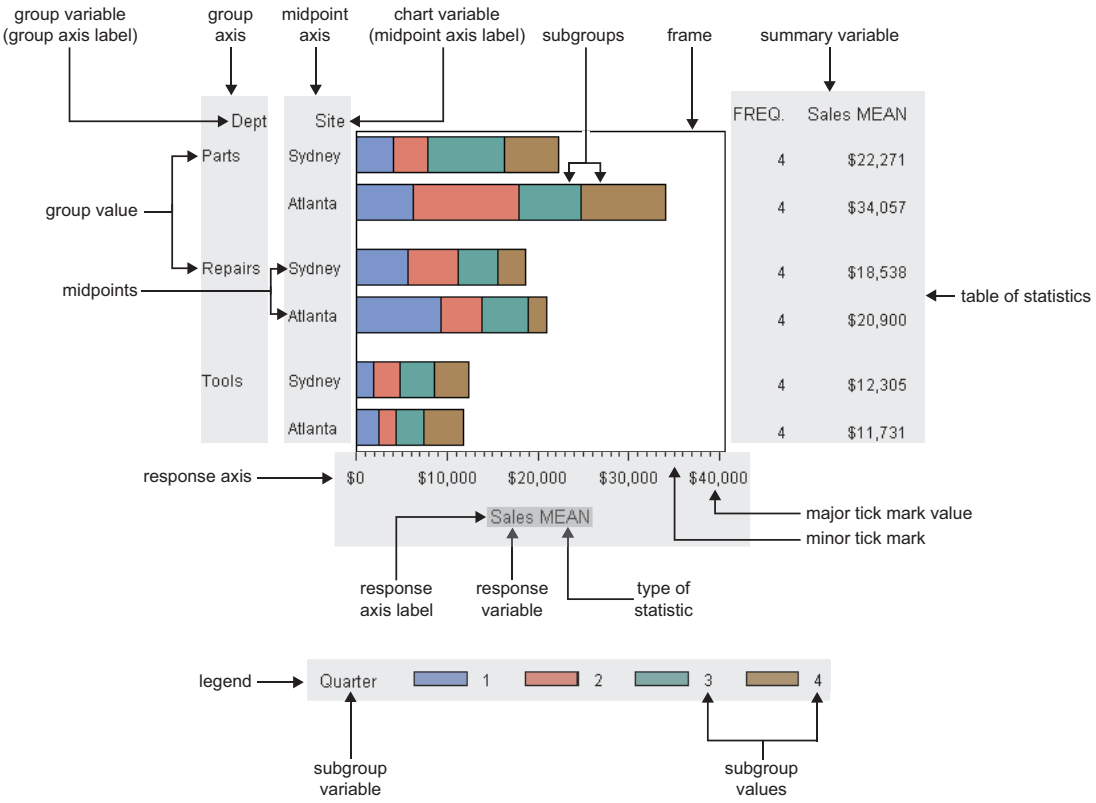
The program for this chart is Example 11 on page 1088. For more information on producing star charts, see "STAR Statement" on page 1055. For an alternative way of producing similar types of charts, see Chapter 47, "The GRADAR Procedure," on page 1419.

Concepts

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*.

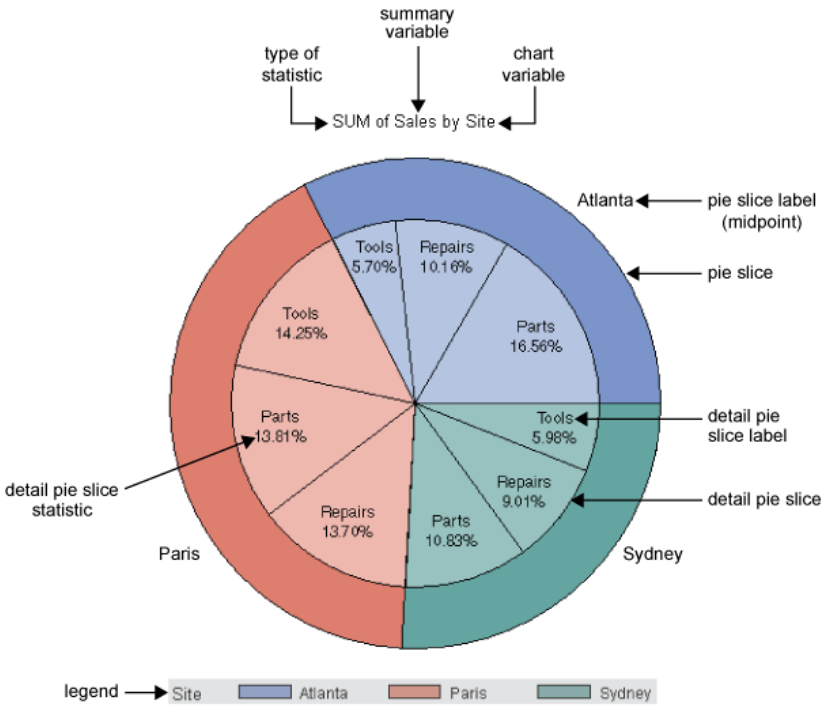
Figure 36.8 on page 996 and Figure 36.9 on page 997 illustrate these terms as well as other terms used with the GCHART procedure.

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.

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 contribute.

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: If you apply a format that converts multiple values or a range of values to a single formatted value, then 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.

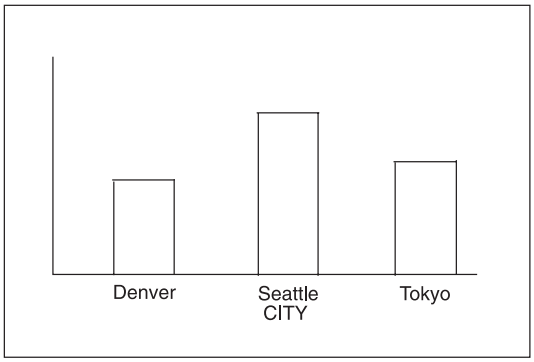
About Midpoints

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



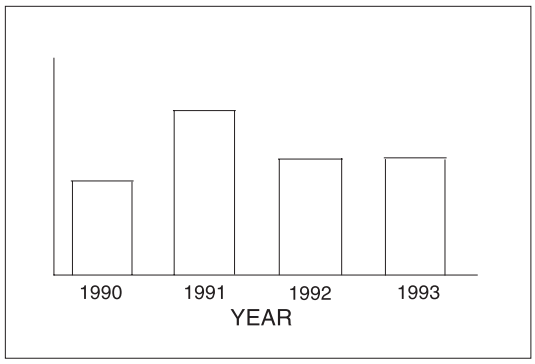
(In pie charts, midpoint values that compose a small percentage of the total for the chart might be placed in the OTHER slice and will not produce a separate midpoint.)

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



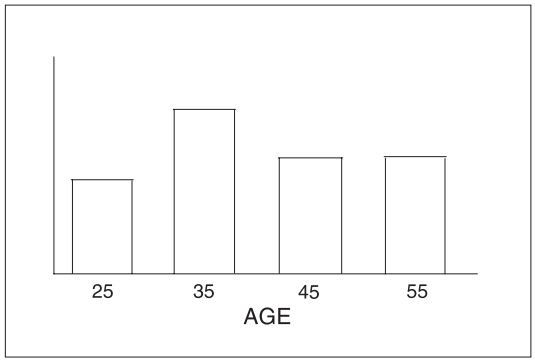
By default, numeric midpoints are arranged in ascending order. The DISCRETE option is very useful for working with dates and numeric values with text user-defined formats. If the numeric variable has an associated format, each formatted value generates a separate midpoint. Formatted numeric variables 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 36.10 on page 998, 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, to control the range of values each midpoint represents, or to 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.

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

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

Value of COUNT	Value of HEIGHT	Number of times the observation is used	Value used for HEIGHT
.	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 and 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

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 "PATTERN Statement" on page 240.

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 and rotates it 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 and 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 Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

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 on all patterns, see "PATTERN Statement" on page 240. 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 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's a default or user-defined pattern, use the COUTLINE= option in the action statement that generates the 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 IMAGE= option. 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 listing of recognized image file types, see "Image File Types Supported by SAS/GRAPH" on page 181.

Procedure Syntax

Requirements: At least one BLOCK, HBAR, HBAR3D, VBAR, VBAR3D, PIE, PIE3D, DONUT, or STAR statement is required.

Global statements: AXIS, FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

Reminder: The procedure can include the BY, FORMAT, LABEL, and WHERE statements as well as the SAS/GRAPH NOTEstatement.

Supports:

RUN-group processing

PROC GCHART<DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>;

BLOCK *chart-variable(s)* </ *option(s)*>;

HBAR | HBAR3D | VBAR | VBAR3D*chart-variable(s)* </ *option(s)*>;

PIE | PIE3D | DONUT *chart-variable(s)* </ *option(s)*>;

STAR *chart-variable(s)* </ *option(s)*>;

PROC GCHART Statement

Identifies the data set containing the chart variables. Can specify annotation and an output catalog.

Requirements: An input data set is required.

Syntax

PROC GCHART<DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>;

Options

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 ANNOTATE= in the action statement.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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.

See also: "SAS Data Sets" on page 54 and "About Chart Variables" on page 997

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.

See also: "Specifying the Catalog Name and Entry Name for Your GRSEGs" on page 100

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map in an HTML output file. The information in the image map data set includes the shape and coordinates of the elements in the graph and drill-down URLs that have been

associated with those elements. The drill-down URLs are provided by one or two variables in the input data set. These variables are identified to the GCHART procedure with the HTML= or HTML_LEGEND= or both options.

The %IMAGEMAP macro generates the image map in the HTML output file. The macro takes two arguments, the name of the image map data set and the name or fileref of the HTML file, as shown in the following example:

```
%imagemap(imgmapds, myimgmap.html);
```

Restriction: Not supported by Java or ActiveX

BLOCK Statement

Creates block charts in which the height of the blocks represents the value of the chart statistic for each category of data.

Requirements: At least one chart variable is required.

Global statements: LEGEND, PATTERN, TITLE, FOOTNOTE

Supports: Drill-down functionality

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, and 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 59 for details.

Syntax

BLOCK *chart-variable(s)* </ *option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options
  - ANNOTATE=*Annotate-data-set*
  - BLOCKMAX=*max-value*
  - CAXIS=*grid-color*
  - COUTLINE=*block-outline-color* | SAME
  - CTEXT=*text-color*
  - LEGEND=LEGEND<1...99>
  - NOHEADING
  - NOLEGEND
  - PATTERNID=BY |GROUP |MIDPOINT |SUBGROUP
  - WOUTLINE=*block-outline-width*
- midpoint options
  - DISCRETE
  - GROUP=*group-variable*
  - LEVELS=*number-of-midpoints*
  - MIDPOINTS=*value-list*
  - MIDPOINTS=OLD
  - MISSING
  - SUBGROUP=*subgroup-variable*
- statistic options
  - FREQ=*numeric-variable*
  - G100
  - SUMVAR=*summary-variable*
  - TYPE=*statistic*
- catalog entry description options
  - DESCRIPTION='*entry-description*'
  - NAME='*entry-name*'
- ODS options
  - HTML=*variable*
  - HTML_LEGEND=*variable*
- axes options
  - GAXIS=AXIS<1...99>
  - MAXIS=AXIS<1...99>
  - RAXIS=*value-list* | AXIS<1...99>

Required Arguments

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 also: "About Chart Variables" on page 997

Options

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 on specifying colors, see Chapter 12, "SAS/GRAPH Colors and Images," on page 167. For a complete description of the graphics options, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate charts produced by the BLOCK statement.

Note: Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block charts. △

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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 as if *max-value* 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 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 other devices.

Style reference: Color attribute of the GraphAxisLines element.

Featured in: Example 2 on page 1067

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 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 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.

Style reference: Color attribute of the GraphOutlines element.

Featured in: Example 2 on page 1067

Restriction: Partially supported by Java and ActiveX

See also: "Controlling Block Chart Patterns and Colors" on page 1014 and "About Patterns" on page 1002

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 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 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 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 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.

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. △

Style reference: Color attributes of the GraphValueText and the GraphLabelText elements.

DESCRIPTION='description'
specifies the description of the catalog entry for the chart. The maximum length for *entry-description* is 256 characters. The description does not appear on the chart. By default, the GCHART procedure assigns a description of the form BLOCK CHART OF *variable*, where *variable* is the name of the chart variable.

The *entry-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 294. 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.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window.
- the catalog-entry properties that you can view from the Explorer window.
- the Description field of the PROC GREPLAY window.
- the data tip text for Web output (depending on the device driver you are using). See "Data Tips for Web Presentations" on page 598 for details.

Alias: DES=

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 associated with it, then each formatted value is treated as a midpoint.

The LEVELS= option is ignored when you use DISCRETE. The MIDPOINTS= option overrides DISCRETE.

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 the PERCENT, CPERCENT, FREQ, or CFREQ statistics 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*.

See also: "Calculating Weighted Statistics" on page 1001

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. GAXIS= 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 "AXIS Statement" on page 198 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.

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

Restriction: Supported by Java and ActiveX only.

See also: "AXIS Statement" on page 198

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 percent 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 percent 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.

Featured in: Example 2 on page 1067

HTML=*variable*

identifies the variable in the input data set whose values create links in the HTML file that is created by the ODS statement. These links are associated with an area of the chart and point to the data or graph you want to display when the user drills down on the area. There is no limit on the length of the variable.

See also: "Overview of Enhancing Web Presentations" on page 596

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file that is created by the ODS statement. These links are associated with a legend value and point to the data or graph that you want to display when the user drills down on the value. The values of *variable* can be up to 1024 characters long. Characters after the 1024-character limit (including any closing quotes) are truncated.

Restriction: Not supported by Java orActiveX

See also: "Overview of Enhancing Web Presentations" on page 596

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 "LEGEND Statement" on page 225 for more information.

Featured in: Example 2 on page 1067

Restriction: Partially supported by Java and ActiveX

See also: SUBGROUP= on page 1013 and "LEGEND Statement" on page 225

LEVELS=*number-of-midpoints*
specifies the number of midpoints for the numeric chart variable. The range for each midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985). The LEVELS= option is ignored if the following is true:

- The chart variable is character type.
- The DISCRETE option is used.
- The MIDPOINTS= option is used.

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.

The Java and ActiveX devices do not support all AXIS statement options. See "AXIS Statement" on page 198 for more information.

Featured in: Example 4 on page 1072

Restriction: Supported by Java and ActiveX only

See also: "AXIS Statement" on page 198 and "About Midpoints" on page 998

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 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 ORDER= option on page 205 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 display using the HPOS= and VPOS= graphics options.

Featured in: Example 2 on page 1067

See also: "About Midpoints" on page 998

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='*entry-name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default name is GCHART. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, GCHART1.

See also: "About Filename Indexing" on page 99

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.

Featured in: Example 2 on page 1067

Restriction: Not supported by Java or ActiveX

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 the way 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 SUBGROUP=, 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. △

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

See also: "Controlling Block Chart Patterns and Colors" on page 1014

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= option on page 1029 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 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 or if there is an ORDER= list in the AXIS statement assigned to the response axis.

The Java and ActiveX devices do not support all AXIS statement options. See "AXIS Statement" on page 198 for more information.

Featured in: Example 4 on page 1072 and "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

Restriction: Supported by Java and ActiveX only

See also: "AXIS Statement" on page 198

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.

Featured in: Example 2 on page 1067

See also: "LEGEND Statement" on page 225

SUMVAR=*summary-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 summary variable used with the BLOCK statement have a maximum length of 8.

When you use SUMVAR=, the TYPE= option value must be either SUM or MEAN. With SUMVAR=, the default is TYPE=SUM.

Featured in: Example 1 on page 1066

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

Because you cannot specify the statistics PERCENT, CPERCENT, FREQ, or CFREQ in conjunction with the SUMVAR= option, you must use FREQ= to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum. See also "Calculating Weighted Statistics" on page 1001.

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.

Featured in: Example 2 on page 1067

See also: "About Chart Statistics" on page 1000

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

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 1002 for more information on 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= on page 242.

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= option on page 1007.

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, and 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 PATTERNID= option on page 1012 for details.

Axis color

By default, axis elements use the color specified in the current style. To change the grid color, use the CAXIS= option. To change the axis text color, use the CTEXT= option.

Controlling Block Chart Text

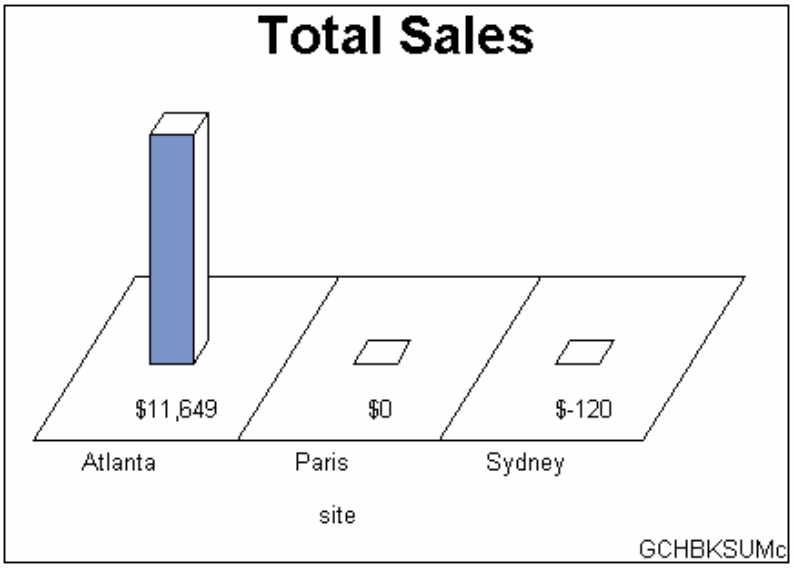
To control the font and size of text on the chart, use the FTEXT= and HTEXT= graphics options. See Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for a description of these options.

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 on page 1067.

Displaying Negative or Zero Values

The relative block heights in the chart represent the scaled value of the chart statistic value for the midpoint. If the statistic has a value of 0 or, in the case of sum and mean, a negative value, the base of the block is drawn in the square for the corresponding midpoint. Figure 36.13 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 Statements

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.

Requirements: At least one chart variable is required.

Global statements: AXIS, LEGEND, PATTERN, TITLE, FOOTNOTE

Supports: Drill-down functionality

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/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, using the first color in the device's color list

You can use statement options to select or order the midpoints (bars), to control the tick marks on the response axis, to change the type of chart statistic, to display specific statistics, and to 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 Chapter 14, "SAS/GRAPH Statements," on page 197 for more information.
- add titles, footnotes, and notes to the chart. See "TITLE, FOOTNOTE, and NOTE Statements" on page 279 for more information.
- use an Annotate data set to enhance the chart. See Chapter 29, "Using Annotate Data Sets," on page 641 for more information.
- display an image in the axis frame or backplane area. See the IFRAME= option on page 1026.
- display images in the bars of an HBAR or VBAR chart. See the PATTERN statement IMAGE= option on page 241.

Syntax

HBAR | HBAR3D | VBAR | VBAR3D *chart-variable(s)* </ *option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options

  ANNOTATE=*Annotate-data-set*

  CAUTOREF=*reference-line-color*

  CAXIS=*axis-color*

  CERROR=*error-bar-color*

  CFRAME=*background-color*

  CLM=*confidence-level*

  COUTLINE=*bar-outline-color* | SAME

  CREF=*reference-line-color*|(*reference-line-color*)|*reference-line-color-list*

  CTEXT=*text-color*

  FRAME | NOFRAME

  GSPACE= *group-spacing*

  IFRAME= *fileref* | '*external-file*'

IMAGESTYLE = TILE | FIT
LAUTOREF=*reference-line-type*
LEGEND=LEGEND<1...99>
LREF=*reference-line-type*|(*reference-line-type*)|*reference-line-type-list*
NOLEGEND
NOPLANE
PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP
SHAPE=*3D-bar-shape* (HBAR3D and VBAR3D only)
SPACE=*bar-spacing*
WAUTOREF=*reference-line-width*
WIDTH=*bar-width*
WOUTLINE=*bar-outline-width*
WREF=*reference-line-width*

□ statistic options

CFREQ
CFREQLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
CPERCENT
CPERCENTLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
ERRORBAR=BARS | BOTH | TOP
FREQ
FREQLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
FREQ=*numeric-variable*
G100
INSIDE=*statistic*
MEAN
MEANLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
NOSTATS (HBAR and HBAR3D only)
OUTSIDE=*statistic*
PERCENT
PERCENTLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
PERCENTSUM
SUM
SUMLABEL='*column-label*' | NONE (HBAR and HBAR3D only)
SUMVAR=*summary-variable*
TYPE=*statistic*

□ midpoint options

DISCRETE
GROUP=*group-variable*
LEVELS=*number-of-midpoints*|ALL
MIDPOINTS=*value-list*
MIDPOINTS=OLD
MISSING
RANGE
SUBGROUP=*subgroup-variable*

- axes options
  - ASCENDING
  - AUTOREF
  - AXIS=AXIS<1...99>
  - CLIPREF
  - DESCENDING
  - FRONTREF (HBAR3D and VBAR3D only)
  - GAXIS=AXIS<1...99>
  - MAXIS=AXIS<1...99>
  - MINOR=*number-of-minor-ticks*
  - NOAXIS
  - NOBASEREF
  - NOZERO
  - RANGE
  - RAXIS=*value-list* | AXIS<1...99>
  - REF=*value-list*
- catalog entry description options
  - DESCRIPTION='*entry-description*'
  - NAME='*entry-name*'
- ODS options
  - HTML=*variable*
  - HTML_LEGEND=*variable*

Required Arguments

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 also: "About Chart Variables" on page 997

Options

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 on specifying colors, see Chapter 12, "SAS/GRAPH Colors and Images," on page 167. For details on specifying images, see "Specifying Images in SAS/GRAPH Programs" on page 181. For a complete description of the graphics options, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate charts produced by the bar chart statement.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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.

Featured in: Example 5 on page 1075

AXIS=AXIS<1...99>

See RAXIS= on page 1032.

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 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 other devices.

This option also specifies the default color for all reference lines.

Style reference: Color attribute of the GraphAxisLines attribute.

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 in the backplane in 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.

CFRAME= is overridden by the NOFRAME and IFRAME= options.

Note: If the background color, the bar color, and the outline color are the same, you might not be able to distinguish the bars. △

Alias: CFR=

Style reference: Color attribute of the GraphWalls element.

Featured in: Example 4 on page 1072

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.

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

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

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.

Featured in: Example 5 on page 1075

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= for details on 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 on 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 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.

Style reference: Color attribute of the GraphOutlines element.

Featured in: Example 3 on page 1070, Example 5 on page 1075 and Example 6 on page 1078

See also: "Controlling Bar Chart Patterns, Colors, and Images" on page 1037 and "About Patterns" on page 1002

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=

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

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 is of the form (*color1 color2 ...colorN*) or (*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, 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 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.

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. △

Style reference: Color attributes of the GraphValueText and the GraphLabelText elements.

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 or that is specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

DESCRIPTION='description'

specifies the description of the catalog entry for the chart. The maximum length for *entry-description* is 256 characters. The description does not appear on the chart. By default, the GCHART procedure assigns a description of the form BAR CHART OF *variable*, where *variable* is the name of the chart variable.

The *entry-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 294. 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.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window.
- the catalog-entry properties that you can view from the Explorer window.
- the Description field of the PROC GREPLAY window.
- the data tip text for Web output (depending on the device driver you are using). See "Data Tips for Web Presentations" on page 598 for details.

Alias: DES=

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

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 associated with it, each formatted value is treated as a midpoint.

The LEVELS= 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.

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

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 percent. 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 non-missing 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);
```

If you want the error bars to represent a given number C of standard errors instead of a confidence interval, and 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, if 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 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.

FRAME | NOFRAME
specifies whether the two-dimentional axis area frame or the three-dimensional backplane is drawn. The default is FRAME, which draws a frame around the axis area (in two-dimensional bar charts) or generates a colored three-dimensional backplane (in three-dimensional bar charts). 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 backplane 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 NOFRAME option overrides the CFRAME= and IFRAME= options and the IBACK= goption"IBACK" on page 386.

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=

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618 and Example 6 on page 1078

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.

Featured in: Example 5 on page 1075

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

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 will break 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.

Featured in: Example 5 on page 1075 and Example 6 on page 1078

Restriction: Not supported by Java or ActiveX

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*.

See also: "Calculating Weighted Statistics" on page 1001

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 percent 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 percent 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 "AXIS Statement" on page 198 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.

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

Restriction: Partially supported by Java and ActiveX

See also: "AXIS Statement" on page 198

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 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 and 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. If the group label in a vertical bar chart is narrower than all the bars in the group, 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.

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

GSPACE=*group-spacing*
specifies the amount of extra space between groups of bars. *Group-space* can be any non-negative 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.

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

HTML=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS statement. These links are associated with an area of the chart and point to the data or graph you want to display when the user drills down on the area. There is no limit on the length of the variable.

See also: "Overview of Enhancing Web Presentations" on page 596

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS statement. These links are associated with a legend value and point to the data or graph you want to display when the user drills down on the value. The values of *variable* can be up to 1024 characters long. Characters after the 1024-character limit (including any closing quotes) are truncated.

Restriction: Not supported by Java or ActiveX.

See also: "Overview of Enhancing Web Presentations" on page 596

IFRAME=*fileref* | *'external-file'*
specifies an image file to use on a 2–D chart's frame area or a 3–D chart's backplane. 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 file name of the image file you want to use. The format of external-image-file varies across operating environments.

See also the IMAGESTYLE= option and "Displaying an Image in Graph Frame" on page 184.

This option is overridden by the NOIMAGEPRINT goption.

To fill the backplane frame of two-dimensional bar charts, see the IBACK= goption.

Restriction: Not supported by Java

IMAGESTYLE= TILE | FIT
specifies whether to use multiple instances of an image to fill the axis frame or backplane (TILE) or to stretch a single instance of an image to fill the axis frame or backplane frame (FIT). The TILE value is the default.

See also the IFRAME= option.

Restriction: Not supported by Java

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. 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 INSIDE=.

Statistic can be one of the following:

- FREQ
- CFREQ
- CPERCENT | CPCT

- MEAN
- PERCENT | PCT
- SUM

If the bars are subgrouped, only the following statistics are valid:

- FREQ
- PERCENT | PCT
- SUBPCT
- 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 1001.

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.

Featured in: Example 4 on page 1072, and "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

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;
```

The Java and ActiveX devices do not support all LEGEND statement options. See "LEGEND Statement" on page 225 for more information.

Featured in: Example 4 on page 1072

See also: "LEGEND Statement" on page 225 and SUBGROUP= on page 1033 option

LEVELS=*number-of-midpoints*|ALL
specifies the number of midpoints for a numeric chart variable. The range for each midpoint is calculated automatically, using the algorithm in Terrell and Scott (1985).

If you specify LEVELS=ALL, then all unique midpoint values are graphed. If your data contains a large number of unique midpoint values (over 200), you can use the XPIXELS and YPIXELS GOPTIONS to enable the device driver to render a larger (and more readable) graph.

The LEVELS= 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.

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 is of the form (*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, or 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.

Style reference: LineStyle attribute of the GraphGridLines element.

Alias: LR=

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.

The Java and ActiveX devices do not support all AXIS statement options. See "AXIS Statement" on page 198 for more information.

Featured in: Example 4 on page 1072

Restriction: Partially supported by Java and ActiveX

See also: "AXIS Statement" on page 198 and "About Midpoints" on page 998

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.

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

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.

Featured in: Example 6 on page 1078

Restriction: Not supported by Java and ActiveX

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 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 ORDER= option on page 205 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 may be unreadable. In this case the procedure writes a warning to the SAS log. On many devices, this problem can be corrected by either adjusting the size of the text with the HTEXT= graphics option or by increasing the number of cells in your graphics display 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 MIDPOINTS= and ORDER= in the AXIS statement.

Featured in: Example 5 on page 1075

See also: "About Midpoints" on page 998

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 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='*entry-name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default name is GCHART. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name-for example, GCHART1.

See also: "About Filename Indexing" on page 99

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

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 then 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 axis frame or the three-dimensional backplane, 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. The NOLEGEND option is ignored if the SUBGROUP= option is not used.

NOPLANE
removes either the horizontal or vertical three-dimensinal 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 backplane, use the NOFRAME option in the procedure.

This option is not supported by the GRADAR Procedure.

Featured in: "Example 7. Using BY-group Processing to Generate a Series of Charts" on page 309.

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

NOZERO
suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar. The NOZERO option usually is used with the GROUP= option to suppress 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: If a bar is omitted and if you have also specified bar labels with the VALUE= option in an AXIS statement, then the labels can be shifted and not displayed with the correct bar. △

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

OUTSIDE=*statistic*
displays the values of the specified statistic above the bars. For the Java and ActiveX devices, this option is valid for both horizontal and vertical bar charts. 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 INSIDE=.

Statistic can be one of the following:

- FREQ
- CFREQ
- PERCENT | PCT
- CPERCENT | CPCT
- SUM
- MEAN

Featured in: Example 4 on page 1072 and "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP
specifies the way 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. △

Featured in: Example 4 on page 1072 and "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

See also: "Controlling Bar Chart Patterns, Colors, and Images" on page 1037

PERCENT
prints the percentages of observations having a given value for the chart variable 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 or CFREQ option is specified.

Alias: PCT

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

PERCENTSUM
calculates a percent of the sum variable for horizontal bar charts. The PERCENTSUM option is ignored if the SUMVAR= option is not specified.

Alias: PCTSUM

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

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

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 (<), and the ending value is indicated with the greater-than-or-equal-to symbol (>=). 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= option on page 1029. If specified, the DISCRETE option on page 1022 overrides the RANGE option.

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= option on page 1029 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 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 or if there is an ORDER= list in the AXIS statement assigned to the response axis.

The Java and ActiveX devices do not support all AXIS statement options. See "AXIS Statement" on page 198 for more information.

Featured in: Example 4 on page 1072 and "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

Restriction: Partially supported by Java and ActiveX

See also: "AXIS Statement" on page 198

REF=*value-list*

draws reference lines at the specified points on the response axis. See the MIDPOINTS= option on page 1029 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

Featured in: "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

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 non-negative number, including decimal values. Units are character cells. By default, the GCHART procedure calculates spacing based on the size of the axis area and the number of 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.

Featured in: Example 4 on page 1072 and "Example: Creating Bar Charts with Drill-Down for the Web" on page 618

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.

Featured in: Example 4 on page 1072, "Example: Creating Bar Charts with Drill-Down for the Web" on page 618 and Example 5 on page 1075

See also: "LEGEND Statement" on page 225

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.

See also: "About Chart Statistics" on page 1000, "Displaying Statistics in Horizontal Bar Charts" on page 1036, and "Displaying Statistics in Vertical Bar Charts" on page 1036

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=*summary-variable*

specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of *summary-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.

Featured in: Example 3 on page 1070 and Example 6 on page 1078

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

Because you cannot use TYPE=FREQ, TYPE=CFREQ, TYPE=PERCENT, or TYPE=CPERCENT 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 1001.

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 SUBGROUP= on page 1033.

Featured in: Example 6 on page 1078

See also: "About Chart Statistics" on page 1000 for a complete description of statistic types

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.

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;
```

If the requested bar width results in a chart that is too large to fit in the space available for the midpoint axis, then 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.

Featured in: Example 4 on page 1072

WOUTLINE=*bar-outline-width*
specifies the width of the outline in pixels. The WOUTLINE= option affects both the bar and the subgroup outlines.

Style reference: LineThickness attribute of the GraphOutlines element.

Restriction: Not supported by Java

WREF=*reference-line-width*
specifies line widths for reference lines. Line widths are specified as whole numbers. To specify colors for these reference lines, use the CREF= option.

Style reference: LineThickness attribute of the GraphReference element.

Restriction: Not supported by Java

The Chart Statistic and the Response Axis

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 "AXIS Statement" on page 198 for a complete discussion.

Displaying Statistics in Horizontal Bar Charts

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 SUMVAR=, 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 HTEXT= and FTEXT= graphics options.

Displaying Statistics in Vertical Bar Charts

Statistic values on vertical bar 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 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, 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 Java and ActiveX devices, 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. 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 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. Remember that although changing the number of midpoints for numeric variables can change the range of values for individual midpoints, it does not change the range of values for the chart as a whole. For details, see "About Midpoints" on page 998.

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 ORDER= option on page 205.

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 will use 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 1002 for more information on 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= option on page 242 in "PATTERN Statement" on page 240.

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= option (see COUTLINE=).

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, and 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 PATTERNID= option on page 1031 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, action 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 backplane frame of two-dimensional bar charts developed with the HBAR and VBAR statements. In three–dimensional bar charts, you can apply images to the backplane frame. For details, see “Specifying Images in SAS/GRAPH Programs” on page 181.

PIE, PIE3D, and DONUT Statements

Create pie or donut charts in which 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.

Requirements: At least one chart variable is required.

Global statements: LEGEND, PATTERN, TITLE, FOOTNOTE

Supports: Drill-down functionality

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, including the content and position of the slice labels, and 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 display 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.

Syntax

PIE | PIE3D | DONUT *chart-variable(s)* </ *option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options

  ANNOTATE=*Annotate-data-set*
  CFILL=*fill-color*
  COUTLINE=*slice-outline-color* | SAME
  DETAIL_RADIUS=*percent* (PIE and DONUT only)
  EXPLODE=*value-list*
  FILL=SOLID | X
  INVISIBLE=*value-list*
  NOHEADING
  RADIUS=
  WOUTLINE=*slice-outline-width*

- statistic options

  FREQ=*numeric-variable*
  SUMVAR=*summary-variable*
  TYPE=*statistic*

- midpoint options

  DISCRETE
  LEVELS=*number-of-midpoints*|ALL
  MIDPOINTS=*value-list*
  MIDPOINTS=OLD
  MISSING

OTHER=*percent-of-total*

- detail pie options (PIE and DONUT only)
  - DETAIL=*variable*
  - DETAIL_THRESHOLD=*percent*
- grouping and subgrouping options
  - ACROSS=*number-of-columns*
  - DOWN=*number-of-rows*
  - GROUP=*group-variable*
  - NOGROUPHEADING
  - SUBGROUP=*subgroup-variable*
- slice-ordering options
  - ANGLE=*degrees*
  - ASCENDING
  - CLOCKWISE
  - DESCENDING
  - JSTYLE
- slice-labeling options
  - CTEXT=*text-color*
  - LEGEND | LEGEND=LEGEND<1...99>
  - MATCHCOLOR
  - NOLEGEND
  - OTHERLABEL='*text-string*'
  - PERCENT=ARROW | INSIDE | NONE | OUTSIDE
  - PLABEL=(*text argument(s)*)
  - SLICE=ARROW | INSIDE | NONE | OUTSIDE
  - VALUE=ARROW | INSIDE | NONE | OUTSIDE
- detail pie slice-labeling options (PIE and DONUT only)
  - DETAIL_PERCENT=BEST|NONE
  - DETAIL_SLICE=BEST|NONE
  - DETAIL_VALUE=BEST|NONE
- donut-labeling options (DONUT only):
  - DONUTPCT=*percent*
  - LABEL=(*text argument(s)*)
- catalog entry description options
  - DESCRIPTION='*entry-description*'
- ODS options
  - HTML=*variable*
  - HTML_LEGEND=*variable*

Required Arguments

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 also: "About Chart Variables" on page 997

Options

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 on specifying colors, see Chapter 12, "SAS/GRAPH Colors and Images," on page 167. For a complete description of the graphics options, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

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 charts produced by the PIE, PIE3D, or DONUT statement.

Note: Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with pie or donut charts. △

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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.

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, if no pattern is specified in the pattern statement or with the FILL= option, 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, or, if the NOGSTYLE option is specified, it 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.

Featured in: Example 9 on page 1084

See also: "Controlling Bar Chart Patterns, Colors, and Images" on page 1037 and "About Patterns" on page 1002

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 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.

Style reference: Color attribute of the GraphOutlines element.

Featured in: Example 7 on page 1080, Example 8 on page 1083. and

See also: "Controlling Slice Patterns and Colors" on page 1053 and "About Patterns" on page 1002

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 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 the PIE/DONUT statement
2. the color specified by the CTEXT= option in the PIE/DONUT statement
3. the color specified by the CTEXT= option in a GOPTIONS statement
4. 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 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.

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. △

Style reference: Color attributes of the GraphValueText and the GraphLabelText elements.

Featured in: Example 8 on page 1083.

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 the description of the catalog entry for the chart. The maximum length for *entry-description* is 256 characters. The description does not appear on the chart. By default, the GCHART procedure assigns a description of the form PIE (or PIE3D or DONUT) CHART OF *variable*, where *variable* is the name of the chart variable.

The *entry-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 294. 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.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window.
- the catalog-entry properties that you can view from the Explorer window.
- the Description field of the PROC GREPLAY window.
- the data tip text for the entire chart area for web output (depending on the device driver you are using). See "Data Tips for Web Presentations" on page 598 for details.

Alias: DES=

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. 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 associated with it, each formatted value is treated as a midpoint.

The LEVELS= option is ignored when you use DISCRETE. The MIDPOINTS= option overrides DISCRETE.

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.

Featured in: Example 8 on page 1083

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*
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. See the MIDPOINTS= option on page 1047 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.

When you use EXPLODE=, the radius is reduced to allow room for exploded slices.

When used with subgroups, the EXPLODE= option is supported only by the ActiveX and Java devices.

Featured in: Example 7 on page 1080

FILL=SOLID | X
specifies the fill pattern for *all* slices in the chart:

SOLID S
rotates a solid fill through the color list of the current style as many times as necessary. SOLID is the default.

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, or the first color in the device's color list if the NOGSTYLE option is specified. 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.

Restriction: Partially supported by Java and ActiveX

See also: "Controlling Bar Chart Patterns, Colors, and Images" on page 1037 and "PATTERN Statement" on page 240

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 also: "Calculating Weighted Statistics" on page 1001

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 except 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.

Featured in: Example 10 on page 1086

See also: "BY Statement" on page 216

HTML=*variable*

identifies the variable in the input data set whose values create links in the HTML file that is created by the ODS statement. These links are associated with an area of the chart and point to the data or graph that you want to display when the user drills down on the area.

See also: "Overview of Enhancing Web Presentations" on page 596.

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links in the HTML file created by the ODS statement. These links are associated with a legend value and point to the data or graph that you want to display when the user drills down on the value. The values of *variable* can be up to 1024 characters long. Characters after the 1024-character limit (including any closing quotes) are truncated. If either subgroups or the DETAIL= option are specified, then the HTML_LEGEND= option is ignored.

Restriction: Not supported by Java and ActiveX.

See also: "Overview of Enhancing Web Presentations" on page 596.

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= option on page 1047 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.

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*

The Java and ActiveX devices do not support all of the suboptions. See "Text Description Suboptions for Donut" on page 1050 for a complete description.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Featured in: Example 8 on page 1083

Restriction: Partially supported by Java and ActiveX

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.

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. △

Specifying LEGEND=LEGEND*n* assigns the specified LEGEND statement to the legend. The Java and ActiveX devices do not support all LEGEND statement options. See "LEGEND Statement" on page 225 for more information.

Featured in: Example 8 on page 1083

Restriction: Partially supported by Java and ActiveX

See also: "LEGEND Statement" on page 225.

LEVELS=*number-of-midpoints*|ALL
specifies the number of midpoints for a numeric chart variable. The range for each midpoint is calculated automatically.

If you specify LEVELS=ALL, then all unique midpoint values are graphed. If your data contains a large number of unique midpoint values (over 200), you can use the XPIXELS and YPIXELS GOPTIONS to enable the device driver to render a larger (and more readable) graph.

The LEVELS= 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 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 ORDER= option on page 205 in the AXIS statement.

Midpoints that represent small percentages are collected into a generic midpoint named OTHER. See the OTHER= option on page 1048 and the OTHERLABEL= option on page 1048 for more information.

Featured in: Example 9 on page 1084

See also: "About Midpoints" on page 998

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='*entry-name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default name is GCHART. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name-for example, GCHART1.

See also: "About Filename Indexing" on page 99

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.

Featured in: Example 8 on page 1083

Restriction: Not supported by Java and ActiveX

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 percent or less of the total is put in the OTHER category.

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. △

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.

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= 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

For more information, see "Controlling Slice Patterns and Colors" on page 1053.

Style reference: Color attribute of the GraphData1 to GraphDataN element, depending on the number of slices in the pie.

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 1052. By default, PERCENT=NONE (percentage is not displayed).

Whether the slice percent displays 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 or 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.

Featured in: Example 9 on page 1084 and Example 10 on page 1086

PLABEL=(*text argument(s)*)
defines the text that is displayed on the pie slice label. *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.

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

COLOR=*color*

FONT=*font*

HEIGHT=*text-height* <*units*>
The Java and ActiveX devices do not support all of the suboptions. See "Text Description Suboptions for Donut" on page 1050 for a complete description.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Style reference: Font Attributes of the GraphValueText element.

RADIUS=*value*
specifies the radius of the pie and donut in GCHART. RADIUS=*n*, where *n* 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. For a description of the option values, see "Selecting and Positioning Slice Labels" on page 1052. By default, SLICE=OUTSIDE (the name is outside of the slice).

Featured in: Example 9 on page 1084 and Example 10 on page 1086

SUBGROUP=*subgroup-variable*
divides the chart into concentric rings according to the values of *subgroup-variable*. For DEVICE=JAVA, subgroups are implemented using drill-down functionality instead of concentric rings. In the resulting graph, you can select a pie slice to display subgroup information. *Subgroup-variable* can be either character or numeric and is always treated as a discrete 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. Any inner subgroup that contains a value that is not in the outer subgroup, places the new slice for that value 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 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.

If EXPLODE is also used, it is ignored.

Featured in: Example 8 on page 1083 and Example 9 on page 1084

See also: "Controlling Bar Chart Patterns, Colors, and Images" on page 1037 and "LEGEND Statement" on page 225

SUMVAR=*summary-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 SUMVAR=, the TYPE= option must be either SUM or MEAN. With SUMVAR=, the default is TYPE=SUM.

Featured in: Example 7 on page 1080

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.

See also: "About Chart Statistics" on page 1000 and "Calculating Weighted Statistics" on page 1001

VALUE=ARROW | INSIDE | NONE | OUTSIDE

controls the position and style of the slice value (chart statistic) for each slice. For a description of the option values see "Selecting and Positioning Slice Labels" on page 1052. By default, VALUE=OUTSIDE (the value is outside the slice).

Featured in: Example 9 on page 1084n

WOUTLINE=*slice-outline-width*

specifies the width of the outline in pixels. WOUTLINE= affects both the slice 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 the PLABEL= option in the PIE statement uses text description suboptions to change the attributes of the following text string or strings that follow the suboption.

ANGLE=*degrees*
specifies the angle at which the baseline of the text string(s) 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).

Alias: A=degrees

Valid in: DONUT

Restriction: Not supported by Java

COLOR=*color*
specifies the color for the text string(s). The COLOR= suboption stays in effect until another COLOR= specification is encountered. If you omit COLOR=, LABEL= uses the color defined by the current style. It ignores the CTEXT= graphics option. See Chapter 12, "SAS/GRAPH Colors and Images," on page 167 for details on specifying *color*.

Alias: C=color

Valid in: DONUT, PIE, PIE3D

Restriction: Not supported by Java

FONT=*font*
F=*font*
specifies the font for the text string or strings. If you omit FONT=, LABEL= uses the font that is specified by the FTEXT= graphics option. If no font is specified, it uses the default hardware font, NONE. See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 for details on specifying *font*. The Java and ActiveX devices do not support all fonts.

Alias: F=font

Valid in: DONUT, PIE, PIE3D

Restriction: Partially supported by Java and ActiveX

HEIGHT=*text-height <units>*
specifies the height of the text string or strings. *Text-height* is the number of units. If you omit HEIGHT=, LABEL= 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*, HEIGHT= uses the unit that is specified by the GUNIT= graphics option, or the default unit, CELLS.

Alias: H=text-height <units>

Valid in: DONUT, PIE, PIE3D

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=LEFT

Restriction: Not supported by Java and ActiveX

ROTATE=*degrees*
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

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.

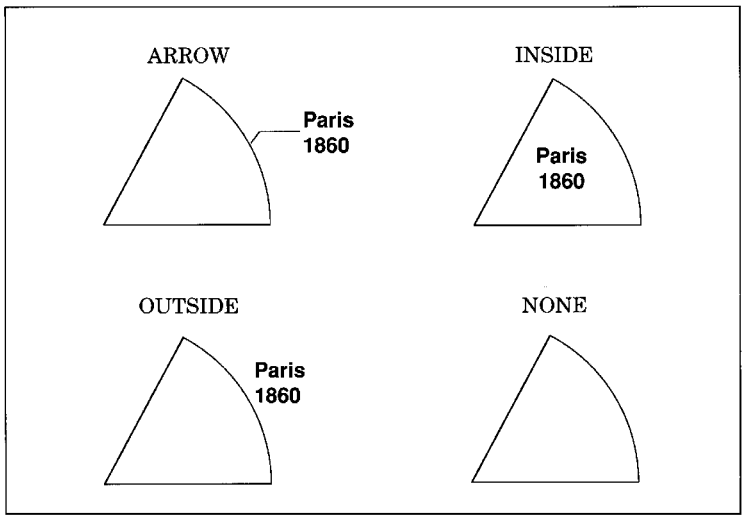
INSIDE
places the text inside the slice. The label overlays the slice fill patterns. This labeling method increases the radius of the pie.

NONE
suppresses the text.

OUTSIDE
places the text outside of the slice.

Figure 36.14 on page 1052 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. PERCENT= 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 59 for details.)

Controlling 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 will use 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 1002 for more information on 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 the VALUE= option on page 245 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, whereas the CPATTERN= graphics option and a COLORS= list of one color skip the solid pattern and, beginning with P2N0, use each pie hatch pattern with the specified color, and use the fill color for the outline color.
  - specify only COLOR= 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. 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 of the defined patterns are exhausted.

See “About Patterns” on page 1002 for more information on how the GCHART procedure uses patterns and outlines. See “PATTERN Statement” on page 240 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 example, *SUM of SALES by SITE*. 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;
```

Because the AXIS statement cannot be used by the PIE, PIE3D, and DONUT statements, you should use the FTEXT= and HTEXT= graphics options 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.

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.

Requirements: At least one chart variable is required.

Global statements: FOOTNOTE, PATTERN, TITLE,

Supports: Drill-down functionality (slices only)

Restriction: Not supported by Java and ActiveX

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, to change the type of chart statistic, and to modify the appearance of the chart, including 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 display 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.

Syntax

STAR *chart-variable(s)* </ *option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options

  ANGLE=*degrees*

  ANNOTATE=*Annotate-data-set*

  ASCENDING*Annotate-data-set*

  CFILL=*fill-color*

COUTLINE=*star-outline-color* | SAME
DESCENDING
FILL=SOLID | X
LEGEND=LEGEND<1...99>
NOCONNECT
NOLEGEND
NOSPINE
STARMAX=*max-value*
STARMIN=*min-value*
WOUTLINE=*slice-outline-width*

□ statistic options
FREQ=*numeric-variable*
SUMVAR=*summary-variable*
TYPE=*statistic*

□ midpoint options
DISCRETE
LEVELS=*number-of-midpoints*
MIDPOINTS=*value-list*
MIDPOINTS=OLD
MISSING

□ grouping options
ACROSS=*number-of-columns*
DOWN=*number-of-rows*
GROUP=*group-variable*

□ labeling options
CTEXT=*text-color*
MATCHCOLOR
NOGROUPHEADING
NOHEADING
PERCENT=ARROW | INSIDE | NONE | OUTSIDE
SLICE=ARROW | INSIDE | NONE | OUTSIDE
VALUE=ARROW | INSIDE | NONE | OUTSIDE

□ catalog entry description options
DESCRIPTION='*entry-description*'
NAME='*entry-name*'

□ ODS options
HTML=*variable*
HTML_LEGEND=*variable*

Required Arguments

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 also: "About Chart Variables" on page 997

Options

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 on specifying colors, see Chapter 12, "SAS/GRAPH Colors and Images," on page 167.

ACROSS=*number-of-columns*
draws *number-of-columns* stars across the procedure output area. ACROSS= 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 charts that are produced by the STAR statement.

Note: Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with star charts. △

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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.

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. If no pattern is specified in the pattern statement or with the FILL= option, the procedure starts with the default solid fill and then, beginning with P2N0, uses each default star hatch pattern with the specified color. For the outline color, the procedure uses the default color, which is retrieved from the current style, or, if the NOGSTYLE option is specified, it 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 style. 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), and 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.

Style reference: Color attribute of the GraphOutlines element.

Featured in: Example 12 on page 1089

See also: "Selecting Patterns for the Star Charts" on page 1064 and "About Patterns" on page 1002

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 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 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 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.

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. △

Style reference: Color attributes for the GraphLabelText and the GraphValueText elements.

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 with the MIDPOINTS= option or that is specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

DESCRIPTION='description*n*'
specifies the description of the catalog entry for the chart. The maximum length for *entry-description* is 256 characters. The description does not appear on the chart. By default, the GCHART procedure assigns a description of the form STAR CHART OF *variable*, where *variable* is the name of the chart variable.

The *entry-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 294. 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.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window
- the catalog-entry properties that you can view from the Explorer window
- the Description field of the PROC GREPLAY window
- the data tip text for Web output (depending on the device driver you are using). See "Data Tips for Web Presentations" on page 598 for details.

Alias: DES=

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 associated with it, each format value is treated as a separate value.

The LEVELS= option is ignored when you use the DISCRETE option. The MIDPOINTS= option overrides the DISCRETE option.

Featured in: Example 12 on page 1089

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
S
: rotates a solid fill through the list of colors available in the default style as many times as necessary. SOLID is the default.

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, or the first color in the device's color list if the NOGSTYLE option is specified. 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.

Featured in: Example 12 on page 1089

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 also: "Calculating Weighted Statistics" on page 1001

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 except 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 in the HTML file that is created by the ODS statement. These links are associated with a legend value and point to the data or graph that you want to display when the user drills down on the value. The values of variable can be up to 1024 characters long. Characters after the 1024-character limit (including any closing quotes) are truncated.

See also: "Overview of Enhancing Web Presentations" on page 596.

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links in the HTML file that is created by the ODS statement. These links are associated with an area of the chart and point to the data or graph that you want to display when the user drills down on the area. Only star charts with slices support drill-down functionality. There is no limit on the length of the variable.

See also: "Overview of Enhancing Web Presentations" on page 596.

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;
```

The Java and ActiveX devices do not support all LEGEND statement options. See "LEGEND Statement" on page 225 for more information.

Featured in: Example 2 on page 1067

Restriction: Partially supported by Java ActiveX

See also: SUBGROUP= on page 1013 and "LEGEND Statement" on page 225

LEVELS=*number-of-midpoints*
specifies number of midpoints for a numeric chart variable. The range for each midpoint is calculated automatically using the algorithm described by Terrell and Scott (1985). The LEVELS= 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 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 ORDER= option on page 205 in the AXIS statement.

See also: "About Midpoints" on page 998

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='*entry-name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default name is GCHART. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, GCHART1.

See also: "About Filename Indexing" on page 99

NOCONNECT
draws only star spines without connecting lines. By default, the spines are connected to form slices.

Featured in: Example 12 on page 1089

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.

Featured in: Example 12 on page 1089

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 1063. 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 1063. 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.

SUMVAR=*summary-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 SUMVAR=, the default is TYPE=SUM.

Featured in: Example 11 on page 1088

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 also: "About Chart Statistics" on page 1000 and "Calculating Weighted Statistics" on page 1001

VALUE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice value (chart statistic) for each slice. For a description of the option values, see "Selecting and Positioning Spine and Slice Labels" on page 1063. By default, VALUE=OUTSIDE (the value is outside of the slice).

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.

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 36.14 on page 1052 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

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 1002 for more information on 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 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 VALUE= option on page 245 in "PATTERN Statement" on page 240.

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. If you use the CFILL= option, the procedure starts with the default solid color and uses the foreground color for outlines. If you use CPATTERN= or a COLORS= list of one color, the procedure skips the default solid fill and, beginning with P2N0, uses each default star hatch pattern with the specified color, and uses 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 1002 for more information on how the GCHART procedure uses patterns and outlines. See "PATTERN Statement" on page 240 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 example, **SUM of SALES by SITE**. 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—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, as shown in this example:

```
label site="00"x;
```

Because the AXIS statement cannot be used by the STAR statement, you should use the FTEXT= and HTEXT= 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. For a description of these graphics options, see Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327.

Examples

Note: 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 HTML;** ***procedure-code;*** **ODS HTML CLOSE;**). See "RUN-Group Processing" on page 56 for more information. △

Example 1: Specifying the Sum Statistic in a Block Chart

Procedure features:

BLOCK statement option:

SUMVAR=

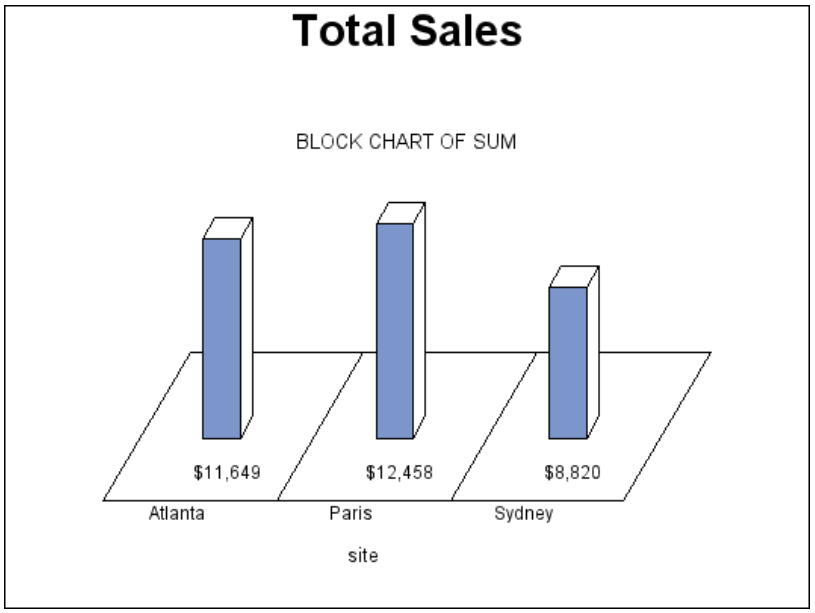
Other features:

FORMAT statement

GOPTIONS statement option:

BORDER

Sample library member: GCHBKSUM



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.

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 title and footnote.

```
title "Total Sales";
footnote j=r "GCHBKSUM ";
```

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 summary 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

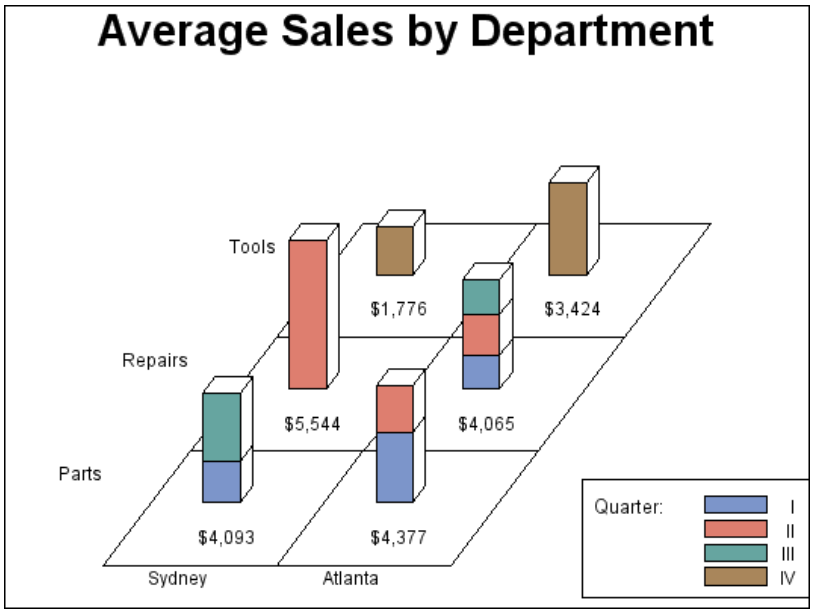
Procedure features:

BLOCK statement options:

CAXIS=
GROUP=
LEGEND=
MIDPOINTS=
NOHEADING
SUBGROUP=
TYPE=

Other features:
GOPTION statement option:
BORDER
LABEL statement
LEGEND statement
Default pattern rotation

Sample library member: GCHBKGRP



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 which 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.

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 title and footnote.

```
title "Average Sales by Department";
footnote j=r "GCHBKGRP ";
```

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 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

Procedure features:

HBAR statement options:

SUMVAR=

VBAR3D statement options:

SUMVAR=

Other features:

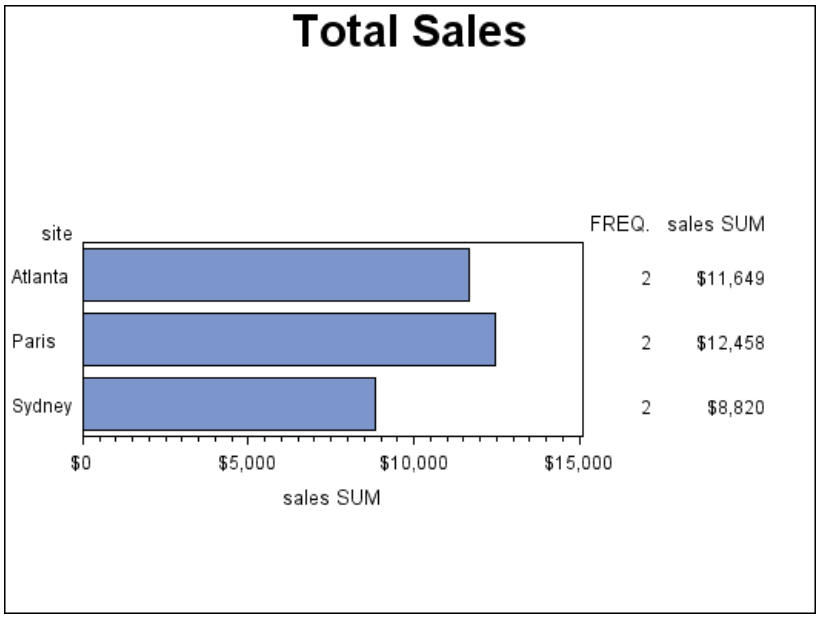
FORMAT statement

GOPTIONS statement option:

BORDER

RUN-group processing

Sample library member: GCHBRSUM

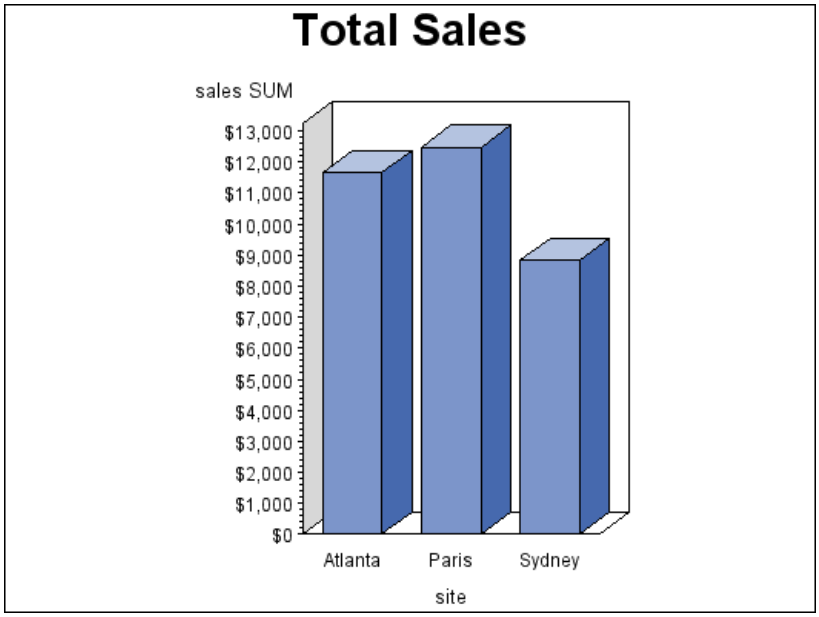


This example produces two bar charts that show the total sales for three sites by charting the values of the character variable SITE and calculating the sum of the variable SALES for each site.

In the horizontal bar chart shown above, 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.



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 title and footnote for the first chart.

```
title1 "Total Sales";
footnote1 j=r "GCHBRSUM(a)";
```

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 summary variable SALES is assigned a dollar format.

```
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. The FOOTNOTE1 statement replaces the previous footnote.

```
  vbar3d site / sumvar=sales;
footnote1 j=r "GCHBRSUM(b)";
run;
quit;
```

Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart

Procedure features:

VBAR statement options:

CFRAME=
INSIDE=SUBPCT
LEGEND=
MAXIS=
OUTSIDE=SUM
RAXIS=
SPACE=
SUBGROUP=
WIDTH=

Other features:

AXIS statement

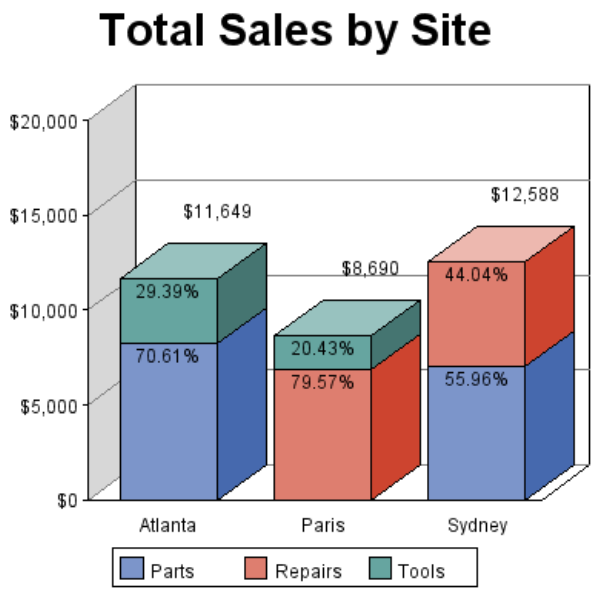
FORMAT statement

GOPTIONS statement option:

BORDER

LEGEND statement

Sample library member: GCHBRGRP



This example subgroups by department the three-dimensional vertical bar chart of total sales for each site that is shown in Example 3 on page 1070. 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 of each subgroup segment.

Both the LEGEND statement and the AXIS statement use the ORIGIN= option to line up the legend and the chart by explicitly positioning their lower left corners.

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
Tools Paris 4 1775.74
Tools Atlanta 4 3424.19
Repairs Sydney 2 5543.97
Repairs Paris 3 6914.25
;
```

Define title and footnote.

```
title1 "Total Sales by Site";
footnote1 j=r "GCHBRGRP ";
```

Modify the midpoint axis. The LABEL= option suppresses the axis label. The ORIGIN= option positions the left end of the horizontal axis at a point that is 25% of the width of the graphics output area.

```
axis1 label=none
origin=(24,);
```

Modify the response axis. The ORDER= option specifies the major tick values for the response axis. The OFFSET= option moves the top tick mark to the end of the axis line.

```
axis2 label=none
order=(0 to 30000 by 5000)
minor=(number=1)
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. The ORIGIN= option specifies the same position as in the AXIS1 statement.

```
legend1 label=none
shape=bar(3,3)
cborder=black
origin=(24,);
```

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 LEGEND= option assigns the LEGEND1 statement to the subgroup legend. The CFRAME= option specifies the color for the three-dimensional planes.

```
proc gchart data=totals;
format quarter roman.;
format sales dollar8.;
vbar3d site / sumvar=sales subgroup=dept inside=subpct
     outside=sum
     width=9
     space=4
     maxis=axis1
     raxis=axis2
     cframe=gray
     legend=legend1;
run;
quit;
```

Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart

Procedure features:

HBAR statement options:

AUTOREF
COUTLINE=
CLIPREF
SUBGROUP=

HBAR3D statement options:

FREQ
FREQLABEL=
MIDPOINTS=

Other features:

GOPTIONS statement option:

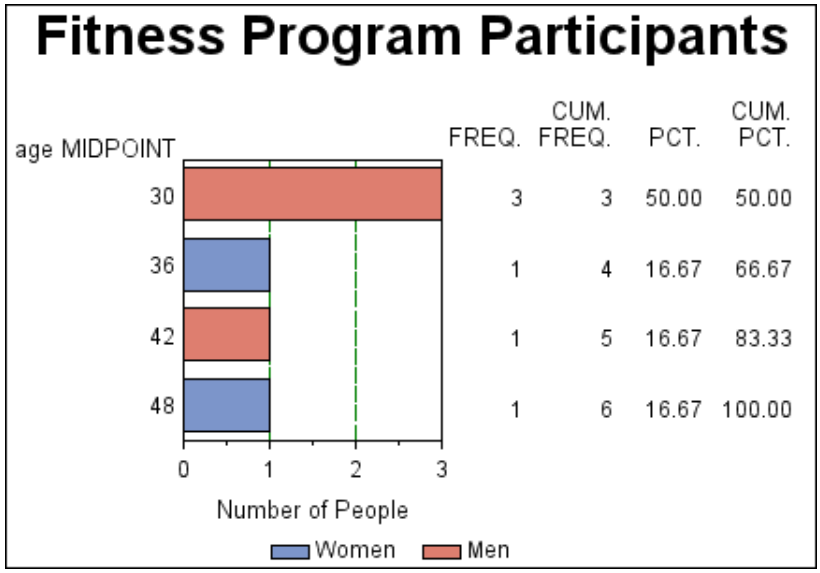
BORDER

AXIS statement

LEGEND statement

RUN-group processing

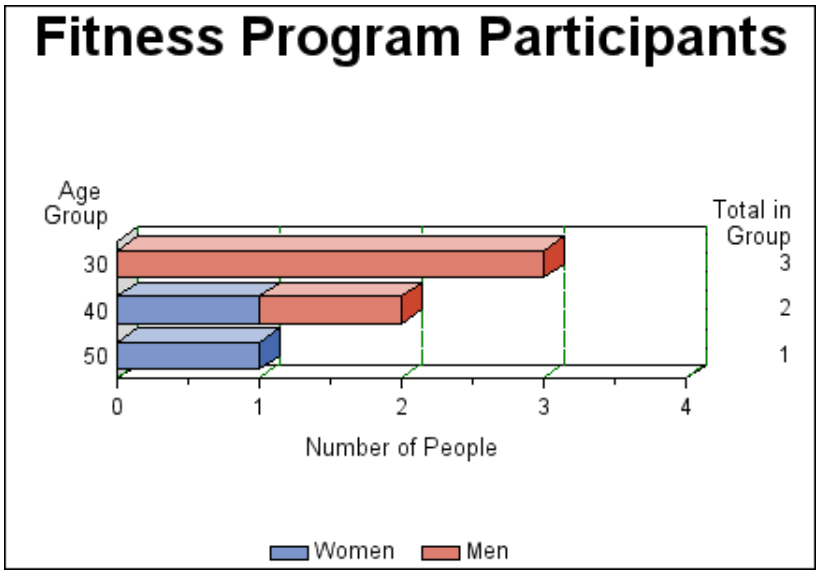
Sample library member: GCHBRMID



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.



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. The BORDER option draws a black border around the graph.

```
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 and footnote.

```
title1 "Fitness Program Participants";
footnote j=r "GCHBRMID(a)";
```

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");
```

Modify the width, color, and type of reference lines. The REF= option defines which reference lines will be highlighted using the type, color and width options. The WREF= option specifies the width of the reference line. The LREF= option specifies the type of reference line. The FREQ option requests that only the frequency statistic appears in the table. The FREQLABEL= option specifies the text for the column header in the table of statistics.

```
wref=(5 5);
lref=(2 1);
```

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.

```
proc gchart data=fitness;
hbar age / subgroup=sex legend=legend1 autoref
     clipref
     raxis=axis1;
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);
```

Define the footnote for the second chart.

```
footnote j=r "GCHBRMID(b)";
```

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 header 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
     ;
run;
quit;
```

Example 6: Generating Error Bars in a Horizontal Bar Chart

Procedure features:

HBAR statement options:

CLM=
ERRORBAR=
FREQLABEL=
MEANLABEL=
NOFRAME
SUMVAR=
TYPE=

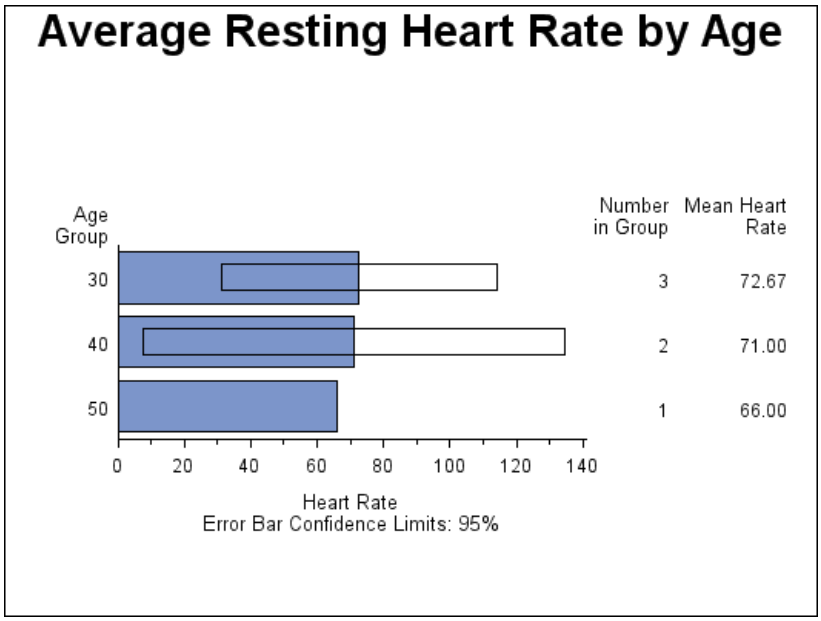
Other features:

GOPTIONS statement option:

BORDER

AXIS statement

Sample library member: GCHERRBR



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.

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. The BORDER option draws a black border around the graph.

```
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 and footnote.

```
title1 "Average Resting Heart Rate by Age";
footnote j=r "GCHERRBR";
```

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
clm=95
midpoints=(30 40 50)
raxis=axis1
maxis=axis2
noframe;
run;
quit;
```

Example 7: Specifying the Sum Statistic for a Pie Chart

Procedure features:

PIE statement options:

SUMVAR=

PIE3D statement options:

EXPLODE=
SUMVAR=

Other features:

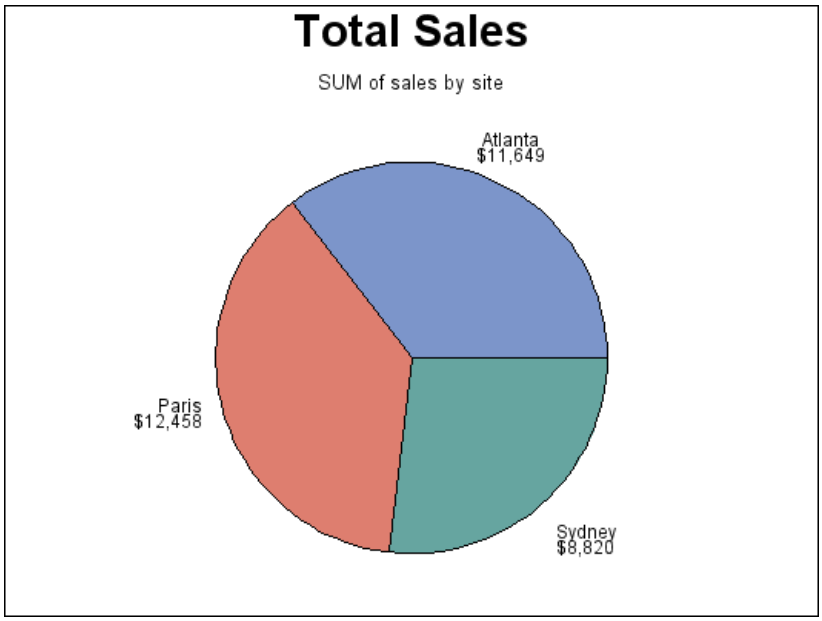
GOPTIONS statement option:

BORDER

FORMAT statement

RUN-group processing

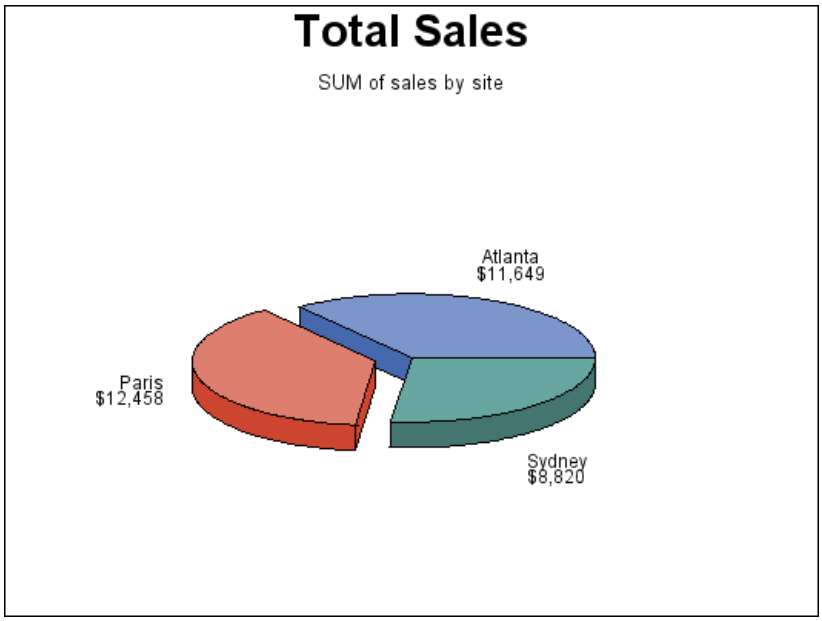
Sample library member: GCHPISUM



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.



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 title and footnote.

```
title "Total Sales";
footnote j=r "GCHPISUM(a) ";
```

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 summary variable SALES is assigned a dollar format.

```
proc gchart data=totals;
format sales dollar8.;
pie site / sumvar=sales;
run;
```

Define footnote for second pie chart.

```
   footnote j=r "GCHPISUM(b)";
```

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

Procedure features:

DONUT statement options:

DONUTPCT=
LABEL=
LEGEND=
NOHEADING
SUBGROUP=

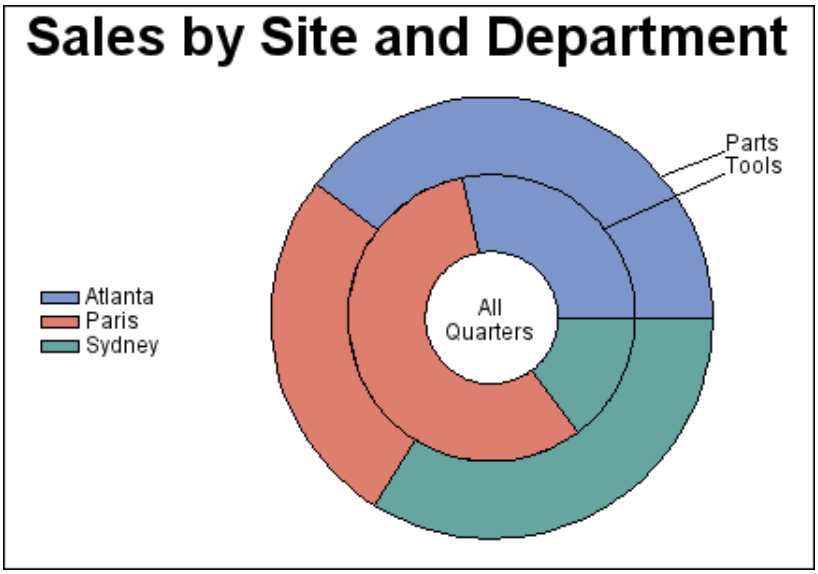
Other features:

GOPTIONS statement option:

BORDER

LEGEND Statement

Sample library member: GCHSBGRP



This example produces a donut chart that is similar to the pie chart in Example 7 on page 1080 in 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.

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 title and footnote.

```
title "Sales by Site and Department";
footnote j=r "GCHSBGRP ";
```

Modify the subgroup legend. The LABEL= options 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

Procedure features:

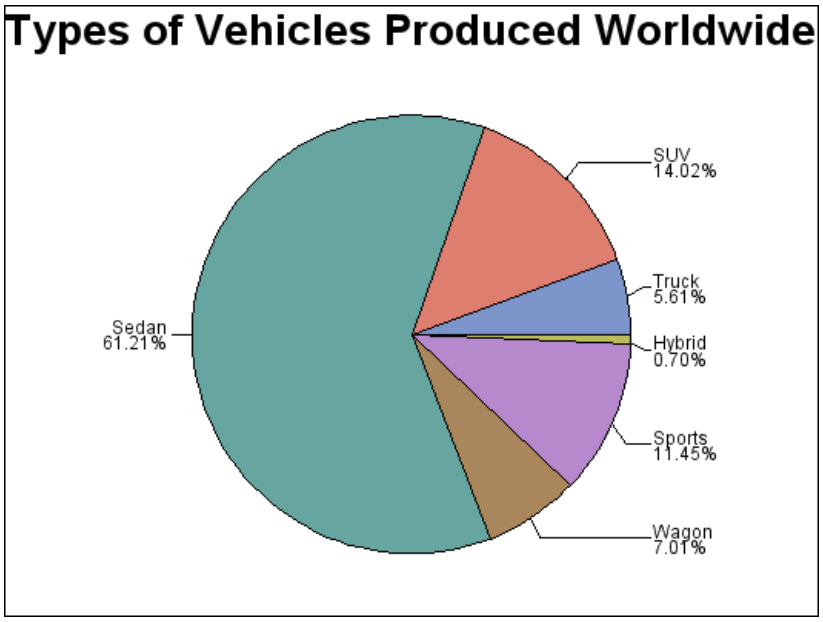
PIE statement options:

MIDPOINTS=
PERCENT=ARROW

SLICE=ARROW
VALUE=NONE

Other features:
GOPTIONS Statement option:
BORDER

Sample library member: GCHLABEL



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, which are arranged by the MIDPOINTS= option so that similar types of vehicles are shown next to each other in the pie chart.

Set the graphics environment.

```
goptions reset=all border;
```

Define title and footnote.

```
title "Types of Vehicles Produced Worldwide";
footnote j=r "GCHLABEL";
```

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

Procedure features:

PIE statement options:

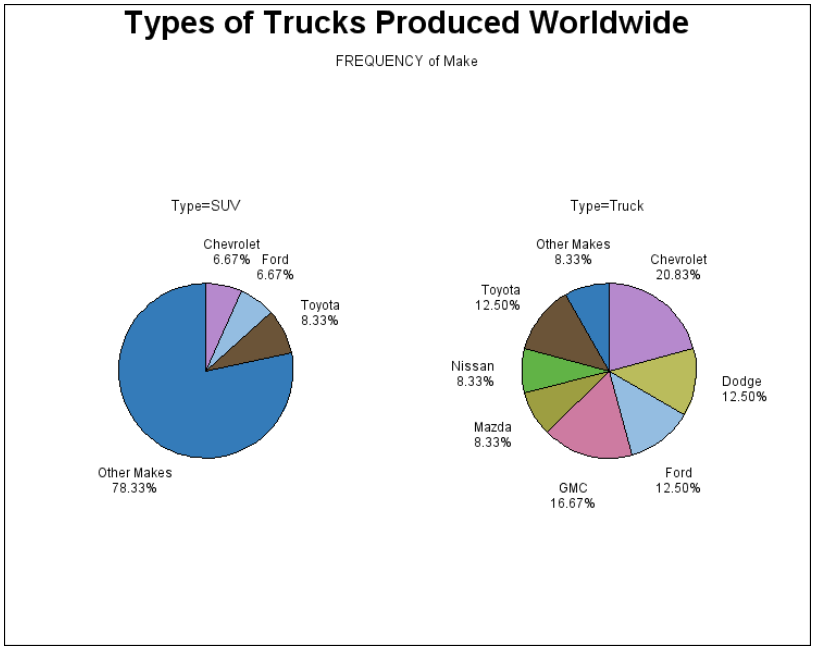
ACROSS=
CLOCKWISE
GROUP=
OTHER=
PERCENT=OUTSIDE
SLICE=OUTSIDE

Other features:

GOPTIONS statement option:

BORDER

Sample library member: GCHPIGRP



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.

Set the graphics environment.

```
goptions reset=all border;
```

Define title and footnote.

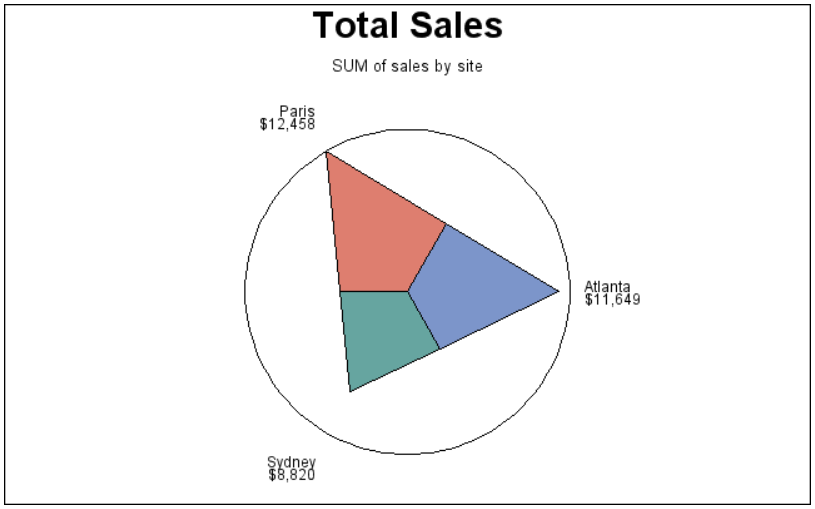
```
title "Types of Trucks Produced Worldwide";
footnote j=r "GCHPIGRP";
```

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 percent 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

Procedure features:
STAR statement options:
SUMVAR=
Other features: FORMAT statement
Sample library member: GCHSTSUM



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, and the midpoint value and the formatted values of the sales statistics are printed beside each slice.

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 title and footnote.

```
title "Total Sales";
footnote j=r "GCHSTSUM ";
```

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 summary 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

Procedure features:

STAR statement options:

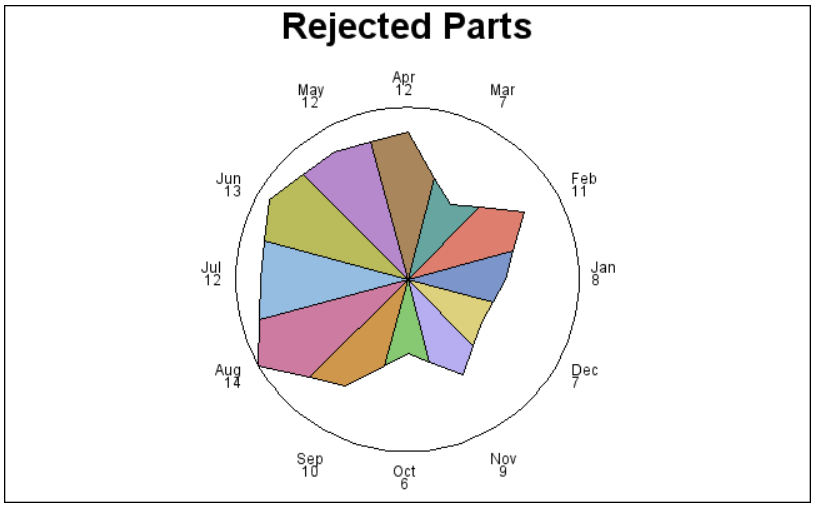
DISCRETE
FILL=
NOCONNECT
NOHEADING
SUMVAR=

Other features:

GOPTIONS statement option:

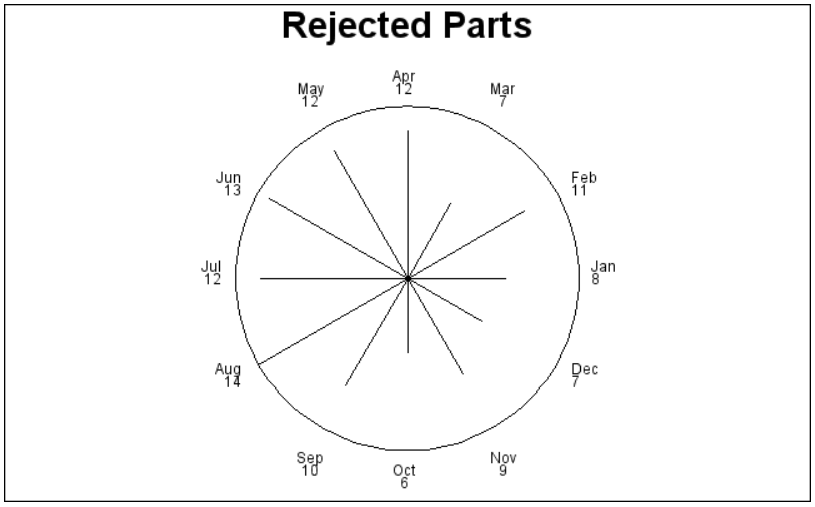
BORDER

Sample library member: GCHDSCRT



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.



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 title and footnote.

```
title "Rejected Parts";
footnote j=r "GCHDSCRT(a) ";
```

Produce the first star chart. DISCRETE must be specified because the numeric chart variable, DATE is assigned the WORDDATE3. Using FILL=S fills all the slices with solid patterns.

```
proc gchart data=rejects;
format date worddate3.;
star date / discrete
sumvar=badparts
noheading
fill=s;
run;
```

Define footnote for the second chart.

```
footnote j=r "GCHDSCRT(b) ";
```

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

Procedure Features:

PIE statement options:

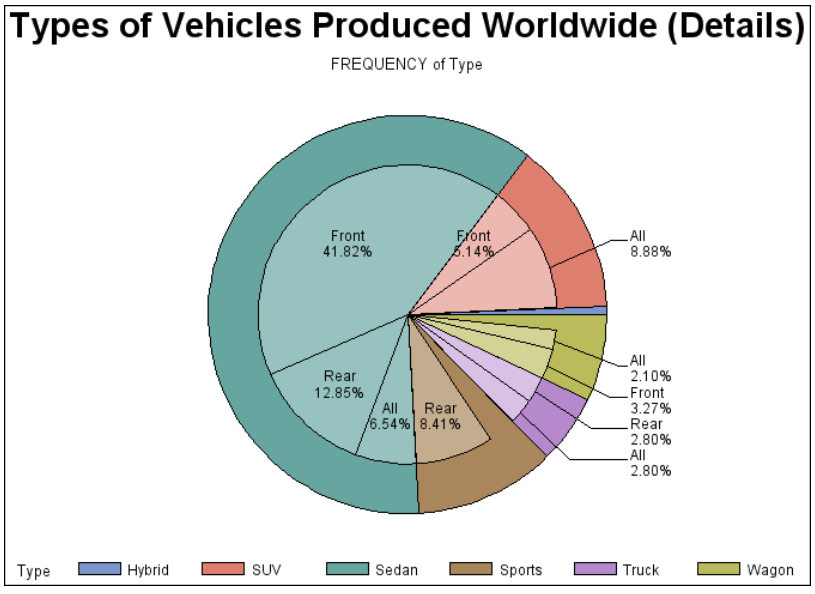
DETAIL=
DETAIL_PERCENT=
DETAIL_SLICE=
DETAIL_THRESHOLD=
DETAIL_VALUE=
LEGEND

Other features:

GOPTIONS statement option:

BORDER

Sample library member: GCHDTPIE



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.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote.

```
title "Types of Vehicles Produced Worldwide (Details)";
footnote1 j=r "GCHDTPIE";
```

Produce the detail pie chart. This graph uses the data set entitled CARS found in the SASHELP library. The DETAIL= option produces a 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 two percent of the entire pie.

```
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," *Applied Statistics, Volume 25, Number 1*, London: The Royal Statistical Society.

Terrell, G. R. and Scott, D. W. (1985), "Oversmoothed Nonparametric Density Estimates," *Journal of the American Statistical Association*, 80.

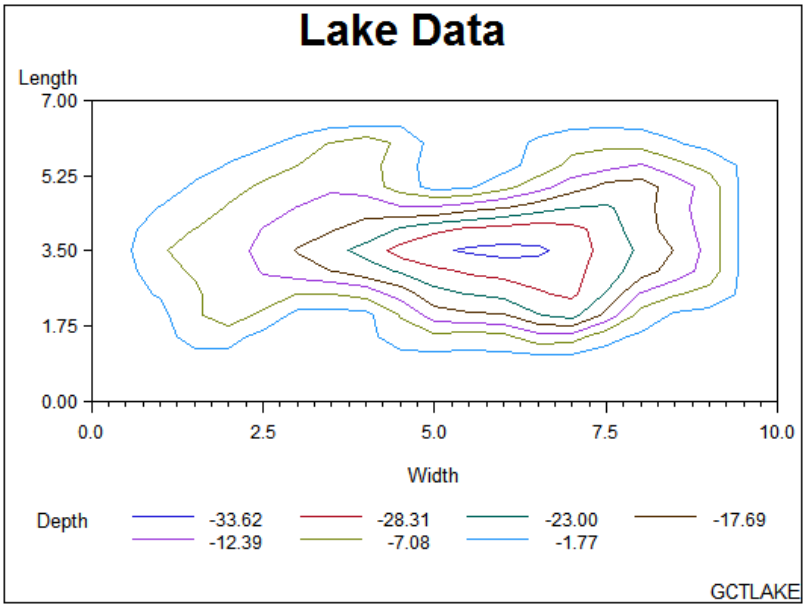
CHAPTER
37

The GCONTOUR Procedure

Overview

The GCONTOUR procedure enables you to generate two-dimensional plots representing three-dimensional relationships. For example, the following contour plot Display 37.1 on page 1096 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.

Display 37.1 Simple Contour Plot



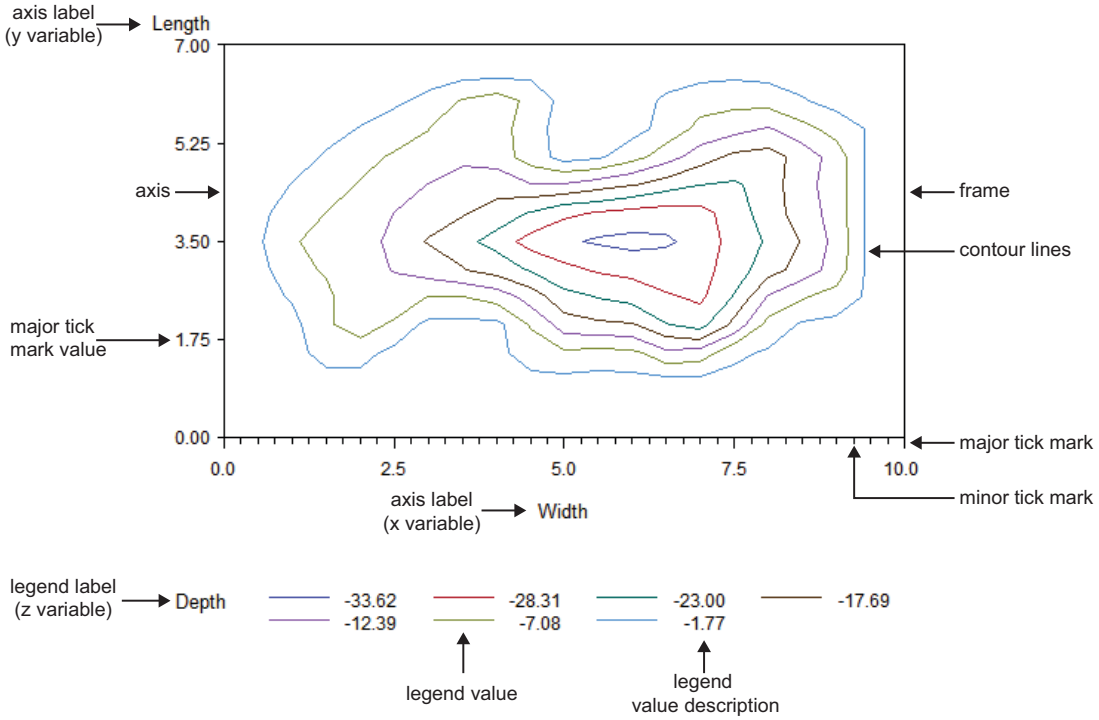
With PROC GCONTOUR, you can do the following actions:

- 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

CONTOUR Plot

Figure 37.1 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 fifty percent 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 54, "The G3GRID Procedure," on page 1571).

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 54, "The G3GRID Procedure," on page 1571. For an interpolation example see Example 1 on page 1581 .

Procedure Syntax

Requirements: At least one PLOT statement is required.

Global statements: AXIS, FOOTNOTE, GOPTIONS, LEGEND, PATTERN, SYMBOL, TITLE

Reminder: The procedure can include the BY, FORMAT, LABEL, NOTE, and WHERE statements.

PROC GCONTOUR <DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<INCOMPLETE>;
PLOT *y*x=z </option(s)*>;

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.

Requirements: An input data set is required.

Syntax

PROC GCONTOUR <DATA= *input-data-set*>
<ANNOTATE=*Annotate-data-set* >
<GOUT=<*libref.*>*output-catalog*>

<INCOMPLETE>;

Options

PROC GCONTOUR statement options affect all graphs produced by the procedure.

ANNOTATE=*Annotate-data-set*
ANNO=*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.

Restriction: Partially supported by Java and ActiveX

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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 also: "SAS Data Sets" on page 54 and "Concepts" on page 1097

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.

Requirements: A plot request is required.

Global statements: AXIS, FOOTNOTE, GOPTIONS, LEGEND, NOTES, PATTERN, SYMBOL, TITLE

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. Additionally, you can filter your data with a WHERE clause, format your data values, add labels to the variables, and generate multiple plots with a BY statement or multiple plot statements.

Syntax

PLOT *y*x=z </option(s)>*;

option(s) can be one or more options in the following categories:

- appearance options:
  - ANNOTATE=*Annotate-data-set*
  - CAXIS=*axis-color*
  - CFRAME=*background-color*
  - COUTLINE=*outline-color*
  - CTEXT=*text-color*
  - GRID
  - NOAXIS | NOAXES
  - NOFRAME
- horizontal axis options:
  - AUTOHREF
  - CAUTOHREF=*reference-line-color*
  - CHREF=*reference-line-color* | *(reference-line-color)*|*reference-line-color list*
  - HAXIS=AXIS<1...99>
  - HMINOR=*number-of-minor-tick-marks*
  - HREF=*value-list*
  - HREVERSE
  - LAUTOHREF=*reference-line-type*
  - LHREF=*reference-line-type* | *(reference-line-type) reference-line-type list*
  - WAUTOHREF=*reference-line-width*
  - WHREF=*reference-line-width*|*(reference-line-width)*| *reference-line-width list*
  - XTICKNUM=*number-of-major-tick-marks*
- vertical axis options:
  - AUTOVREF
  - CAUTOVREF=*reference-line-color*
  - CVREF=*reference-line-color* | *(reference-line-color)* | *reference-line-color list*
  - LAUTOVREF=*reference-line-type*
  - LVREF=*reference-line-type* | *(reference-line-type)* | *reference-line-type list*
  - VAXIS=AXIS<1...99>
  - VMINOR=*number-of-minor-tick-marks*
  - VREF=*value-list*
  - VREVERSE
  - WAUTOVREF=*reference-line-width*

WVREF= *reference-line-width* | *(reference-line-width)* | *reference-line-width list*
YTICKNUM=*number-of-major-tick-marks*

- contour options:
  CLEVELS=*color(s)*
  JOIN
  LEGEND=LEGEND<1...99>
  LEVELS=*value-list*
  LJOIN
  LLEVELS=*line-type-list*
  NLEVELS=*number-of-levels*
  NOLEGEND
  PATTERN
  SMOOTH
- labeling options:
  AUTOLABEL | AUTOLABEL=*(autolabel-suboptions)*
  where *autolabel-suboptions* can be one or more of these:
  CHECK=*checking-factor* | NONE
  MAXHIDE=*amount<units>*
  REVEAL
  TOLANGLE=*angle*
- catalog entry description options:
  DESCRIPTION="*description*"
  NAME="*name*"

Required Arguments

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.

Options

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 also: Chapter 33, "The GANNO Procedure," on page 913 and Chapter 30, "Annotate Dictionary," on page 667

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.

Featured in: Example 2 on page 1116

Restriction: Not supported by Java and ActiveX

See also: "Autolabel Suboptions" on page 1108

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.

Style reference: Color attribute of the GraphGridLines element

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.

Style reference: Color attribute of GraphAxisLines element

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=

Style reference: Color attribute of the GraphWalls element

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 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=

Style reference: ContrastColor attribute of the GraphReference element

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.

Style reference: Color attribute of the GraphData element

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.

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. △

Restriction: Not supported by Java and ActiveX

Featured in: Example 4 on page 1120

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 it finds from the following list:

1. colors specified for labels and values on assigned AXIS and LEGEND statements.
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, 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 and is used for axis labels and values; 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. △

Featured in: Example 4 on page 1120

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 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=

Style reference: ContrastColor attribute of the GraphReference element

Restriction: Not supported by Java

DESCRIPTION=*"entry-description"*
specifies the description of the catalog entry for the chart. The maximum length for *entry-description* is 256 characters. This description does not appear on the chart. The GCONTOUR procedure assigns a description of the form PLOT OF *y***x*=*z*, where *y***x*=*z* is the request specified in the PLOT statement.

Alias: DES=

GRID
draws reference lines at all major tick marks on both axes. This is the same as 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.

Featured in: Example 2 on page 1116

Restriction: Partially supported by Java and ActiveX

See also: "AXIS Statement" on page 198

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. The HMINOR= option overrides the MINOR= option in an AXIS definition assigned to the horizontal axis.

Alias: HM=

Featured in: Example 3 on page 1118

HREF=*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. 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. 1 specifies a solid line; values 2 through 46 specify dashed lines.

Default: 1 (solid)

Style reference: LineStyle attribute of the GraphGridLines element

Restriction: Not supported by Java

See also: "Specifying Line Types" on page 276 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. 1 specifies a solid line; values 2 through 46 specify dashed lines.

Style reference: LineStyle attribute of the GraphGridLines element

Restriction: Not supported by Java

See also: "Specifying Line Types" on page 276 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.

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.

Restriction: Partially supported by Java (always displayed on the right side of plot) and ActiveX

See also: "LEGEND Statement" on page 225

Featured in: Example 2 on page 1116

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 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. 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= *(reference-line-type)* for lines drawn with the AUTOHREF option

Alias: LH=

Style reference: LineStyle attribute of the GraphReference element

Restriction: Not supported by Java and partially supported by ActiveX

See also: "Specifying Line Types" on page 276 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. 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 also: "Specifying Line Types" on page 276 for the line types represented by each number.

Featured in: Example 3 on page 1118.

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. 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= *(reference-line-type)* for lines drawn with the AUTOVREF option

Alias: LV=

Style reference: LineStyle attribute of the GraphReference element

Restriction: Partially supported by Java and ActiveX

See also: "Specifying Line Types" on page 276 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 name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase. Periods are converted to underscores. If you specify DEVICE=ACTIXIMG or DEVICE=JAVAIMG, then the name that you specify is used for the graphics output file even if the file exists. If the name duplicates an existing GRSEG name, then SAS/GRAPH adds a number to the name to create a unique entry name—for example, GCONTOU1.

Default: GCONTOUR

See also: "About Filename Indexing" on page 99

Featured in: Example 3 on page 1118

NLEVELS=*number-of-levels*
specifies the number of contour levels to plot. Valid values are integers from 1 to 100.

Restriction Partially supported by Java and ActiveX

Featured in: Example 3 on page 1118

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 and assigning the pattern for the level closest to the mean.

To explicitly define patterns, use PATTERN definitions for map or plot patterns.

Featured in: Example 4 on page 1120

See also: "PATTERN Statement" on page 240

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 also: "AXIS Statement" on page 198

Featured in: Example 3 on page 1118

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. The VMINOR= option overrides the MINOR= option in an AXIS definition that is assigned to the vertical axis.

Alias: VM=

Featured in: Example 3 on page 1118

VREF=*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.

Default: Current style setting, 1 if NOGSTYLE

Style reference: LineThickness attribute of the GraphGridLines element

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.

Default: Current style setting, 1 if NOGSTYLE

Style reference: LineThickness attribute of the GraphGridLines element

Restriction: Not supported by Java and ActiveX

WHREF=*reference-line-width*
specifies a line width for reference lines specified by the HREF= option. The *reference-line-width* can be any number greater than zero, and does not need to be an integer.

Default: Current style setting, 1 if NOGSTYLE

Style reference: LineThickness attribute of the GraphReference element

Restriction: Not supported by Java and ActiveX

WVREF=*reference-line-width*
specifies a line width for reference lines specified by the VREF= option. The *reference-line-width* can be any number greater than zero, and does not need to be an integer.

Default: Current style setting, 1 if NOGSTYLE

Style reference: LineThickness attribute of the GraphReference element

Restrictions: 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, if you specify MAXHIDE=50 PCT, and if the graphics output area is 9 inches wide, 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. Additionally, 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.

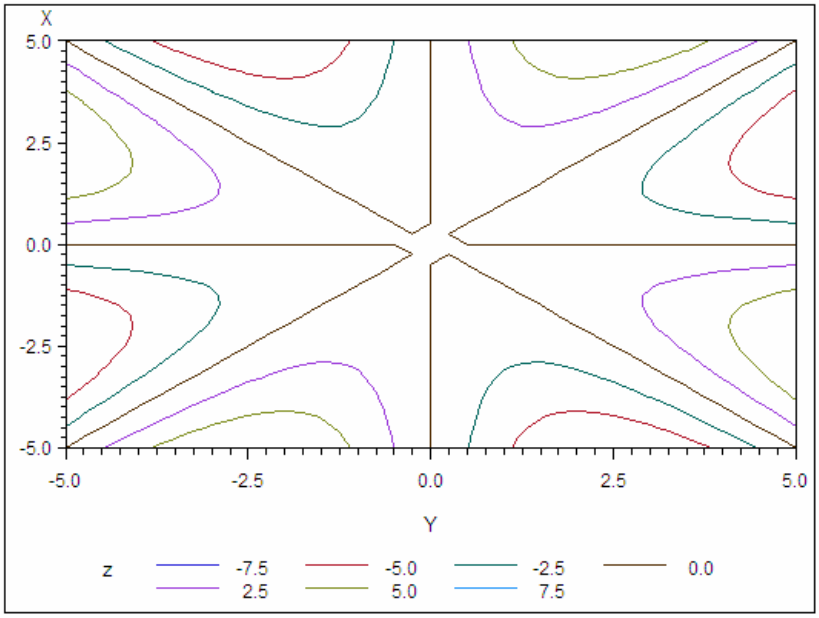
Default: 30

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 37.2 Selecting Contour Levels

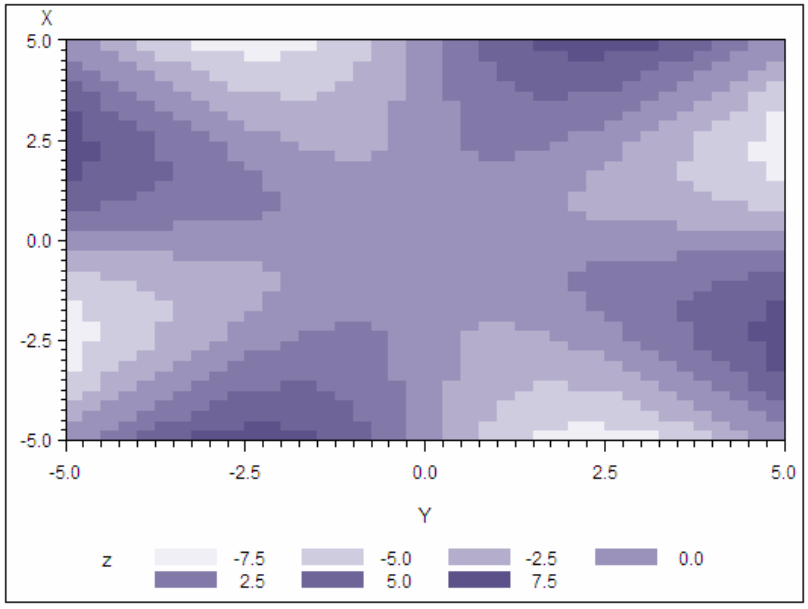


Using the PATTERN option with the LEVELS= option generates a plot with contour levels that are displayed as solid filled rectangles. The rectangles are formed by points in the *(x, y)*grid. The contour pattern of a rectangle, or grid cell, is determined by average value of the *z* variable for the four corners of the rectangle. The grid cell is assigned the pattern for the level closest to the calculated mean. For example, if you have specified contour levels of 0, 5, and 10, and the plot contains a grid cell with a mean of 100, it 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 on page 1120. Contour plots using the same contour levels can present your data differently, if one plot uses a pattern and the other does not. The contour pattern boundaries do not correspond to the contour lines shown in Example 4 on page 1120.

Figure 37.3 Contour Plot with the PATTERN Option

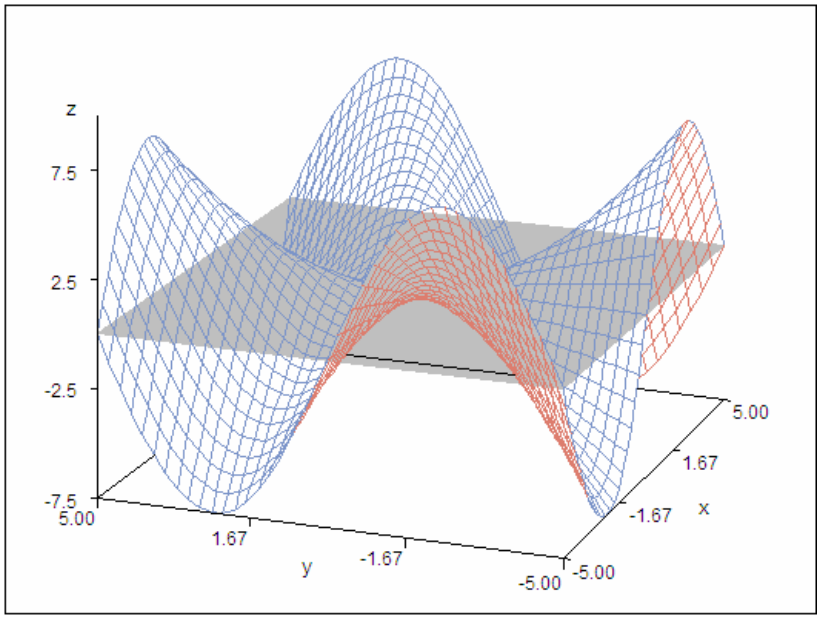


Using the data to create a surface plot with the G3D procedure, the contour lines in a GCONTOUR procedure plot represent the intersection of the plane and the surface.

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* and located at *z* values of –7.5, –5.0, –2.5, and so on.

Figure 37.4 G3D Surface Plot



Specifying Axis Order

You can use AXIS statements to modify the text and appearance of plot axes, and then you can 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, which 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 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 37.5 The ORDER= option, values match the range of the data values

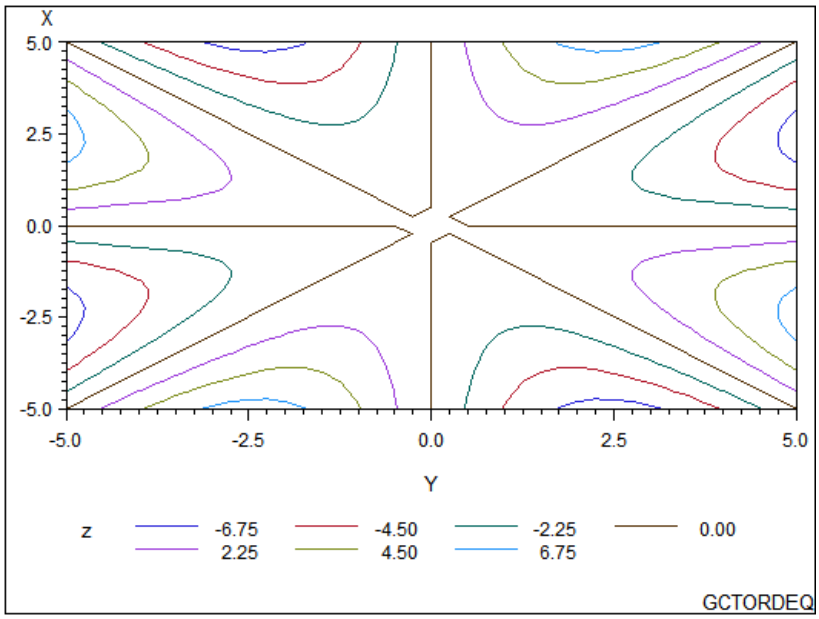


Figure 37.6 The ORDER= option, values clip the range of data values

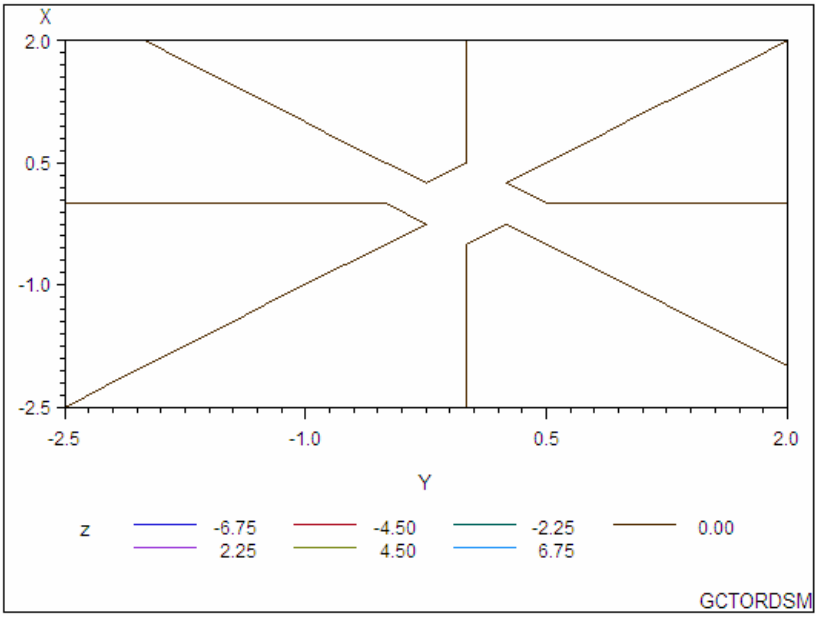
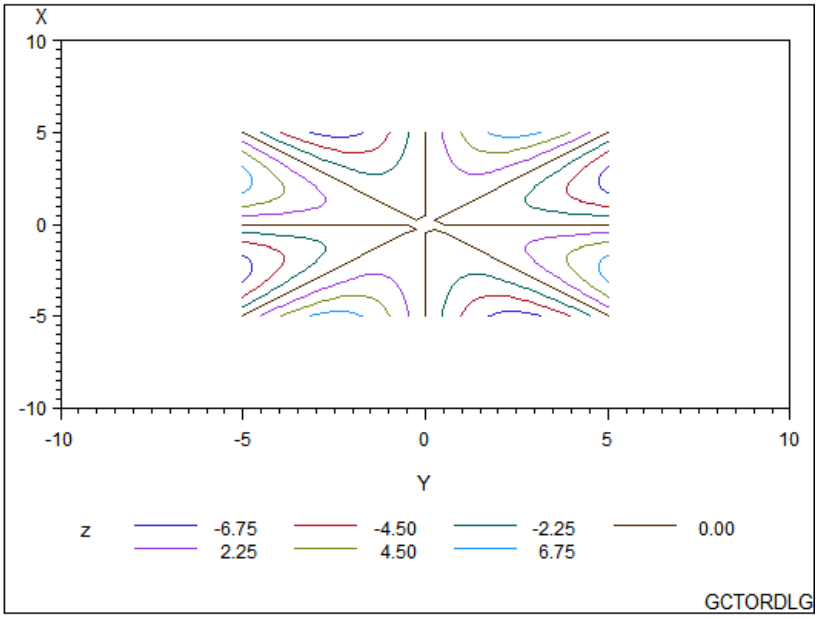


Figure 37.7 The ORDER= option, values extend beyond the range of the data values



Modifying Contour Lines and Labels with the SYMBOL Statement

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 “SYMBOL Statement” on page 252 for details.

Table 37.1 on page 1114 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
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 percent is ignored by the GCONTOUR procedure and a value of 10 percent is substituted.

For more information, see "SYMBOL Statement" on page 252.

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 on 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:

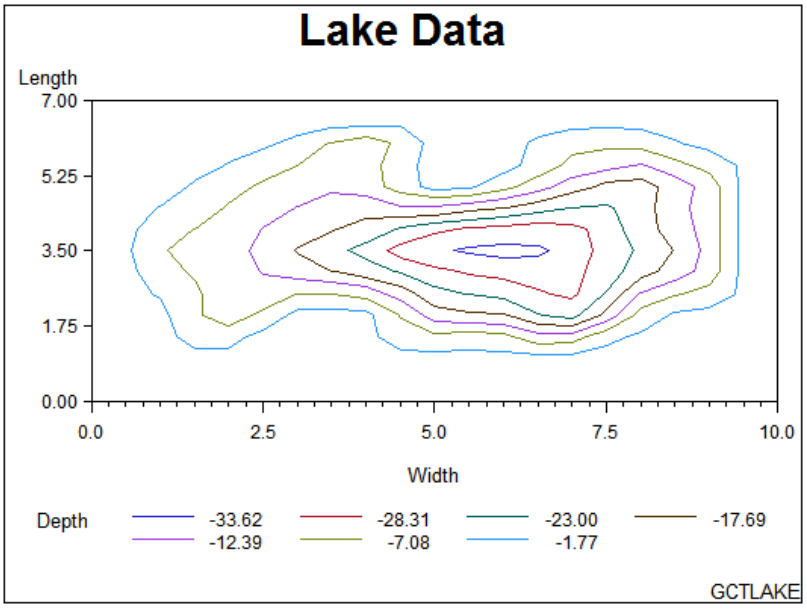
```
symbol1 font="Arial Rounded MT Bold" value="DEEP";
```

You must specify both FONT= and VALUE= options or the text is not used. For an example, see Example 2 on page 1116.

Examples

Example 1: Simple Contour

Procedure features: PLOT Statement
Data Set SASHELP.LAKE
Sample library member: GCTLAKE



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.

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.

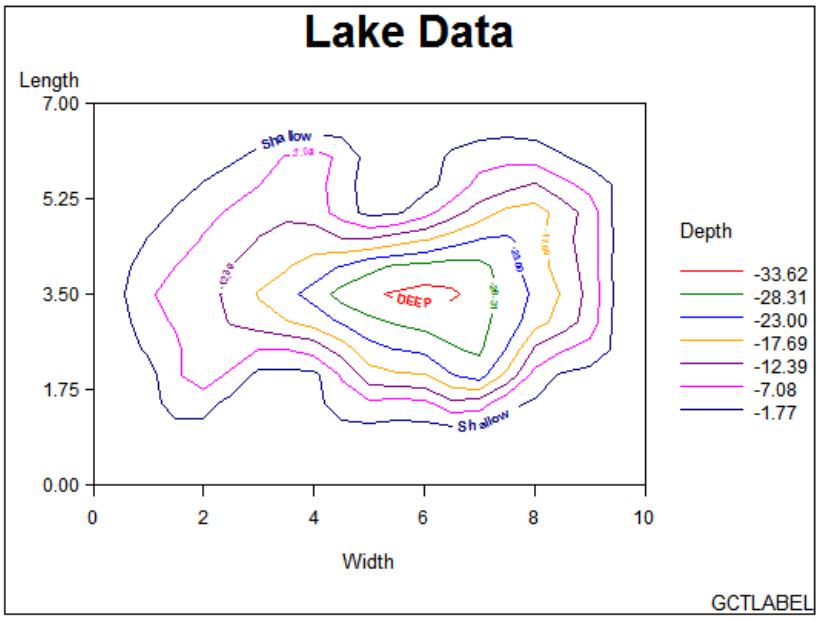
```
title "Lake Data";
footnote j=r "GCTLAKE";
```

Generate contour plot. Generate a simple contour plot using SASHELP.LAKE. Use one PLOT statement to define the grid 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

Procedure features: AUTOLABEL, HAXIS, LEGEND
Data Set SASHELP.LAKE
Sample library member: GCTLABEL



This example modifies Example 1 on page 1115 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.

Additionally, the LEGEND statement attributes are modified to reposition the legend closer to the data.

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 to 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 will be 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.

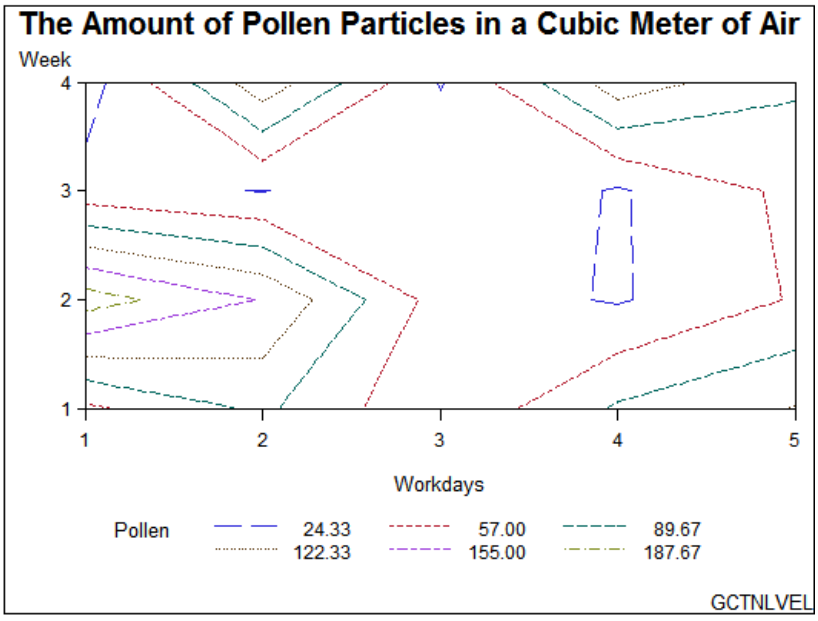
```
proc gcontour data=sashelp.lake;
   plot length*width=depth/
       autolabel=(check=none)
      haxis=axis1
      legend=legend1
      name="GCTLABEL";
   run;
quit;
```

Example 3: Specifying Contour Levels

Procedure features: HMINOR, LLEVELS, NAME, NLEVELS, VAXIS, VMINOR

Data Set POLLEN

Sample library member: GCTNLVEL



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.

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 titles and footnotes. 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.

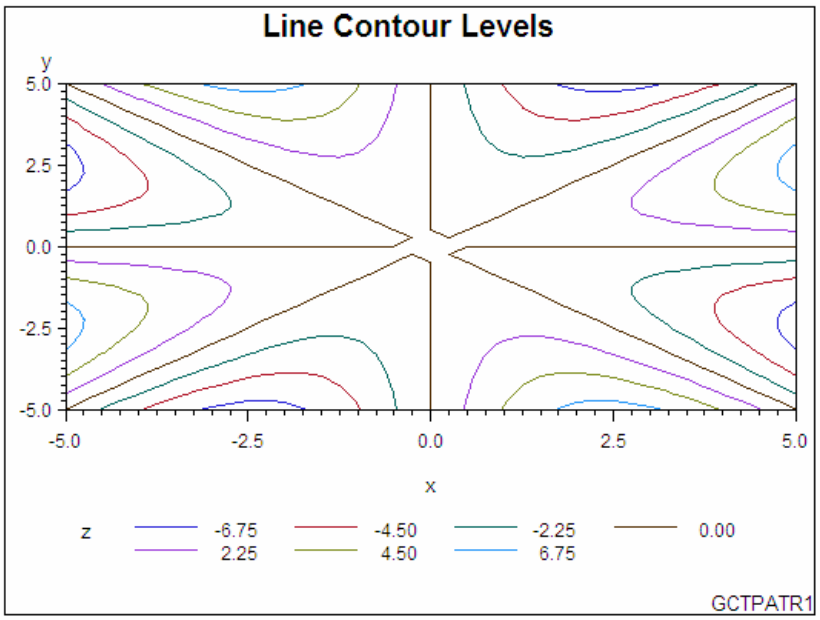
```
proc gcontour data=pollen;
    plot week*workdays=pollen/
       hminor=0
       llevels= 2 20 21 33 25 41
       name="GCTNLVEL"
       nlevels=6
       vaxis=axis1
       vminor=0;
    run;
 quit;
```

Example 4: Using Patterns and Joins

Procedure features: COUTLINE, CTEXT, HAXIS, JOIN, LEGEND, PATTERN
Data Set SWIRL
Sample library member: GCTPATJN

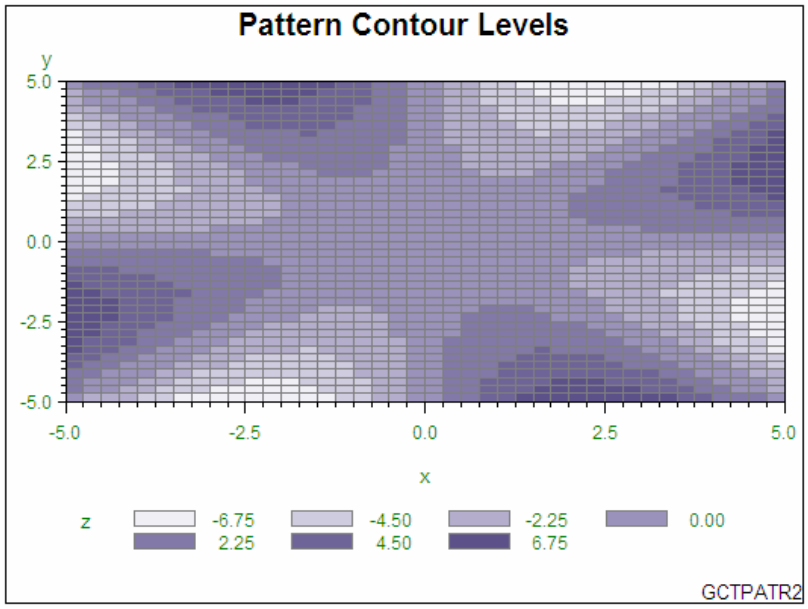
This example demonstrates the differences between using lines and patterns to represent contour levels. The first PLOT statement generates a plot with lines representing contour levels.

Figure 37.8 Line Contour Levels



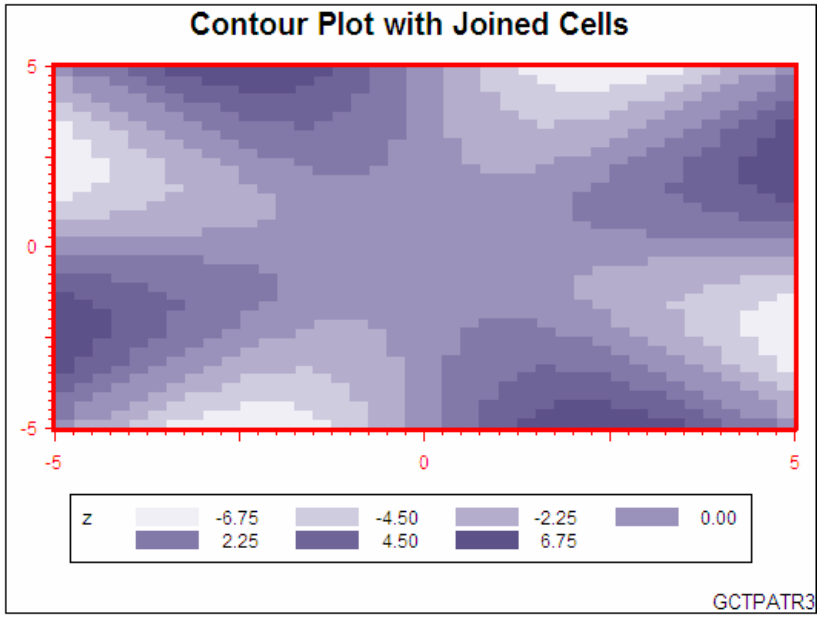
The second PLOT statement specifies the PATTERN option to fill and color contour levels. Additional PLOT statement options outline filled areas in gray and specify green text for all text on the axes and in the legend.

Figure 37.9 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.

Figure 37.10 Contour Plot with Joined Cells



Set the graphics environment. Draw a border around the graphics output area.

```
goptions reset=all border;
```

Crate 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

The GDEVICE Procedure

Overview

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:

- 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 Chapter 6, "Using Graphics Devices," on page 67 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 SAS supplied device entries that are supported by your operating environment, see the SASHELP.DEVICES catalog.

Concepts

Device 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.
Specify the current catalog by;

- using the CATALOG= option in the PROC GDEVICE statement (this is required to open a driver entry in update mode)
- assigning 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:

1. any of the GDEVICE libraries do not contain a DEVICES catalog
2. 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 reasons, or it 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. △

Ways to Use the GDEVICE Procedure

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 1127
- submit GDEVICE procedure statements in a SAS program (program mode). See “Running the GDEVICE Procedure In Program Mode” on page 1128

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 allow 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 1126.)

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 1136.

In a windowing environment, you can switch between the GDEVICE windows and program statements while you are running the procedure. See “FS Statement” on page 1134 and the NOFS option in “PROC GDEVICE Statement” on page 1129.

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 1126.) 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.

Procedure Syntax

Requirements: 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*
required-parameters
<*optional-parameters*>;

COPY *device-entry*
<FROM=<*libref.*>*SAS-catalog*>
<NEWNAME=*new-device-entry*>;

DELETE *device-entry*;

FS;

LIST *device-entry* | _ALL_ | _NEXT_ | _PREV_ | DUMP>;
MODIFY *device-entry*
parameter(s)
QUIT | *END* | *STOP*;
RENAME *device-entry* NEWNAME=*entry-name*;

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.

PROC GDEVICE <CATALOG=<*libref.*>*SAS-catalog*>
<BROWSE>
<NOFS>;

Options

Options used in the PROC GDEVICE statement affect the way 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*
CAT=<*libref.*>*SAS-catalog*
C=<*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 1127.

To edit the device entries in a catalog, you must specify the CATALOG= option.

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.

Requirements: You must have write access to the device catalog in order to add entries, and use CATALOG= in the PROC GDEVICE statement.

Restriction: Not valid in browse mode.

ADD *device-entry*
required-parameters
<*optional-parameters*>;

required-parameters are all of these parameters:

MODULE=*driver-module*
XMAX=*width* <IN | CM>
YMAX=*height* <IN | CM>
XPIXELS=*width-in-pixels*
YPIXELS=*height-in-pixels*

plus one or both of these parameter pairs:

LCOLS=*landscape-columns*
LROWS=*landscape-rows*

or

PCOLS=*portrait-columns*
PROWS=*portrait-rows*

optional-parameters can be one or more of these options:

ASPECT=*scaling-factor*
AUTOCOPY=Y | N
AUTOFEED=Y | N
CBACK=*background-color*
CELL=Y | N
CHARACTERS=Y | N
CHARREC=(*charrec-list(s)*)
CHARTYPE=*hardware-font-chartype*
CIRCLEARC=Y | N
CMAP=('*from-color* : *to-color*' <...,'*from-color-n* : *to-color-n*'>)
COLORS=(<*colors-list*>)
COLORTYPE=NAME | RGB | HLS | GRAY | CMY | CMYK | HSV | HSB
DASH=Y | N
DASHLINE='*dashed-line-hex-string*'X
DESCRIPTION='*text-string*'
DEVMAP=*device-map-name* | NONE
DEVOPTS='*hardware-capabilities-hex-string*'X
DEVTYPE=*device-type*
DRVINIT1='*system-command(s)*'
DRVINIT2='*system-command(s)*'
DRVQRY | NODRVQRY
DRVTERM1='*system-command(s)*'
DRVTERM2='*system-command(s)*'
ERASE=Y | N
FILECLOSE=DRIVERTERM | GRAPHEND
FILL=Y | N
FILLINC=0...9999

FORMAT=CHARACTER | BINARY
GACCESS=*output-format* | '*output-format* > *destination*'
GCOPIES=*current-copies*
GEND='*string*' <...'*string-n*'>
GEPILOG='*string*' <...'*string-n*'>
GPROLOG='*string*' <...'*string-n*'>
GPROTOCOL=*module-name*
GSFLEN=*record-length*
GSFMODE=APPEND | REPLACE | PORT
GSFNAME=*fileref*
GSIZE=*lines*
GSTART='*string*' <...'*string-n*'>
HANDSHAKE=HARDWARE | NONE | SOFTWARE | XONXOFF
HEADER='*command*'
HEADERFILE=*fileref*
HORIGIN=*horizontal-offset* <IN | CM>
HOSTSPEC='*text string*'
HSIZE=*horizontal-size* <IN | CM>
ID='*description*'
INTERACTIVE=USER | GRAPH | PROC
LFACTOR=*line-thickness-factor*
MAXCOLORS=*number-of-colors*
MAXPOLY=*number-of-vertices*
MODEL=*model-number*
NAK='*negative-handshake-response*'X
PAPERFEED=*feed-increment* <IN | CM>
PATH=*angle-increment*
PENSORT=Y | N
PIEFILL=Y | N
POLYGONFILL=Y | N
POSTGRAPH1='*system-command(s)*'
POSTGRAPH2='*system-command(s)*'
PREGRAPH1='*system-command(s)*'
PREGRAPH2='*system-command(s)*'
PROCESS='*command*'
PROCESSINPUT=*fileref*
PROCESSOUTPUT=*fileref*
PROMPT=0...7
PROMPTCHARS='*prompt-chars-hex-string*'X
QMSG | NOQMSG
RECTFILL='*rectangle-fill-hex-string*'X
REPAINT=*redraw-factor*
ROTATE=LANDSCAPE | PORTRAIT
ROTATION=*angle-increment*
pen-speed

SWAP=Y | N
SYMBOL=Y | N
SYMBOLS='*hardware-symbols-hex-string*'X
TRAILER='*command*'
TRAILERFILE=*fileref*
TRANTAB=*table* | *user-defined-table*
TYPE= CAMERA | CRT | EXPORT | PLOTTER | PRINTER
UCC='*control-characters-hex-string*'X
VORIGIN=*vertical-offset* <IN | CM>
VSIZE=*vertical-size* <IN | CM>

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.

required-parameters
specify the required parameters for the device. All of the required parameters for the ADD statement correspond to device parameters of the same name. Refer to Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for a description of each parameter.

Options

All optional parameters for the ADD statement correspond to device parameters of the same name. Refer to Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for a description of each parameter.

Note: The COLORS= device parameter is not required; the device entry is created even if you do not specify it. However, the GDEVICE procedure issues an error message if you do not specfy at least one color for the COLORS= option. △

Best Practice

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.

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.

Requirements: You must have write access to the catalog where the device entry is being copied.

Restriction: Not valid in browse mode.

See also: "Creating or Modifying Device Entries" on page 1142

Featured in: Example 1 on page 1143

COPY *device-entry where*;

Where *where* must be one or both of these options:

FROM=<*libref.*>*SAS catalog*

NEWNAME=*new-device-entry*

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.

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.

Requirements: 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.

Restriction: Not valid in browse mode.

Caution: 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.

DELETE *device-entry*;

Required Arguments

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.

Requirements: You must be running SAS in a windowing environment.

FS;

Options

No options.

LIST Statement

Lists all of the parameters of the specified device entry in the Output window.

Default: _ALL_

See also: "Running the GDEVICE Procedure In Program Mode" on page 1128

LIST <*device-entry*>
<_ALL_>
<_NEXT_>
<_PREV_>
<DUMP>;

Options

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 on 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.

Requirements: 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.

Restriction: Not valid in browse mode.

See also: "Creating or Modifying Device Entries" on page 1142

Featured in: Example 1 on page 1143

MODIFY *device-entry*
parameter(s);

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.

parameter(s)
are the parameters you want to modify. These can be any of the parameters listed in the ADD statement, whether listed as required or optional for ADD. See "ADD Statement" on page 1129 for a complete list. Refer to Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for a description of each parameter.

Details

To modify a device entry, create your own catalog and then copy the device entries 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 the COPY statement, the COPY command available after you choose 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 you have just made. To change a value 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 type the CANCEL command in the command line to cancel changes you have made to the fields.) △

QUIT Statement

Saves all modifications made to device entries during the procedure. Exits the GDEVICE procedure.

QUIT | *END* | *STOP*;

Options

No options.

RENAME Statement

Changes the name of the device entry to the name specified in the statement.

Requirements: 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.

Restriction: Not valid in browse mode.

RENAME *device-entry*
NEWNAME=*entry-name*;

Required Arguments

device-entry
specifies the one-level name of the device entry that you want to rename.

Restriction: The entry must exist in the current catalog.

NEWNAME=*entry-name*
specifies the new entry name. *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.

Using the GDEVICE Procedure

Using the GDEVICE Windows

You can use GDEVICE windows instead of program mode to view, modify, copy, create, or delete device entries. 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 Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327. For a complete list of device parameters, see "ADD Statement" on page 1129.

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 you specified in the PROC GDEVICE statement. You can use it to

- 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.

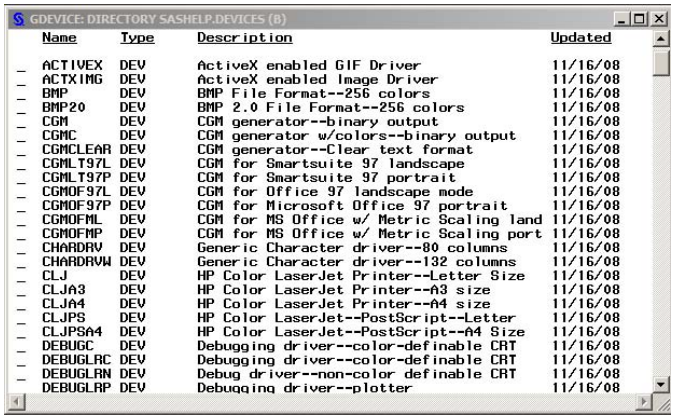
D
: delete the selected device entry. You cannot restore a device entry once it has been deleted.

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

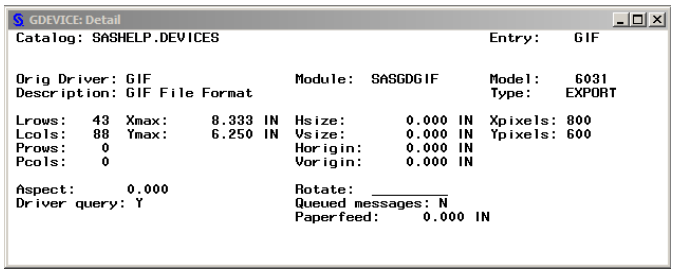


After the Directory window opens, you can navigate through the GDEVICE windows by selecting the **View ▶ Next Screen**.

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



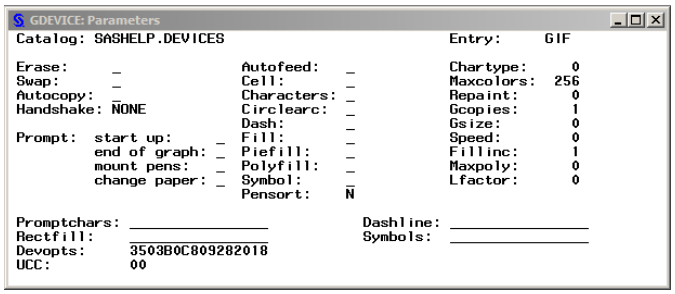
Parameters window

This window includes additional device parameters that affect the way 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. △

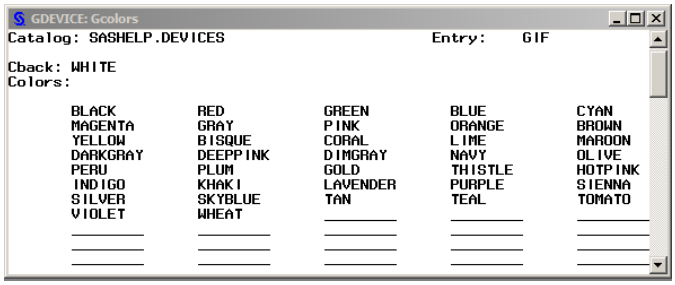
Figure 38.3 The Parameters Window



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.

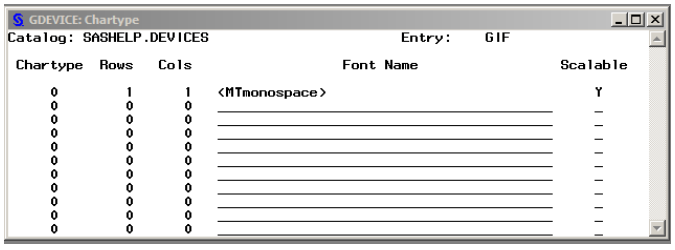
Figure 38.4 The Gcolors Window (partial view)



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)

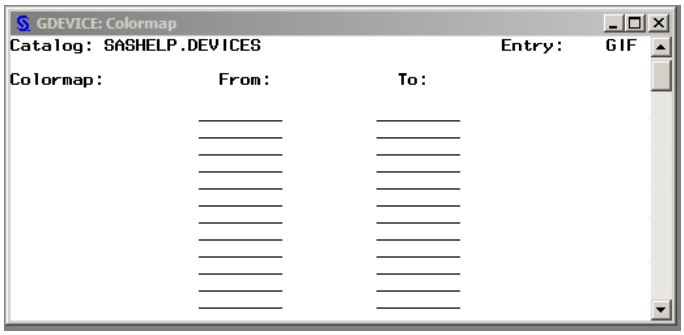


Colormap window

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)



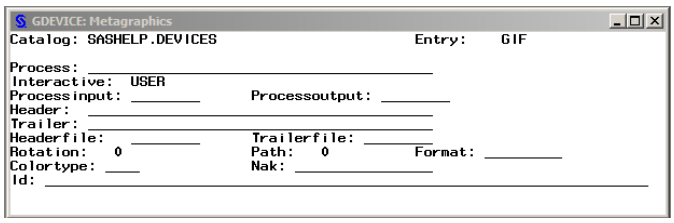
Metagraphics window

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 Chapter 15, “Graphics Options and Device Parameters Dictionary,” on page 327 for details.

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 Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for details.

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.

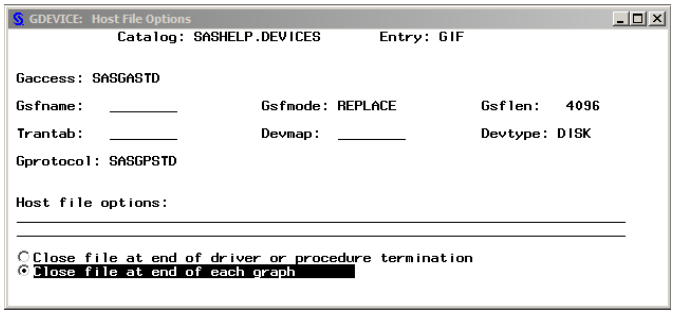
Gend window

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 91.)

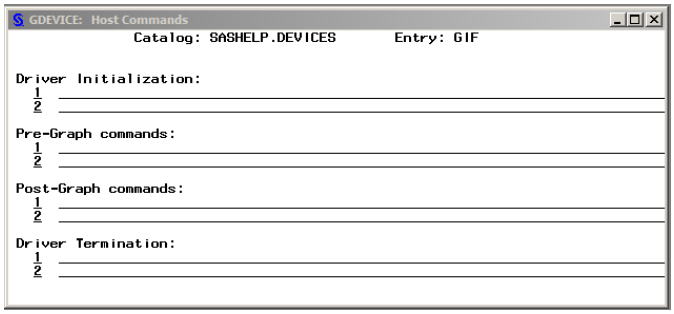
Figure 38.8 The Host File Options Window



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

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 1126.

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 it finds whose name matches the one you have specified. (See "Search Order of Device Catalogs" on page 1127.)

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 suit 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 on 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 1 on page 1143).

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 1132. 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 you want to modify into your

personal catalog. Change the parameters in the new device entry. See Example 1 on page 1143 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 140 and "Precedence of Appearance Option Specifications" on page 141. See also "Style Attributes Versus Device Entry Parameters" on page 134.

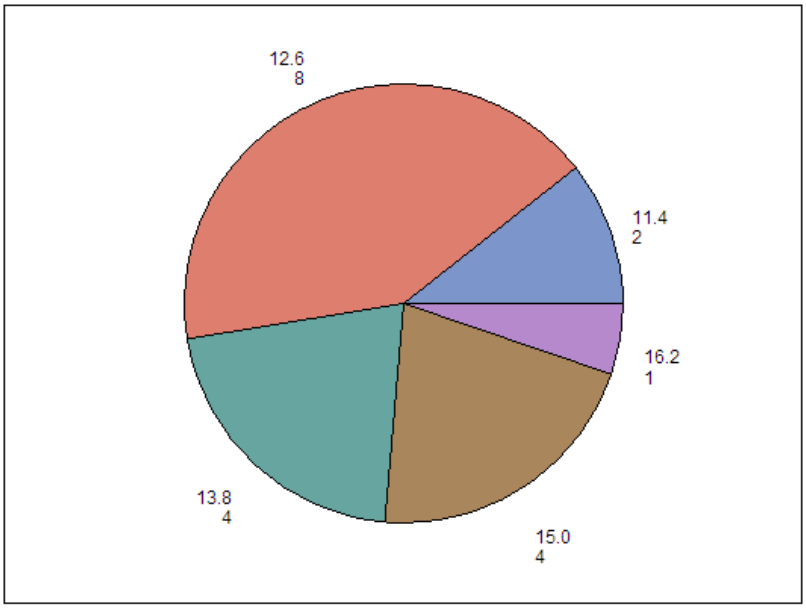
Examples

Example 1: Creating a Custom Device Entry with Program Statements

COPY statement

MODIFY statement

Figure 38.10 Pie Chart Created with Default WIN Device Entry



This example shows how to use GDEVICE procedure statements to modify a device entry by copying the original entry into a personal catalog and changing the device parameters.

This example permanently changes the default color list for the WIN device entry.

Assign the libref GDEVICE0. The LIBNAME statement assigns the libref to the aggregate file storage location that contains (or will contain) the DEVICES catalog.

```
libname gdevice0 "SAS-data-library";
```

Start the GDEVICE procedure. NOFS causes GDEVICE to use program mode. CATALOG= assigns GDEVICE0.DEVICES as the current 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 WIN that is placed in GDEVICE0.DEVICES. The name of a catalog entry cannot exceed eight characters.

```
copy win from=sashelp.devices newname=mypscol;
```

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 mypscol
  description="WIN with new color list"
  colors=(black cx95c051 cxA359B2 cxD65259 cx69D6D2 cxFFB74F cx929cff);
```

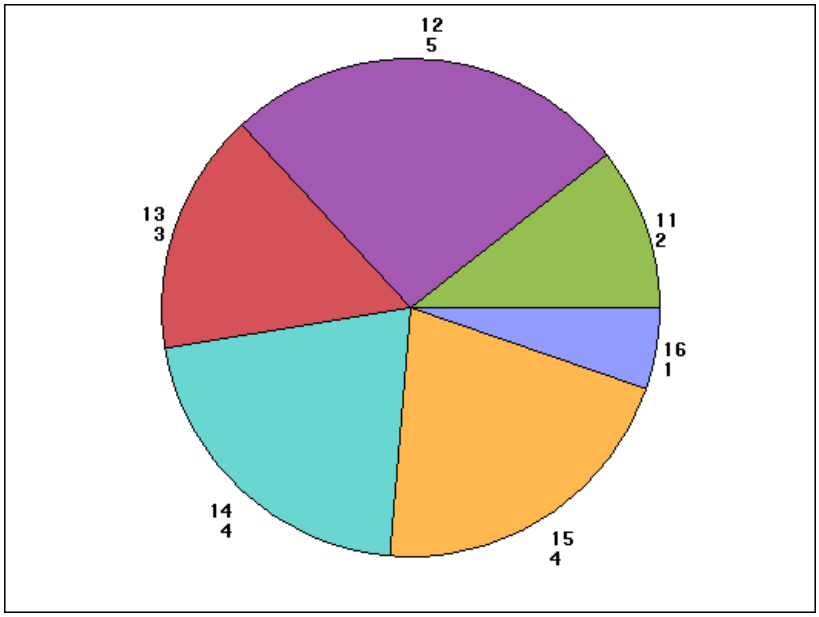
Exit the procedure.

```
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=mypscol;
proc gchart data=sashelp.class;
  pie age/discrete noheading;
 run;
quit;
```

Figure 38.11 Pie Chart Created with Customized WIN Device Entry



CHAPTER

39

The GEOCODE Procedure

Overview of the GEOCODE Procedure

Geocoding is the process of adding geographic coordinates (latitude and longitude values) to an address. The coordinates typically represent the center of a ZIP code, a city, an address, or any geographic region. After geocoding, the coordinates can be used to display a point on a map or to calculate distances. Geocoding also enables you to add attribute values such as census blocks to an address.

The GEOCODE procedure requires two types of SAS data sets:

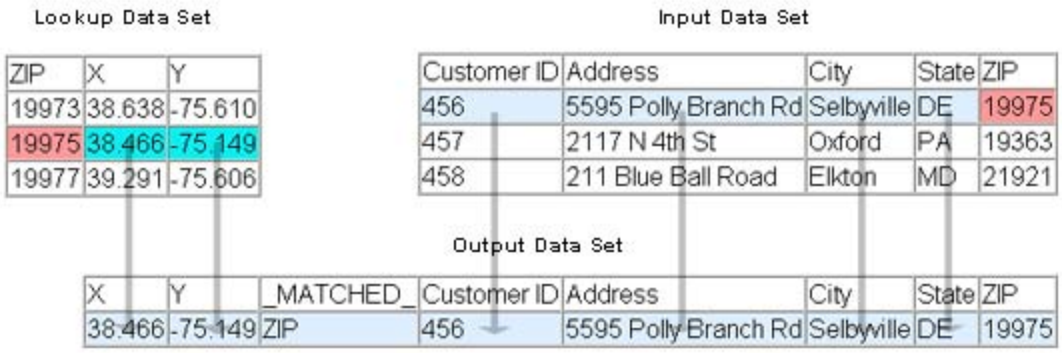
input address data sets
: contain variables that relate to specific geographic locations. For example, mailing address variables such as ZIP codes and street addresses, or custom geographic variables such as sales regions.

lookup data sets
: contain reference variables and geographic coordinates. For example, a lookup data set for the ZIP method contains ZIP codes and the geographic coordinates that are associated with the ZIP codes. Some geocoding methods require multiple lookup data sets.

When the GEOCODE procedure finds a match in the lookup data set, it adds the associated coordinates to the observation in the output data set. Longitude is stored as the X variable, and latitude is stored as the Y variable.

The following image shows how the ZIP geocoding method of the GEOCODE procedure obtains coordinates for the output data set by matching the ZIP code in the input data set:

Figure 39.1 Geocoding with ZIP Codes



The GEOCODE procedure also adds a variable named _MATCHED_ that indicates how the coordinates were found. Possible values for the _MATCHED_ variable are as follows:

Street	A match was found for either the street address and ZIP code or the street address, city, and state.
ZIP	A match was found for the ZIP code.
ZIP+4	A match was found for the ZIP code and ZIP+4 extension.
ZIP mean	Multiple observations in the lookup data set matched the ZIP code, and the coordinate values were averaged.
City	A match was found for the city and state.
City mean	Multiple observations in the lookup data set matched the city and state, and the coordinate values were averaged.
variable-name	For CUSTOM and RANGE geocoding, a variable name indicates that a match was found for that variable.
None	No match was found for the address.

For each observation in the input data set, the GEOCODE procedure attempts to match the address variable value to a value in the lookup data set. For most geocoding methods, the lookup data set is expected to contain only one matching observation. For example, the SASHELP.ZIPCODE data set contains only one observation for each ZIP code. If the lookup data set contains multiple matches, then the first matching observation is returned, except as noted in the following paragraph.

Some geocoding methods do process multiple matches. For example, if you are using ZIP code geocoding and no match is found, then the GEOCODE procedure attempts to find a matching city-and-state pair. The SASHELP.ZIPCODE data set contains multiple observations for many city-and-state pairs. When a ZIP code is not found in this lookup data set, a matching city-and-state pair is searched for. If one match is found, then the coordinates for the matching pair are used. However, if multiple matches are found, then the coordinate values for those matches are averaged. If you are using the STREET or PLUS4 geocoding method and no match is found for the combined ZIP code and ZIP+4 values, then the GEOCODE procedure searches for the five-digit ZIP code only.

Concepts

Output Data Sets

By default, the GEOCODE procedure produces an output data set that contains all of the variables from the input address data set and the X, Y, and _MATCHED_ variables. You can also choose to add variables from the lookup data set to the output data set by using the ATTRIBUTEVAR= option. For example, if you are using SASHELP.ZIPCODE as the lookup data set, then you could assign the county name (COUNTYNM) to each matched observation in the output data set.

The default name for the output data set is DATA*n*, where *n* is the smallest integer that makes the name unique. For example, if the DATA1 data set already exists, then the default name for the output data set is DATA2.

The label of the output data set contains the text, "geocoded *date*" where *date* is the date when the output was created. This text is appended to the label from the input data set, if one exists.

For the STREET geocoding method, additional variables are included in the output data set. See “Output Variables for Street Geocoding” on page 1164.

The SASHELP.ZIPCODE Data Set

The default lookup data set for ZIP code geocoding and CITY geocoding is SASHELP.ZIPCODE. This data set is provided with Base SAS, and is updated for each SAS release.

You can download updated versions of the SASHELP.ZIPCODE data set from the SAS Maps Online Web site: **`www.sas.com/mapsonline`**.

SASHELP.ZIPCODE contains the following variables:

Name:	Label:
ZIP	The 5-digit ZIP code
Y	Latitude (decimal degrees) of the center of the ZIP code. 0.0 for APO/FPO

X	Longitude (decimal degrees) of the center of the ZIP code. 0.0 for APO/FPO
ZIP_CLASS	ZIP code classification: M=APO/FPO; P=Post office box; U=Unique ZIP code used for large organization, businesses, or buildings; Blank=Standard/non-unique
CITY	Name of the city or organization
STATE	Two-digit number (FIPS code) for the state or territory
STATECODE	Two-character postal code for the state or territory name
STATENAME	Full name of the state or territory
COUNTY	FIPS county code. Blank for APO/FPO addresses.
COUNTYNM	Name of county or parish. Blank for APO/FPO addresses.
MSA	Metropolitan Service Area code by common population; no MSA for rural areas
AREACODE	Area code for the ZIP code. Blank for APO/FPO addresses.
AREACODES	Multiple area codes for the ZIP code. Blank for APO/FPO addresses.
ALIAS_CITY	Alternate names for the city. Each name is separated by "||".
TIMEZONE	Time zone for the ZIP code. Blank for APO/FPO addresses.
GMTOFFSET	Difference (hours) between GMT and time zone for the ZIP code.
DST	ZIP code observes Daylight Savings Time: Y is Yes N is No
PONAME	USPS Post Office name

Alternate ZIP Code and ZIP+4 Lookup Data Sets

While the SASHELP.ZIPCODE data set is the default lookup data set for the ZIP and CITY geocoding methods, data from other sources can be used as long as it is read into a SAS data set.

For ZIP code geocoding, any lookup data set must contain the following variables:

Default Name:	Description:
ZIP	Five-digit ZIP code
X	Longitude of the center coordinate
Y	Latitude of the center coordinate

For CITY geocoding, these additional variables are required:

CITY	Name of the city
STATECODE	Two-character postal code for the state or province name

Note: If you use an alternative ZIP code lookup data set, then the variable data types should match those of the SASHELP.ZIPCODE data set. △

When you use ZIP+4 geocoding, you must specify an alternative lookup data set because the SASHELP.ZIPCODE data set does not contain any ZIP+4 values. This data set must contain the following variables:

Default Name:	Description:
ZIP	Five-digit ZIP code

PLUS4	Four-digit ZIP+4 extension
X	Longitude of the central coordinate
Y	Latitude of the central coordinate

You can specify different names for the variables by using options in the PROC GEOCODE statement. For example, the LOOKUPPLUS4 option specifies the name of the ZIP+4 extension variable in the lookup data set.

The ZIP and PLUS4 variables can contain either character data or numeric data. The data type must match the type of the corresponding variable in your input data set.

Note: The character values in your input and lookup data sets do not need to be a case-sensitive match. Character value matching in the GEOCODE procedure is not case sensitive. △

Additional attribute variables can also be in the alternate lookup data set even if they are not used to find matches. You can add these variables to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

You can obtain a lookup data set for ZIP+4 geocoding from the SAS Maps Online Web site at **www.sas.com/mapsonline**. On the Downloads page, select Geocoding to access the downloads that are related to geocoding.

An alternative source for ZIP+4 lookup data is the Geo*Data product from Melissa Data. You can use the %GCDMEL9 autocall macro to convert Geo*Data files to SAS data sets. For more information, see "%GCDMEL9 Autocall Macro" on page 1152.

U.S. Military ZIP Codes

ZIP codes for U.S. military post offices are provided in the ZIPMIL data set in the SASHELP library. You can combine this data set with the ZIPCODE data set to support military ZIP codes.

Data Sets for Range Geocoding

Note: Range geocoding is for SAS 9.2 Phase 2 and later. △

For Range geocoding, a lookup data set and a range data set are required. The range data set identifies ranges of IP addresses. The lookup data set contains geographic coordinates. Both the range data set and the lookup data set must contain a key variable that identifies locations for each IP range.

The lookup data set must contain the following variables:

- a key variable that corresponds to a key variable in the range data set.
- an X variable that contains the longitude value of the center coordinate. The default variable name is X.
- a Y variable that contains the latitude value of the center coordinate. The default variable name is Y.

The range data set must contain the following variables:

- a variable that specifies the beginning value of a range of IP addresses
- a variable that specifies the ending value of a range of IP addresses
- a key variable that corresponds to a key variable in the lookup data set

You can obtain lookup and range data from third-party vendors. One vendor is MaxMind, Inc. at **www.maxmind.com**. You can use the %MAXMIND autocall macro to convert comma-separated value (CSV) files from MaxMind into SAS data sets. For more information, see "%MAXMIND Autocall Macro" on page 1152.

%GCDMEL9 Autocall Macro

Overview of the %GCDMEL9 Autocall Macro

The %GCDMEL9 autocall macro enables you to directly import Geo*Data files from Melissa Data as SAS data sets. Geo*Data files contain third-party ZIP+4 lookup data for use with PLUS4 geocoding.

Geo*Data files are available for each state. The files are provided as text files within compressed (ZIP) archives. Melissa Data also provides the PKUNZIP utility to extract the text files.

The %GCDMEL9 macro uses the following macro variables:

DATASETNAME
specifies the name of the output data set.

DATASETPATH
specifies the location where the output data set is created.

DATASETLABEL
(optional) specifies a label for the output data set.

LIBNAME
specifies the name for a new library that is assigned for the location that you specified in the DATASETPATH macro variable.

UNZIPPEDPATH
specifies the location of the extracted Geo*Data files that you want to import. The %GCDMEL9 macro attempts to read all of the text (.txt) files in this directory.

WORKPATH (Optional)
specifies the path where temporary files are written. The default path is the path for the WORK library.

Usage Example for the %GCDMEL9 Autocall Macro

In this example, a Geo*Data file for the state of Delaware (DE.txt) is extracted to **C:\Mydata**. The lookup data set is created in the directory **C:\Geocode** and assigned the libref ZIP4. The resulting data set is named ZIP4.DELAWARE.

The following code imports the data:

```
/* Define macro variables */
   %let UNZIPPEDPATH=C:\Mydata;
   %let DATASETPATH=C:\Geocode;
   %let DATASETNAME=Delaware;
   %let LIBNAME=ZIP4;
   %let DATASETLABEL=ZIP+4 lookup data for Delaware;
   /* Submit autocall macro */
   %GCDMEL9;
```

%MAXMIND Autocall Macro

Overview of the %MAXMIND Autocall Macro

The %MAXMIND autocall macro enables you to convert IP geocoding data from MaxMind, Inc. into SAS data sets. The %MAXMIND autocall macro supports MaxMind's IP data in comma-separated value (CSV) format.

Note: This feature is for SAS 9.2 Phase 2 and later. △

The %MAXMIND macro uses the following macro variables:

CSVPATH
specifies the path where the MaxMind CSV files are located. You must extract the files from the ZIP archive before using the %MAXMIND autocall macro.

IPDATAPATH
specifies the path where the output SAS data sets are created. You must have write permissions for this path.

CSVBLOCKSFILE
specifies the filename for the CSV file that contains IP address range values. The file that you specify must contain the startIpNum and endIpNum variables.

CSVLOCATIONFILE
specifies the filename for the CSV file that contains longitude and latitude values.

CSVCOUNTRYFILE (Optional)
specifies the name of the optional MaxMind CSV file that contains country names.

WORKPATH (Optional)
specifies the path where temporary files are written. The default path is the path for the WORK library.

The %MAXMIND macro creates the CITYBLOCKS and CITYLOCATION data sets in the path that you specified for the IPDATAPATH variable. The libref IPDATA is created automatically for this path.

Usage Example for the %MAXMIND Autocall Macro

In this example, data from MaxMind is located in **`C:\Mydata`**. The output SAS data sets are created in the directory **`C:\Geocode`**.
The following code imports the data:

```
%let CSVPATH=C:\Mydata;
%let IPDATAPATH=C:\Geocode;
%let CSVBLOCKSFILE=GeoLiteCity-Blocks.csv;
%let CSVLOCATIONFILE=GeoLiteCity-Location.csv;
%let CSVCOUNTRYFILE=GeoIPCountryWhois.csv;
%maxmind;
```

The imported data sets are IPDATA.CITYBLOCKS and IPDATA.CITYLOCATION.

Optimizing Performance

Overview of Enhancing Performance

Geocoding often requires very large lookup data sets, which can affect the performance of the GEOCODE procedure. You can optimize your geocoding performance by performing the following actions:

- □ Index your lookup data sets by using the appropriate variables.
- □ Load the lookup data sets into memory by using the SASFILE statement.

Indexing your Lookup Data Sets

If you use alternative lookup data sets, then indexing your lookup data sets can improve performance. You should create an index by using the variables that are appropriate for your geocoding method.

Note: The SASHELP.ZIPCODE data set and the ZIP4 data set from SAS Maps Online are optimized for use with the GEOCODE procedure. Additionally, data sets that you convert by using the %GCDMEL9 and %MAXMIND autocall macros are indexed automatically. No modifications are needed for any of these data sets. △

Note: The STREET geocoding data sets that are provided by SAS are already indexed for the GEOCODE procedure. △

For ZIP+4 geocoding, you should create a simple index on the ZIP variable and a compound index on the ZIP and ZIP+4 variables.

For RANGE geocoding, you should sort your lookup data set by the key variable, and then create a simple index with the key variable. You should sort the range data set by the beginning IP address variable, and then create two simple indexes for the beginning and ending IP address variables.

For more information, see "Understanding SAS Indexes" in the *SAS Language Reference: Concepts*.

Loading Data Sets Into Memory

You can load your lookup data sets into memory by using the SASFILE statement. Loading data into memory reduces I/O processing and can improve the speed of your geocoding operation. You should test your geocoding operations with the lookup data sets loaded into memory to determine whether there is sufficient memory and whether your performance is increased.

For more information, see "SASFILE Statement" in the *SAS Language Reference: Dictionary*.

Procedure Syntax

PROC GEOCODE <*option(s)*>;

PROC GEOCODE Statement

Identifies the data set that contains the address data that you want to geocode. You can also specify an output data set, the geocoding method, alternate names for geocoding variables, and additional attributes variables to associate with the matched addresses.

Syntax

PROC GEOCODE <*option(s)*>;

option(s) can be one or more of the following:
DATA= *address-data-set*
ADDRESSCITYVAR= *character-variable*
ADDRESSPLUS4VAR= *variable*
ADDRESSSTATEVAR= *character-variable*
ADDRESSVAR= *variable*
ADDRESSZIPVAR= *variable*
ATTRIBUTEVAR= *variable-list*
BEGINRANGEVAR= *numeric-variable*
ENDRANGEVAR= *numeric-variable*
FIPS= *FIPS-data-set*
LOOKUP= *lookup-data-set*
LOOKUPCITYVAR= *character-variable*
LOOKUPKEYVAR= *variable*
LOOKUPPLUS4VAR= *variable*
LOOKUPSTATEVAR= *character-variable*
LOOKUPSTREET= *street-matching-data-set*
LOOKUPVAR= *variable*
LOOKUPXVAR= *numeric-variable*
LOOKUPYVAR= *numeric-variable*
LOOKUPZIPVAR= *variable*
METHOD= *geocoding-method*
NOCITY
NOZIP
NOSTIMER
OUT=*output-data-set*
RANGEDATA= *data-set*
RANGEDECIMAL
RANGEKEYVAR= *variable*
TYPE= *street-type-data-set*

Options

To facilitate converting existing SAS/GIS batch geocoding programs that use the %GCBATCH autocall macro to the GEOCODE procedure, the option name from the %GCBATCH autocall macro is an acceptable alias for most options. For more information, see the *SAS/GIS: Spatial Data and Procedure Guide*.

DATA= *address-data-set*
specifies the SAS data set that contains address observations that you want to geocode. If you do not specify this option, then the most recently created SAS data set is used.

Note: The character variables in your input address data set must be left-aligned. That is, the values must not contain leading spaces. You can use the LEFT function in a DATA step to align your data if necessary. △

ADDRESSCITYVAR= *character-variable*
specifies the character variable in the input address data set that contains the city names.

Default: CITY

ADDRESSPLUS4VAR= *variable*
specifies the variable in the input address data set that contains ZIP+4 extensions. The variable can be either numeric or character, but it must be the same type as the ZIP+4 variable in the lookup data set (LOOKUPPLUS4VAR=).

Default: PLUS4

ADDRESSSTATEVAR= *character-variable*
specifies the character variable in the input address data set. This variable contains the two-character postal code for state (for example, NY).

Default: STATE

ADDRESSVAR= *variable*
for STREET geocoding, specifies the variable in the address data set that contains the street address values (for example, "1229 North Main St.")

For CUSTOM and RANGE geocoding, the ADDRESSVAR= option specifies the variable in the address data set that contains non-address input values. The variable can be character or numeric. This is used together with the LOOKUPVAR= option to geocode with unconventional values. Examples include internal sales territories, Metropolitan Statistical Areas (MSA), and Internet Protocol (IP) addresses.

Default: For STREET geocoding, the default name is ADDRESS.

ADDRESSZIPVAR= *variable*
specifies the variable in the input address data set that contains the 5-digit ZIP code values. The variable can be either numeric or character, but it must be the same type as the ZIP code variable in the lookup data set (specified by the LOOKUPZIPVAR= option).

Note: The values for the ZIP code variable must be five digits. You can use the Z5. format to prepend leading zeros to any ZIP code values that have fewer than five digits. △

Default: ZIP

ATTRIBUTEVAR= (*variable-1*, *variable-2*, ...*variable-n*)
lists non-geocoding variables in the lookup data set that are to be added to the output data set. Examples include county, census block, and time zone. Variable names can be separated by commas or spaces.

Note: The values for additional attribute variables are not added to observations in output data set where the match type is “City mean” or “ZIP mean”. △

Note: If an attribute variable has the same name as a variable in the address data set, then that variable is not added to the output data set. △

Note: For the STREET geocoding method, only attribute variables from the street segment lookup data set can be included. △

Example: ATTRIBUTEVAR=(STATENAME, COUNTYNM)

BEGINRANGEVAR= *variable*
specifies the numeric variable in the your range data set that contains the beginning IP address for each range of addresses.

ENDRANGEVAR= *variable*
specifies the numeric variable in the your range data set that contains the ending IP address for each range of addresses.

FIPS= *FIPS-data-set*
specifies a SAS data set that is used STREET geocoding method to convert two-character postal codes and city names into US FIPS codes.

Note: The values of the city and state variables in the FIPS data set must be uppercase. △

Default: The SASHELP.PLFIPS data set.

LOOKUP= ***lookup-data-set***
specifies a SAS data set that associates coordinates with addresses. The data set is searched for observations that match the address observations. The variables that are required for your lookup data set depend on your geocoding method. See "Alternate ZIP Code and ZIP+4 Lookup Data Sets" on page 1150.

The data set can also include other attribute variables (such as COUNTY, TIME ZONE, AREA CODE) that can be added to the address observation by using the ATTRIBUTEVAR= option.

Note: The character variables in your lookup data set must be left-aligned. That is, the values must not contain leading spaces. You can use the LEFT function in a DATA step to align your data if necessary. △

Default: For the ZIP geocoding method, the SASHELP.ZIPCODE data set is the default. For other methods, you must specify the LOOKUP= option.

LOOKUPCITYVAR= ***character-variable***
specifies the character variable in the lookup data set that contain the city names.

Default: CITY

LOOKUPKEYVAR= ***variable***
specifies the key variable for the lookup data set. The values of the key variable correspond to values in the variable that you specify for the RANGEKEYVAR= option. The data type of the key variable must match the variable that you specify for the RANGEKEYVAR= option.

LOOKUPPLUS4VAR= ***variable***
specifies the variable in the lookup address data set that contains ZIP+4 extensions. The variable can be either numeric or character, but it must be the same type as the ZIP+4 variable in the input address data set (ADDRESSPLUS4VAR=).

Default: PLUS4

LOOKUPSTATEVAR= ***character-variable***
specifies the character variable in the lookup data set that contains the two-character postal code for the state or province.

Default: STATECODE

LOOKUPSTREET= ***street-matching-data-set***
specifies the street matching data set for associating coordinates with addresses when performing STREET geocoding.

The GEOCODE procedure expects the street matching data set to have a name that ends with M. The library must also contain two corresponding datasets whose names end with S (segment) and P (coordinate). For example, if you specify the street matching data set MYMAPS.STREETM, then the MYMAPS library must also contain the STREETS and STREETP data sets.

For more information about the data sets for STREET geocoding, see "Data Sets for Street Geocoding" on page 1163.

Default: The SASHELP.USM data set. You can download the USM, USS, and USP data sets for the entire United States from SAS Maps Online Web site at **`www.sas.com/mapsonline`**.

LOOKUPVAR= ***variable***
specifies the variable in the lookup data set that contains non-address values. The variable can be character or numeric. This is used together with the ADDRESSVAR=

option to geocode with unconventional values. Examples include internal sales territories, Metropolitan Statistical Areas (MSA), and Internet Protocol (IP) addresses.

LOOKUPXVAR= ***numeric-variable***
specifies the numeric variable in the lookup data set that contains the longitude of the geocoding location.

Default: X

LOOKUPYVAR= ***numeric-variable***
specifies the numeric variable in the lookup data set that contains the latitude of the geocoding location.

Default: Y

LOOKUPZIPVAR= ***variable***
specifies the variable in the lookup data set that contains the five-digit ZIP code values. The variable can be either character or numeric, but it must be the same type as ZIP code variable in the input address data set (ADDRESSZIPVAR=).

Note: The values for a character ZIP code variable must be five digits. You can use the Z5. format to prepend leading zeros to any ZIP code values that have fewer than five digits. △

Default: ZIP

METHOD= ***geocoding-method***
specifies the geocoding method. This parameter is optional. Specify one of the following:

CITY specifies the CITY geocoding method. The GEOCODE procedure attempts to match the city and state from the address data set with the lookup data set. Separate city and state variables are required in the address and lookup data sets. If multiple matches are found, then the coordinates of the matches are averaged.

Note: The city and state matching method is case insensitive. △

Requirements:
CITY geocoding requires the LOOKUP= option.
CITY geocoding also uses the following options:

ADDRESSCITYVAR=
ADDRESSSTATEVAR=
LOOKUPCITYVAR=
LOOKUPSTATEVAR=
LOOKUPXVAR=
LOOKUPYVAR=

If your data does not use the default variable names for any of these options, then you must specify the options that do not use the default.

CUSTOM specifies the CUSTOM geocoding method. The GEOCODE procedure attempts to match custom variables that you specify by using the LOOKUPVAR= and ADDRESSVAR= variables. Examples include internal sales territories and Metropolitan Statistical Areas (MSA).

Requirements: CUSTOM geocoding requires the following options:

ADDRESSVAR=
LOOKUP=
LOOKUPVAR=

If your lookup data set does not use the default variable names for X and Y, then the LOOKUPXVAR= and LOOKUPYVAR= options are required.

PLUS4 specifies the PLUS4 geocoding method. The GEOCODE procedure attempts to match the five-digit ZIP code and ZIP+4 extension from the address data set with the lookup data set. If no match is found, then the ZIP method is used instead. If multiple ZIP matches are found, then the coordinates of the matches are averaged.

Interaction: You can disable the secondary matching by using the NOZIP option.

Requirements: PLUS4 geocoding requires the LOOKUP= option. PLUS4 geocoding also uses the following options:

ADDRESSPLUS4VAR=
ADDRESSZIPVAR=
LOOKUPPLUS4VAR=
LOOKUPZIPVAR=
LOOKUPXVAR=
LOOKUPYVAR=

If your data does not use the default variable names for any of these options, then you must specify the options that do not use the default.

RANGE specifies the RANGE geocoding method. The GEOCODE procedure attempts to match an Internet Protocol (IP) address from the address data set to a range of IP addresses from the range data set. If a match is found, then a key variable is used to match the IP address to a set of coordinates in the lookup data set.

Note: This feature is for SAS 9.2 Phase 2 and later. △

Requirements: RANGE geocoding requires the following options:

ADDRESSVAR=
BEGINRANGEVAR=
ENDRANGEVAR=
LOOKUP=
LOOKUPKEYVAR=
RANGEKEYVAR=

If your lookup data set does not use the default variable names for X and Y, then the LOOKUPXVAR= and LOOKUPYVAR= options are required.

STREET specifies the STREET geocoding method. The GEOCODE procedure attempts to match the street name and ZIP code. If no

match is found, then the GEOCODE procedure attempts to match the street name, city name, and two-character postal code. If the second match fails, then the ZIP method and the CITY method are used instead.

If a street match is found, X and Y coordinate values are interpolated by using the house number, street type suffix, directional prefix, and directional suffix from the input address.

Note: This feature is for the third maintenance release of SAS 9.2 and later. △

For more information about the STREET geocoding method, see "Street Geocoding" on page 1162.

Interaction: You can disable the secondary ZIP matching by using the NOZIP option.

You can disable the secondary CITY matching by using the NOCITY option.

Requirements: STREET geocoding uses the following options:

ADDRESSCITYVAR=
ADDRESSSTATEVAR=
ADDRESSZIPVAR=
ADDRESSVAR=
FIPS=
LOOKUPCITYVAR=
LOOKUPSTATEVAR=
LOOKUPSTREET=
LOOKUPZIPVAR=
LOOKUPXVAR=
LOOKUPYVAR=
TYPE=

If your data does not use the default variable names for any of these options, then you must specify the options that do not use the default.

The following options are not required if you specify the NOCITY option:

ADDRESSCITYVAR=
ADDRESSSTATEVAR=
LOOKUPCITYVAR=
LOOKUPSTATEVAR=

The following options are not required if you specify the NOZIP option:

ADDRESSZIPVAR=
LOOKUPZIPVAR=

ZIP specifies the ZIP code geocoding method. The GEOCODE procedure attempts to match the five-digit ZIP code from the address data set with the lookup data set. If no match is found, then the CITY method is used instead. If multiple CITY matches are found, then the coordinates of the matches are averaged.

Interaction: You can disable the secondary matching by using the NOCITY option.

Requirements: ZIP geocoding uses the following options:

ADDRESSCITYVAR=
ADDRESSSTATEVAR=
ADDRESSZIPVAR=
LOOKUPCITYVAR=
LOOKUPSTATEVAR=
LOOKUPZIPVAR=
LOOKUPXVAR=
LOOKUPYVAR=

If your data does not use the default variable names for any of these options, then you must specify the options that do not use the default.

The following options are not required if you specify the NOCITY option:

ADDRESSCITYVAR=
ADDRESSSTATEVAR=
LOOKUPCITYVAR=
LOOKUPSTATEVAR=

Default: ZIP

Interaction: If you specify more than one method, then the last method that you specify is used.

NOCITY
disables the secondary matching attempt by city and state if STREET or ZIP code geocoding does not find a match.

By default, if ZIP code geocoding does not find a match, or if STREET geocoding does not find a match for the street address or ZIP code, then the GEOCODE procedure attempts to match the city and state values and then averages the results.

Interaction: You cannot use the NOCITY option with the CITY geocoding method.

NOSTIMER
disables the informational messages sent to the SAS log that tracks the progress of the geocoding operation. If the input data set includes 1,000 or more observations, then the GEOCODE procedure writes periodic messages to the SAS log showing the percentage completed and estimated time remaining. This option disables those messages.

Note: If you do not specify this option (because you want the status messages) and your input data set has 1,000 or more observations, and you are still not receiving periodic status messages, then check the setting of the LOGPARM system parameter. Set LOGPARM="WRITE=IMMEDIATE" to cause messages to be written immediately to the SAS log rather than buffered for later output. △

NOZIP
disables the secondary matching attempt by ZIP code when PLUS4 or STREET geocoding do not find a match. By default, if PLUS4 or STREET geocoding do not find a match, then the GEOCODE procedure attempts to match the five-digit ZIP code and average each matching ZIP code coordinate.

Note: If your data set contains many missing ZIP+4 values, then the NOZIP option might improve performance. △

Interaction: You cannot use the NOZIP option with the ZIP geocoding method.

OUT= *output-data-set*
specifies a data set for the geocoded addresses. All of the variables in the input address data set are copied to the output data set. Also added to the output data set are the following:

- X and Y variables for the location of the match
- optional variables specified by the ATTRIBUTEVAR option
- a variable named _MATCHED_ indicating how the match was made (by ZIP code, by city and state, by averaging coordinates, or no match)

If the output data set that you specify already exists, then it is replaced without warning. If the output data set is the same as the input data set, then the input data set is updated by the geocoding operation.

If you omit the OUT= option, then the name of the output data set is DATA*n*, where *n* is the smallest integer that produces a unique name. For example, if the DATA1 data set exists, then the default name for the output data set is DATA2.

RANGEDATA= *data-set*
specifies a data set that associates ranges of IP addresses with locations. The data set should contain variables that identify the starting IP number, ending IP number, and location ID for each range of IP addresses.

RANGEDECIMAL
specifies that the values of the ADDRESSVAR= variable are in decimal form. By default, the IP addresses in the ADDRESSVAR= variable are in dotted quad notation. For example, the IP address 192.168.0.1 is represented as 3232235521 in decimal form.

RANGEKEYVAR= *variable*
specifies the key variable for the lookup data set. The values of the key variable correspond to values in the variable that you specify for the LOOKUPKEYVAR= option. The data type of the key variable must match the variable that you specify for the LOOKUPKEYVAR= option.

TYPE= *type-data-set*
specifies a SAS data set that is used by the STREET geocoding method to standardize variations of common street address elements. For example, the type data set might standardize "parkway", "parkwy", and "pkwy" to a standard form "pkwy" to facilitate matching.

Default: The SASHELP.GCTYPE data set.

Street Geocoding

Overview of Street Geocoding

Note: This feature is for the third maintenance release of SAS 9.2 and later. △

The STREET geocoding method matches street addresses to coordinates on a map. The GEOCODE procedure attempts to match the street name and ZIP code. If no match is found, then the GEOCODE procedure attempts to match the street name, city

name, and two-character postal code. If the second match fails, then the ZIP method and the CITY method are used in succession.

If a street match is found, X and Y coordinate values are interpolated by using the house number, street type suffix, directional prefix, and directional suffix from the input address.

Note: To obtain the best results from STREET geocoding, use the most complete street addresses possible in your input data set. For example, "111 North Main Street" might produce a more accurate result than a "111 Main Street" or "111 North Main." Also include ZIP code values in the input data set to improve the accuracy of your results. △

Data Sets for Street Geocoding

Overview of the Required Data Sets

The STREET geocoding method requires five different lookup data sets:

street matching data set
: contains street names, ZIP codes, FIPS codes, and references to observation numbers for the street segment data set. The name of this data set must end with the letter M.

  The STREET geocoding method uses the street matching data set to match the street name and to determine which observations in the street segment data set are associated with the matching street.

  The FIRST variable identifies the first observation in the street segment data set and the LAST variable identifies the last observation in the street segment data set that is associated with the street match.

  Default: the SASHELP.USM data set

  Tip: The street matching data set is specified by the LOOKUPSTREET= option. See the documentation for the LOOKUPSTREET= option for more information.

street segment data set
: contains variables to identify the street type, street direction prefix, and street direction suffix. Each street segment is associated with a range of house numbers, which is specified by the FROMADD and TOADD variables. The START variable identifies the first observation in the street coordinate data set that is associated with the street segment. The N variable specifies the number of observations in the street coordinate data set that are associated with the street segment.

  The street segment data sets that are provided by SAS contain attribute variables with additional information, such as census tracts and county FIPS codes. The name of this data set must end with the letter S.

  Default: the SASHELP.USS data set

  Tip: The street segment data set is specified indirectly through the LOOKUPSTREET= option. See the documentation for the LOOKUPSTREET= option.

street coordinate data set
: contains X and Y coordinates. The name of this data set must end with the letter P.

  Default: the SASHELP.USP data set

  Tip: The street coordinate data set is specified indirectly through the LOOKUPSTREET= option. See the documentation for the LOOKUPSTREET= option.

street FIPS data set

contains city names, two-character postal codes, and FIPS codes. If a match cannot be found by using the street name and ZIP code, then the STREET geocoding method uses the FIPS data set to determine the FIPS code for the city name and two-character postal code of the input address.

Note: If you choose to create a customized version of this data set, then you must use the same variable names, data types, order, and index that are used in the SASHELP.PLFIPS data set. △

Default: the SASHELP.PLFIPS data set

Tip: The street FIPS data set is specified by the FIPS= option.

street type data set
contains street type suffixes. The STREET geocoding method uses the street type data set to convert street type suffixes to standardized forms.

Note: If you choose to create a customized version of this data set, then you must use the same variable names, data types, order, and index that are used in the SASHELP.GCTYPE data set. △

Default: the SASHELP.GCTYPE data set

Tip: The street type data set is specified by the TYPE= option.

Obtaining Street Lookup Data Sets

The default street matching data sets (USM, USS, and USP) are not installed with SAS/GRAPH. These data sets contain address lookup data for the entire United States. You can download these data sets from the SAS Maps Online Web site at **www.sas.com/mapsonline**.

The USM, USS, and USP data sets are created from US Census Bureau TIGER/Line files. As new TIGER/Line data is released, updated versions of the USM, USS, and USP data sets will be made available.

The GEOEXM, GEOEXS, and GEOEXP data sets in the SASHELP library are installed with SAS by default. These data sets contain data for Wake County in North Carolina in the United States.

Output Variables for Street Geocoding

In addition to the default output variables, the STREET geocoding method creates the following variables in the output data set:

M_ADDR — contains the street address for the match. The M_ADDR value is the match value from the lookup data set.

M_CITY — contains the city name for the match. The M_CITY value is the match value from the lookup data set.

M_STATE — contains the two-character postal code for the match. The M_STATE value is the match value from the lookup data set.

M_ZIP — contains the ZIP code value from the lookup data set.

M_OBS — contains the row number for the match in the lookup data set.

STATUS — indicates the type of match that was found. The following values are used with the _STATUS_ variable:

City/State Match
The street address did not match but a match was found for the city name and two-character postal code.

Found
The street address matched.

ZIP Match
The street address did not match but a match was found for the ZIP code.

(Blank)
No match was found.

NOTES contains tokens that provide additional information about the match. For more information, see “Street Geocoding Note Values” on page 1165.

SCORE Contains a numeric value that indicates an estimate of the relative accuracy of the match.

Street Geocoding Note Values

The STREET geocoding method creates a _NOTES_ variable in the output data set. This variable provides details about the quality of the address match by using token strings. For example, the value "AD ZC NM" contains three tokens that indicate that the street name, ZIP code, and house number matched.

Each token in the _NOTES_ value has an associated score, and the sum of the scores make up the value of the _SCORE_ variable.

The following table displays the tokens and their scores:

Table 39.1 Tokens for the _NOTES_ Variable

Token	Score	Description
AD	20	The street name matched.
CT	5	The city name matched.
DP	10	The street direction prefix matched.
DS	10	The street direction suffix matched.
ENDNM	0	The house number was outside the ranges of values in the lookup data set for the matching street. The geocoded coordinates for the nearest end of the street were used.
MZC	0	Multiple matches were found for the street address and ZIP code.
MZS	0	Multiple matches were found for the street address and city-state pair.
NM	20	The house number matched.

Token	Score	Description
NMOS	15	The house number matched an address range in the lookup data set, but is on the opposite side of the street from the matched range.
NOCT	-5	The city name and postal code could not be matched in the FIPS data set.
NODPA	-10	The input address had no direction prefix but the matching street did have a direction prefix. For example, the input street name was "Main St." but the matching street was "N Main St."
NODPM	-10	The input address had a direction prefix but it was not the same as the direction prefix of the matching street. For example, the input street name was "North Main St." but the matching street was "Main St."
NODSA	-10	The input address had no direction suffix but the matching street did have a direction suffix. For example, the input street name was "Johnson Ave" but the matching street was "Johnson Ave S."
NODSM	-10	The input address had a direction suffix but it was not the same as the direction suffix of the matching street. For example, the input street name was "Johnson Ave South" but the matching street was "Johnson Ave."
NOLNM	0	The lookup data set contains missing values for the house numbers of the matching street. The geocoded coordinates for the center of the matching street were used.
NONM	0	The input address has no house number. The geocoded coordinates for the center of the matching street were used.

Token	Score	Description
NOTYA	-5	The input address had no street type suffix, but the matching address did have a street type suffix. For example, the input address was "110 Main St." but the matching address was "110 N. Main St."
NOTYM	-5	The street type suffix of the input address was not the same as the type suffix of the matching street. For example, the input street name was "Park St." but the matching street name was "Park Ave."
ST	5	The two-character postal code matched.
TY	5	The street type suffix matched.
ZC	15	The ZIP code matched.

Examples

Example 1: Geocoding Using Default Values

Sample library member: GEOSMPL

The following sample shows the simplest form of the GEOCODE procedure specifying only the OUT= option to geocode by five-digit ZIP code. The default lookup data set, SASHELP.ZIPCODE, is used.

Generate the input data set of addresses to geocode.

```
data CUSTOMERS (label="Customer data for geocoding");
infile datalines dlm='#';
   length address $ 24 city $ 24 state $ 2;
   input address  /* House number and street name */
         zip      /* Customer ZIP code (numeric)  */
         city     /* City name                    */
         state    /* Two-character postal code    */   ;
   cust_ID = _n_;   /* Assign customer ID number    */
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
115 E. Water St # 19901 # Dover #
760 Moose Lodge Road # 19934 # Camden #
200 S. Madison Str # 19801 # Wilmington # DE
4701 Limestone Road # 19808 # Wilmington #
```

```
2117 N 4th St # 19363 # Oxford # PA
1313 Mockingbird Lane # . # Delray # CC
133 Silver Lake Dr # 19971 # Rehoboth Beach # DE
11 SE Front Street # 19963 # Milford # DE
402 Nylon Boulevard # . # Seaford # DE
363 E Commerce St # . # Smyrna # DE
5595 Polly Branch Rd # 19975 # Selbyville # DE
1209 Coastal Highway # 19944 # Fenwick Island # DE
2899 Arthursville Rd # 19953 # Hartly # DE
41 Bramhall St # . #  #
9320 Old Racetrack Rd # . # Delmar # DE
281 W Commerce Str # 19955 # Kenton #
211 Blue Ball Road # 21921 # Elkton # MD
3893 Turkey Point Rd # 19980 # Woodside # DE
;
run;
```

Run the GEOCODE procedure, and then print the output data set.

```
proc geocode out=geocoded_customers;
run;
proc print data=geocoded_customers noobs;
run;
```

The result of using all of the default values is that the following is true:

- The input address data set is the most recently created SAS data set (this example assumes that you have just created WORK.CUSTOMERS).
- The ZIP code geocoding method is used.
- The lookup data set is SASHELP.ZIPCODE.
- No variables are added to the output data set other than the X and Y coordinates, and a _MATCHED_ variable indicating whether and how the match was made.

The following output from PROC PRINT shows the output data set after running the GEOCODE procedure. Notice that the following geocoding variables have been added:

- location coordinate variables X and Y from the lookup data set (SASHELP.ZIPCODE)
- a variable named _MATCHED_ indicating whether the location was found by matching ZIP codes or by matching City and State (or whether no location was found because no match was made)

Output 39.1 The GEOCODED_CUSTOMERS Data Set

Y	X	_MATCHED_	address	zip	city	state	cust_ID
34.0695	-118.398	City mean	555 Junk Street	99999	Beverly Hills	CA	1
39.1500	-75.532	ZIP	115 E. Water St	19901	Dover		2
39.0953	-75.570	ZIP	760 Moose Lodge Road	19934	Camden		3
39.7366	-75.549	ZIP	200 S. Madison Str	19801	Wilmington	DE	4
39.7317	-75.669	ZIP	4701 Limestone Road	19808	Wilmington		5
39.7877	-75.961	ZIP	2117 N 4th St	19363	Oxford	PA	6
.	.	None	1313 Mockingbird Lane	.	Delray	CC	7
38.7265	-75.081	ZIP	133 Silver Lake Dr	19971	Rehoboth Beach	DE	8
38.9035	-75.432	ZIP	11 SE Front Street	19963	Milford	DE	9
38.6387	-75.611	City	402 Nylon Boulevard	.	Seaford	DE	10
39.2912	-75.606	City	363 E Commerce St	.	Smyrna	DE	11
38.4663	-75.150	ZIP	5595 Polly Branch Rd	19975	Selbyville	DE	12
38.4593	-75.053	ZIP	1209 Coastal Highway	19944	Fenwick Island	DE	13
39.1509	-75.693	ZIP	2899 Arthursville Rd	19953	Hartly	DE	14
.	.	None	41 Bramhall St	.			15
38.4557	-75.574	City	9320 Old Racetrack Rd	.	Delmar	DE	16
39.2282	-75.666	ZIP	281 W Commerce Str	19955	Kenton		17
39.6264	-75.850	ZIP	211 Blue Ball Road	21921	Elkton	MD	18
39.0695	-75.567	ZIP	3893 Turkey Point Rd	19980	Woodside	DE	19

Example 2: Adding Additional Variables to the Output Data Set

Procedure features:

ATTRIBUTEVAR=, DATA=, OUT=

Sample library member: GEOVARS

The following example illustrates using the ATTRIBUTEVAR= option to add additional variables (from the lookup data set) to the output data set. The example also illustrates using the DATA= option to specify an input address data set.

Generate the input data set of addresses to geocode.

```
data CUSTOMERS (label="Customer data for geocoding");
infile datalines dlm='#';
   length address $ 24 city $ 24 state $ 2;
   input address  /* House number and street name */
         zip      /* Customer ZIP code (numeric)  */
         city     /* City name                    */
         state    /* Two-character postal code    */   ;
   cust_ID = _n_;   /* Assign customer ID number    */
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
115 E. Water St # 19901 # Dover #
760 Moose Lodge Road # 19934 # Camden #
200 S. Madison Str # 19801 # Wilmington # DE
4701 Limestone Road # 19808 # Wilmington #
2117 N 4th St # 19363 # Oxford # PA
1313 Mockingbird Lane # . # Delray # CC
133 Silver Lake Dr # 19971 # Rehoboth Beach # DE
```

```
11 SE Front Street # 19963 # Milford # DE
402 Nylon Boulevard # . # Seaford # DE
363 E Commerce St # . # Smyrna # DE
5595 Polly Branch Rd # 19975 # Selbyville # DE
1209 Coastal Highway # 19944 # Fenwick Island # DE
2899 Arthursville Rd # 19953 # Hartly # DE
41 Bramhall St # . #  #
9320 Old Racetrack Rd # . # Delmar # DE
281 W Commerce Str # 19955 # Kenton #
211 Blue Ball Road # 21921 # Elkton # MD
3893 Turkey Point Rd # 19980 # Woodside # DE
;
```

Geocode the data, and then print the output data set.

```
proc geocode method=ZIP                          /* Geocoding method        */
             data=customers                      /* Address data            */
             out=geocoded_customers              /* Output data set         */
             attributevar=(statename, countynm); /* Include these variables */
run;

proc print data=geocoded_customers noobs;
  var x y _matched_ statename countynm address zip;
run;
```

The following output from PROC PRINT shows the output data set after running the GEOCODE procedure. Notice that the following variables have been added to the output data set:

- location coordinate variables X and Y from the lookup data set (SASHELP.ZIPCODE)
- a variable named _MATCHED_ indicating whether the location was found by matching ZIP codes or by matching City and State (or whether no location was found because no match was made)
- a variable named STATENAME from the lookup data set (that contains the full name of the state or territory)
- a variable named COUNTYNM from the lookup data set (that contains the name of the county or parish)

The attribute variables STATENAME and COUNTYNM are missing where the value for _MATCHED_ is “None.” The attribute variables are also missing where _MATCHED_ is “City mean”—these observations were matched with multiple city-and-state observations in the lookup data set, so the correct values for the attribute variables cannot be determined.

Output 39.2 The GEOCODED_CUSTOMERS Data Set with Additional Variables

X	Y	_MATCHED_	STATENAME	COUNTYNM	address	zip
-118.398	34.0695	City mean			555 Junk Street	99999
-75.532	39.1500	ZIP	Delaware	Kent	115 E. Water St	19901
-75.570	39.0953	ZIP	Delaware	Kent	760 Moose Lodge Road	19934
-75.549	39.7366	ZIP	Delaware	New Castle	200 S. Madison Str	19801
-75.669	39.7317	ZIP	Delaware	New Castle	4701 Limestone Road	19808
-75.961	39.7877	ZIP	Pennsylvania	Chester	2117 N 4th St	19363
.	.	None			1313 Mockingbird Lane	.
-75.081	38.7265	ZIP	Delaware	Sussex	133 Silver Lake Dr	19971
-75.432	38.9035	ZIP	Delaware	Sussex	11 SE Front Street	19963
-75.611	38.6387	City	Delaware	Sussex	402 Nylon Boulevard	.
-75.606	39.2912	City	Delaware	Kent	363 E Commerce St	.
-75.150	38.4663	ZIP	Delaware	Sussex	5595 Polly Branch Rd	19975
-75.053	38.4593	ZIP	Delaware	Sussex	1209 Coastal Highway	19944
-75.693	39.1509	ZIP	Delaware	Kent	2899 Arthursville Rd	19953
.	.	None			41 Bramhall St	.
-75.574	38.4557	City	Delaware	Sussex	9320 Old Racetrack Rd	.
-75.666	39.2282	ZIP	Delaware	Kent	281 W Commerce Str	19955
-75.850	39.6264	ZIP	Maryland	Cecil	211 Blue Ball Road	21921
-75.567	39.0695	ZIP	Delaware	Kent	3893 Turkey Point Rd	19980

Example 3: Street Geocoding

Sample library member: GEOSTRT

The following example illustrates the STREET geocoding method to obtain coordinates based on street addresses. The ATTRIBUTEVAR= option specifies an additional variable to include in the output data set.

Create the input data set.

```
data WORK.CUSTOMERS (label='Input data for street geocoding');
  infile datalines dlm='#';
    length address $ 32
           city    $ 24
           state   $ 2;
   input address  /* House number and street name */
         zip      /* Customer ZIP code (numeric)  */
         city     /* City name                    */
         state;   /* Two-character postal code    */
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
305 Cross Lake Drive # 27526 # Fuquay-Varina # NC
2525 Banks Road # 27603 # Raleigh # NC
2222 SAS Campus Drive # 27513 # Cary # NC
1150 SE Maynard Rd. # 27511 # Cary # NC
2117 Graceland # 27606 # Raleigh # NC
1313 Mockingbird Lane # # Delray # CC
133 Jade Circle # 27545 # Knightdale # NC
1005 W South St # 27603 # Raleigh # NC
N Winds North Drive # 27591 # Wendell # NC
622 Roundabout Road # 27540 # Holly Springs # NC
```

```
Johnson Family Rd # 27526 # #
822 Water Plant Road # # Zebulon # NC
502 Possum Track Road # 27614 # # NC
2590 Wolfpack Lane # 27604 # Raleigh # NC
125 Ferris Wheel Ct # 27513 # Cary # NC
;
run;
```

Geocode the data and then print the output data set.

```
proc geocode                          /* Invoke geocoding procedure       */
   method=STREET                      /* Specify geocoding method         */
   data=WORK.CUSTOMERS                /* Input data set of addresses      */
   out=WORK.GEOCODED                  /* Output data set with X/Y values  */
   lookupstreet=SASHELP.GEOEXM        /* Primary street lookup data set   */
   attributevar=(TRACTCE00);          /* Assign Census Tract to locations */
run;
proc print data=WORK.GEOCODED noobs;
   var address m_addr m_zip m_obs _matched_ _status_  _notes_ _score_ x y tractce00;
run;
```

The following output from the PRINT procedure shows the output data set after running the GEOCODE procedure. In addition to the default output variables, the TRACTCE00 attribute variable was added.

Output 39.3 The GEOCODED Data Set

address	M_ADDR	M_ZIP	M_OBS	_MATCHED_	_STATUS_
555 Junk Street		.	.	City mean	City/State Match
305 Cross Lake Drive	305 Cross Lake Dr	27526	5098	Street	Found
2525 Banks Road	2525 Banks Rd	27603	1189	Street	Found
2222 SAS Campus Drive	199 Sas Campus Dr	27513	16786	Street	Found
1150 SE Maynard Rd.	1150 SE Maynard Rd	27511	12467	Street	Found
2117 Graceland	4400 Graceland Ct	27606	8022	Street	Found
1313 Mockingbird Lane		.	.	None	
133 Jade Circle	173 Jade Cir	27545	9971	Street	Found
1005 W South St	1005 W South St	27603	17643	Street	Found
N Winds North Drive		27591	11009	ZIP	ZIP match
622 Roundabout Road	530 Roundabout Rd	27540	16395	Street	Found
Johnson Family Rd	Johnson Family Rd	27526	10151	Street	Found
822 Water Plant Road	822 Water Plant Rd	27597	20351	Street	Found
502 Possum Track Road	502 Possum Track Rd	27614	15179	Street	Found
2590 Wolfpack Lane	2590 Wolfpack Ln	27604	21291	Street	Found
125 Ferris Wheel Ct	125 Ferris Wheel Ct	27513	6994	Street	Found

NOTES	_SCORE_	X	Y	TRACTCE00
CT ST	10	-118.398	34.0695	.
AD ZC NMOS TY	55	-78.763	35.6061	53104
AD ZC NM TY	60	-78.673	35.6369	53103
AD ZC ENDNM TY	40	-78.763	35.8273	53510
AD ZC NM DP TY	70	-78.764	35.7835	53501
AD ZC ENDNM NOTYA	30	-78.711	35.7889	52402
	0	.	.	.
AD ZC ENDNM TY	40	-78.461	35.8147	54102
AD ZC NM DP TY	70	-78.654	35.7732	51000
ZC	15	-78.388	35.7930	.
AD ZC ENDNM TY	40	-78.831	35.6505	53200
AD ZC NONM TY	40	-78.750	35.5395	53104
AD CT ST NMOS TY	50	-78.339	35.8319	54302
AD ZC NM TY	60	-78.629	35.9520	53801
AD ZC NMOS TY	55	-78.609	35.8234	52705
AD ZC NM TY	60	-78.800	35.7949	53515

CHAPTER
40

The GFONT Procedure

Overview

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 on page 1199.

About 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 "Creating Fonts" on page 1187 for details.

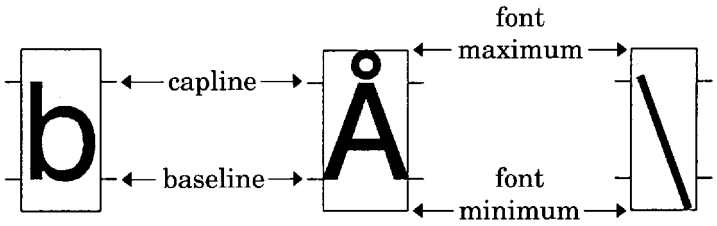
Concepts

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.

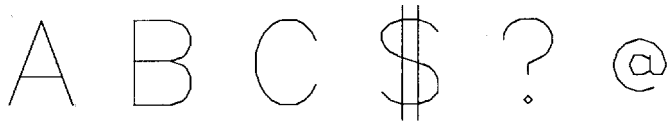
Figure 40.1 Font Characteristics Terminology



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.

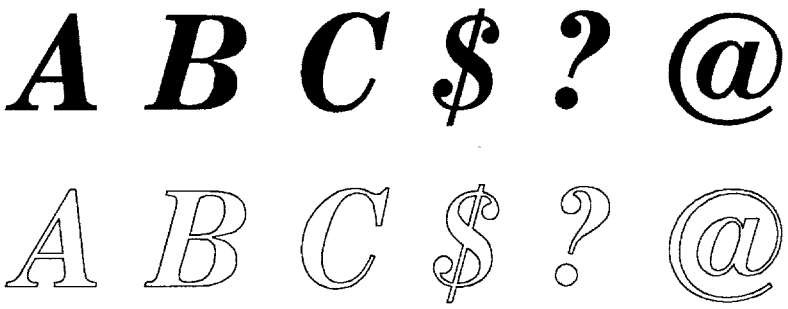
Figure 40.2 Characters from a Stroked Font



- 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 40.3 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, while 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 be made up of a single polygon. The letter C above is a single polygon with one line segment
- A character can be made up 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 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.

To cancel or redefine the libref GFONT*n*, submit the following statement:

```
LIBNAME GFONTn;
```

Procedure Syntax

Requirements: One font name is required. To display a font, include NOBUILD. To create a font, include DATA=.

Global statements: FOOTNOTE, NOTE, TITLE

PROC GFONT NAME=*SAS/GRAPH font* | *device-resident font* | *system font*
mode
<*display-option(s)*>
<*creation-option(s)*>;

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.

Syntax

PROC GFONT NAME=*SAS/GRAPH font* | *device resident font* | *system font*
mode
<*display-option(s)*>
<*creation-option(s)*>;

- *mode* must be one of the following:
  - DATA=*font-data-set*
  - NOBUILD
- *display-option(s)* can be one or more of the following:
  - CTEXT=*text-color*
  - GOUT=<*libref.*>*output-catalog*
  - HEIGHT=*character-height*<*units*>
  - NOKEYMAP
  - NOROMAN
  - NOROMHEX
  - Style element:
    - REFCOL=*reference-line-color*

REFLINES
ROMCOL=*code-color*
ROMFONT=*font*
ROMHEX
ROMHT=*height<units>*
SHOWALL
SHOWROMAN

- *creation-option(s)* can be one or more of the following:

  BASELINE=*y*
  CAPLINE=*y*
  CHARSPACETYPE=DATA | FIXED | NONE | UNIFORM
  CODELEN=1 | 2
  FILLED
  KERNDATA=*kern-data-set*
  MWIDTH=*character-width*
  NODISPLAY
  NOKEYMAP
  RESOL=1...4
  ROMHEX
  SHOWROMAN
  SPACEDATA=*space-data-set*
  UNIFORM

For more detail on using the GFONT syntax, see "Displaying Fonts: Required Arguments and Options" on page 1179 and "Creating Fonts: Required Arguments and Options" on page 1183.

Displaying Fonts: Required Arguments and Options

Required Arguments for Displaying Fonts

NAME=*SAS/GRAPH font* | *device–resident font* | *system font*
specifies of the*SAS*/*GRAPH font* to be displayed. Name can be any of the following values:

- the name of a SAS/GRAPH font stored in the SASHELP.FONTS catalog, and fonts created by the user and stored in a GFONTn catalog. These fonts can be used only by SAS/GRAPH procedures or other procedures that generate *SAS*/*GRAPH* output files.
- the name of 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.
- the name of 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: The *device-resident font name* must be enclosed in quotes.

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

Featured in: Example 1 on page 1199.

.

Options for Displaying Fonts

Options that can be used for either font display or font creation are described here and in "Options for Creating Fonts" on page 1183.

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.

When the syntax of an option includes *units*, specify 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 searched for in the following order:

1. the value of GUNIT= in a GOPTIONS statement
2. the default unit, CELLS

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:

1. the CTEXT= option in the procedure statement
2. the CTEXT= option in a GOPTIONS statement
3. the color specified in the ODS style
4. the first color in the color list

Alias: CT=

Featured in: Example 2 on page 1201.

Note: The CTEXT= value is not stored as part of the font.

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 also: "Specifying the Catalog Name and Entry Name for Your GRSEGs" on page 100

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

Featured in: Example 1 on page 1199.

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:

1. the CTEXT= option in a GOPTIONS statement
2. the color specified in the ODS style
3. 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 1176.

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:

1. the color specified by the CTEXT= option in a GOPTIONS statement
2. 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=

Featured in: Example 1 on page 1199.

Note: The ROMCOL= value is not stored as part of the font.

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:

1. the value of the ODS STYLE variable
2. the FTEXT= option in a GOPTIONS statement
3. SAS-supplied fonts
4. the device-resident font

Alias: RF=

Featured in: Example 1 on page 1199.

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 also: the ROMHEX option on page 1186.

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:

1. the HEIGHT specified in the ODS STYLE
2. the HTEXT= option in a GOPTIONS statement

Alias: RH=

Default: 1

Featured in: Example 1 on page 1199.

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 displays 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

Featured in: Example 1 on page 1199.

Details

```
proc gfont name=weather nobuild romfont=albany;
run;
```

Creating Fonts: Required Arguments and Options

Required Arguments for Creating Fonts

NAME=*font-name*
assigns a name to the font that you create. *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=

Featured in: Example 2 on page 1201.

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 also: "SAS Data Sets" on page 54.

Featured in: Example 2 on page 1201.

When you create a font, define the libref GFONT0. See "Storing User-Created Fonts: GFONT0 Libref" on page 1177.

Note: 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. △

Options for Creating Fonts

Options that can be used for either font display or font creation are described here, and in "Options for Displaying Fonts" on page 1180.

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.

When the syntax of an option includes *units*, specify a unit using one of the following measures:

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

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.

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 40.1 on page 1176 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 are valid values:

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 "The Font Data Set" on page 1187 for details on 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

Featured in: Example 2 on page 1201.

Restriction: If you specify a double-byte character set, the KERNDATA= option and SPACEDATA= option are ignored.

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=

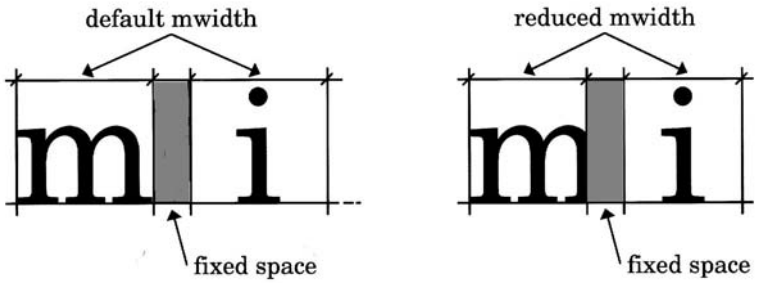
See also: "The Kern Data Set" on page 1196

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.

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 40.4 on page 1185 shows the effect of decreasing the space that is allowed for uniformly spaced characters.

Figure 40.4 Using the MWIDTH= Option to Modify Spacing



See also: the CHARSPACETYPE= option on page 1184 and the UNIFORM option on page 1187

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 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.

CAUTION:
Fonts created with the NOKEYMAP option are 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 also: the NOKEYMAP= option on page 1181 for Displaying Fonts.

RESOL=1...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=

Featured in: Example 2 on page 1201.

ROMHEX
specifies that hexadecimal values display 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 also: the SHOWROMAN option on page 1187, the ROMHEX option on page 1182 for Displaying Fonts, and the NOROMHEX option on page 1181.

SHOWROMAN
specifies that character codes display automatically below the font characters when the GFONT procedure displays the font. If you use the SHOWROMAN option for a font you create, you can later use the NOROMAN option to suppress display of the character codes.

Alias: SR

See also: the ROMHEX option on page 1182, the SHOWROMAN option for Displaying Fonts, and the NOROMAN option on page 1181.

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=

See also: "The Space Data Set" on page 1197.

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.

UNIFORM
specifies that characters are spaced uniformly rather than proportionately. Using the UNIFORM option is the same as specifying CHARSPACETYPE=UNIFORM.

Alias: U

See also: the CHARSPACETYPE= option on page 1184 and the MWIDTH= option on page 1185.

Creating Fonts

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 on page 1201.

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.

See Example 2 on page 1201.

The Font Data Set

The font data set consists of a series of observations that 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.

The variables in the font data set must be assigned certain names and types. The table below summarizes the characteristics of the variables which are described further in “Font Data Set Variables” on page 1189

```
data sashelp. fontdata;
 proc gfont data=fontdata name=mylogo;
run;
```

Specify the font data set with the DATA= argument. The font data set consists of a series of observations that include detailed characteristics of the variables described in “Font Data Set Variables” on page 1189.

Table 40.1 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
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 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 wax 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 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 displays.

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. Assign the LP variable either of the following values:

L — lines

Featured in: Figure 40.5 on page 1190.

Note: Optional for stroked fonts.

Note: Observations that do not contain an LP variable create a shape like the one in Figure 40.5 on page 1190.

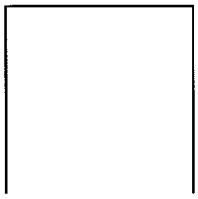
P — polygons. If the observations do not draw a completely closed figure then the figure is closed by the GFONT procedure.

Note: Required for polygons.

Featured in: An LP variable with a value of P for all observations creates a complete box. Figure 40.6 on page 1190

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 40.5 LP Value of L

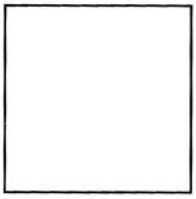


LP (continued)

.

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 40.6 LP Value of P



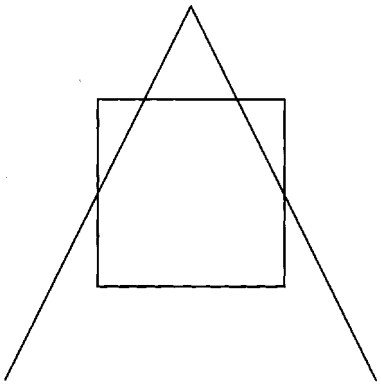
LP (continued)

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 40.7 on page 1191:

OBS	CHAR	SEG	X	Y	LP
1	b	1	1	1	P
2	b	1	1	3	P
3	b	1	3	3	P

OBS	CHAR	SEG	X	Y	LP
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 40.7 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.

Note: If the GFONT procedure encounters the sequence V-C-V in consecutive observations, it draws an arc that connects the two V points and has its center at the C point. If a circle cannot be centered at C, and pass through both V points, the results can be unpredictable.

C — center of a circular arc joining two V points

Restriction: Arcs are limited to 106 degrees or less.

Featured in: Figure 40.8 on page 1192.

Note: After the figure was created, a grid was overlaid, to show the location of the points.

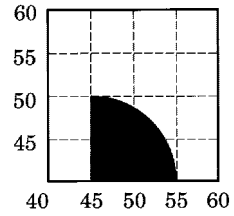
W — width value for CHARSPACETYPE=DATA. An observation with a PTYPE value of W, must always be the first observation for a character. The observation gives the minimum and maximum X values for the character. The Y variable observation contains the maximum X value. Usually, these values include a little extra space for intercharacter spacing.

Restriction: Use a PTYPE of W only if you have specified CHARSPACETYPE=DATA; otherwise, the points are ignored.

Featured in: CHARSPACETYPE= option.

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
5	a	1	55	40	P	V	finish drawing the arc at 55,40

Figure 40.8 Using the PTYPE Variable to Create an Arc



PTYPE (continued)

- 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 GFONT procedure CHARSPACETYPE= DATA 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.

The GFONT procedure requires special instructions to do the following:

- create a stroked character with more than one line segment (an E)
- create a polygon character with an opening (A)

To indicate when one line stops and where the next line begins you can do either of the following:

1. Change the segment number when a new line begins. If the value of LP is L (line), a change in segment number causes the following:
   - 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 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.

2 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.

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.

These observations from a data set called BOXES were used to draw the hollow square in Figure 40.9 on page 1195. The data points that form the figure are laid out on a grid shown next to the square.

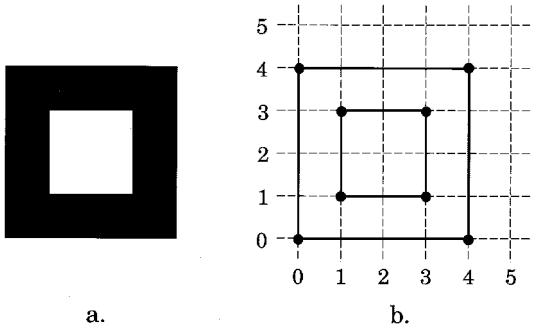
OBS	CHAR	SEG	X	Y	LP
1	b	1	1	1	P
2	b	1	1	3	P
3	b	1	3	3	P
4	b	1	3	1	P
5	b	1	-	-	P
6	b	1	0	0	P
7	b	1	0	4	P
8	b	1	4	4	P
9	b	1	4	0	P

Note 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 40.9 on page 1195 was created with a similar GFONT statement:

```
proc gfont data=boxes name=boxes filled;
```

Note: The FILLED option is included, and only the space between the two squares is filled. △

Figure 40.9 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.

Note: 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 on page 1201.

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

Required kern data set 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. The amount of space to be added or deleted between them is specified. To create a kern data set:

1. Select the pairs of characters to be kerned. Specify the space adjustment (in font units) for each pair.
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 as follows:

```
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
```

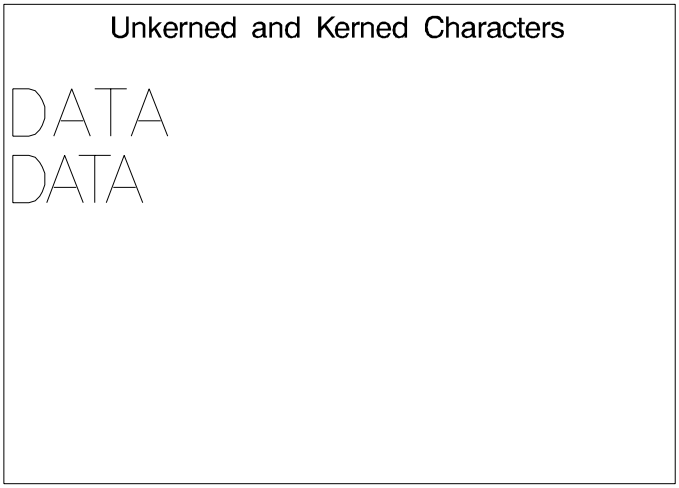
```
                charspacetype=data
                kerndata=kern1
                nodisplay;
    run;
```

"Creating a Kern Data Set" on page 1196 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.

These statements specify the kerned and unkerned fonts, and are used with the GSLIDE procedure to create the fonts:

```
title2 lspace=6 f=font1 h=10 j=l "DATA";
title3 lspace=4 f=font2 h=10 j=l "DATA";
```

Figure 40.10 Comparison of Kerned and Unkerned Text



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. 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

Required space data set 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 the following:

- a point size (SIZE)
- 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:

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;
```

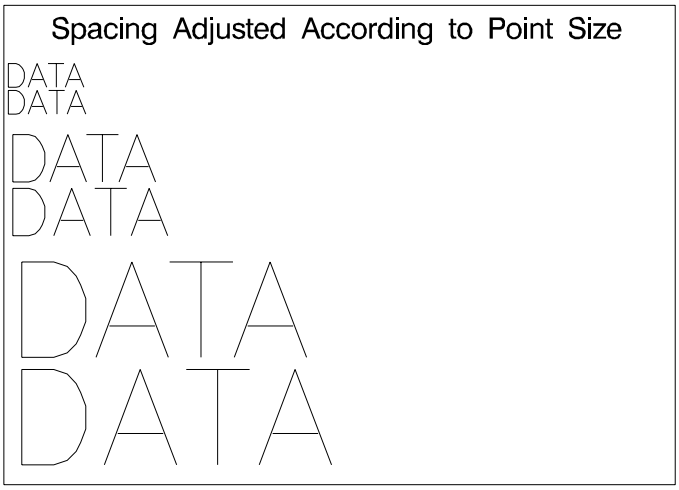
Figure 40.11 on page 1199 illustrates how to use the SPACEDATA= option to create a font in which intercharacter spacing is adjusted according to the height of the characters. The characters A, D, and T are shown as the word DATA. Each pair of lines displays the word DATA and at the same size uses first 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 40.11 on page 1199:

```
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 40.11 Comparison of Text with and without Spacing Adjustments



Examples

These examples illustrate the major features of the GFONT procedure.

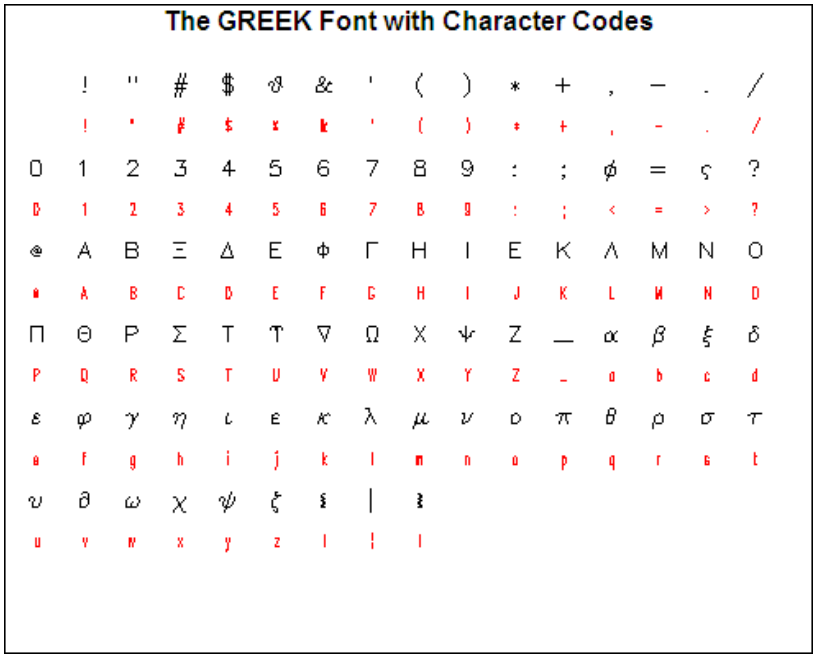
Example 1: Displaying Fonts with Character Codes

Procedure features:

GFONT statement options:

HEIGHT=
NOBUILD
ROMCOL=
ROMFONT=
ROMHT=
SHOWROMAN

Sample library member: GFODISFO



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 you want displayed. The example also illustrates how to modify the appearance of both the font characters, and the character codes.

Set the graphics environment.

```
goptions reset=all border;
```

Define title.

```
title "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=3.7
           romcol=red
           romfont=swissl
           romht=2.7
           showroman;
run;
quit;
```

Example 2: Creating Figures for a Symbol Font

Procedure features:

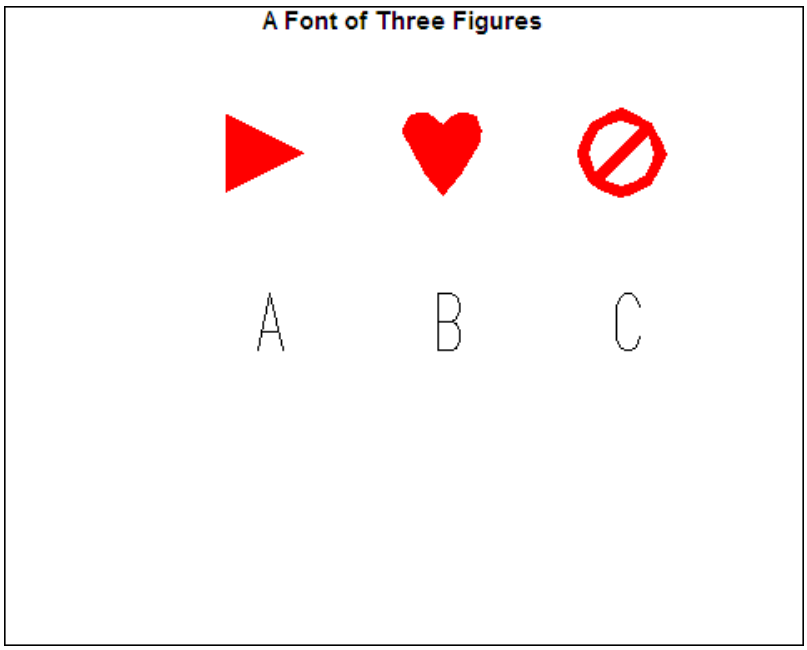
GFONT statement options:

CTEXT=
DATA=
FILLED
NAME=
RESOL=

Other features:

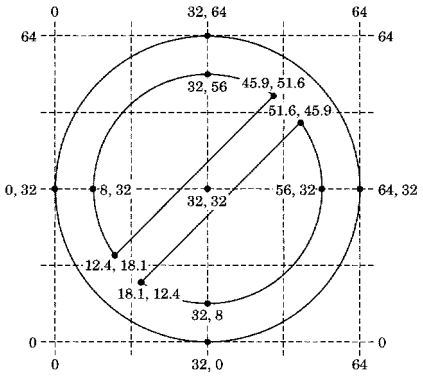
LIBNAME statement

Sample library member: GFOCRFIG



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. Figure 40.12 on page 1202 shows the figure and some of its coordinate points laid out on a grid.

Figure 40.12 Diagram of Circle with Slash Figure



Assign the librefs and set the graphics environment. The LIBNAME statement associates the libref GFONT0 with the SAS data library in which the font catalog is stored.

```
LIBNAME gfont0 "SAS-data-library";
goptions reset=all border;
```

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 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

41

The GINSIDE Procedure

Overview

The GINSIDE procedure compares a data set of X and Y coordinates to a map dataset containing map polygons. The procedure determines whether the X and Y coordinates for each point fall inside of or outside of the map polygons. If the point falls inside of a polygon, then the ID variable is set to the ID value of that polygon. For example, if a map contains states, then the ID variable of the output data set is set to the state that contains the point. The GINSIDE procedure can be used with the SAS/GRAPH map data sets and the results can be used to annotate onto a map with the GMAP procedure.

Note: Points that fall on the border of a polygon might give unpredictable results. △

Procedure Syntax

Requirements: One ID statement is required.

PROC GINSIDE
DATA=*points-data-set*
MAP=*map-data-set*
OUT=*output-data-set* < INSIDEONLY>;
ID *id-variable(s)* ;

PROC GINSIDE Statement

The GINSIDE procedure compares a data set of X and Y coordinates to a map data set containing map polygons and determines whether the X and Y points fall inside or outside of the map polygons.

Requirements: Three data sets are required: a data set containing points, a map data set, and an output data set.

PROC GINSIDE
DATA=*points-data-set*
MAP=*map-data-set*
OUT=*output-data-set* < INSIDEONLY>;

Required Argument

DATA=*points-data-set*
specifies an input data set that contains the X and Y coordinates of the individual points that are being compared to the map polygons.

Note: If this data set contains the same ID variable (or variables) as does the map, the value should be set to MISSING so that the points are not considered to be part of the boundary of the polygon. △

MAP=*map-data-set*
specifies the map data set that contains the polygons that you want to compare the points in the input data set to. This data must conform to the rules for a map data set and contain variables X and Y and one or more ID variables. The ID statement should name that variable or variables.

Note: The X and Y values in the input data set must be in the same projection system and units as the X and Y in the map data set. So, if the map data set has unprojected X and Y values in radians, then the point data set X and Y variable values must also be unprojected and in radians. △

OUT=*output-data-set*
specifies the output data set for the GINSIDE procedure. The output data set contains all of the observations and variables from the input data set, and an ID variable is added.

Options

INSIDEONLY
causes the output data set to contain only points that are inside the map polygons. By default, the data set contains all points.

ID Statement

Specifies the identification variables in the map data set whose polygons will be checked against the points from the input data set.

Requirements: At least one *id-variable* is required.

ID *id-variable(s)*;

Required Arguments

id-variable(s)
specifies one or more identification variables from the map data set. For each X and Y point in the input data set and each ID variable that you specify, if the point lies within a polygon in the map data set, then the ID value of that polygon is written to the output data set. If the point lies outside of all of the polygons, then a missing value is written to the output data set.

For example, the first observation in an input data set contains the values **X=1.37** and **Y=.68**. In the PROC GINSIDE statement, you specify the MAPS.COUNTIES data set, and in the ID statement you specify the STATE variable. The point (1.37,.68) lies within the polygon where STATE=54, so the first value for STATE in the output data set is 54.

Examples

Example 1: Determining Values by Using the GINSIDE Procedure

Procedure features:
ID statement
Sample library member: GINSIDE2

This example uses the GINSIDE procedure to determine the state and county for each pair of coordinates in the input data set.

Create the GPS data set. The X and Y variables are converted from decimal degrees to radians. The X variable is also multiplied by -1 to match the values in the MAPS.COUNTIES data set.

```
data gps;
  input x y site $;
  x=x*arcos(-1)/180;
  x=x*(-1);
```

```
  y=y*arcos(-1)/180;
datalines;
-77.0348 40.0454 a
-78.4437 39.1623 b
-78.4115 39.3751 c
-78.7646 40.6354 d
;
run;
```

Determine the values of STATE and COUNTY for each data point.

```
proc ginside data=gps map=maps.counties out=gpscounties;
  id state county;
run;
```

Sort and print the output data set.

```
proc sort data=gpscounties;
  by site;
run;

proc print data=gpscounties;
  var site state county x y;
run;
```

Output 41.1 shows the values of STATE and COUNTY for each observation in the input data set.

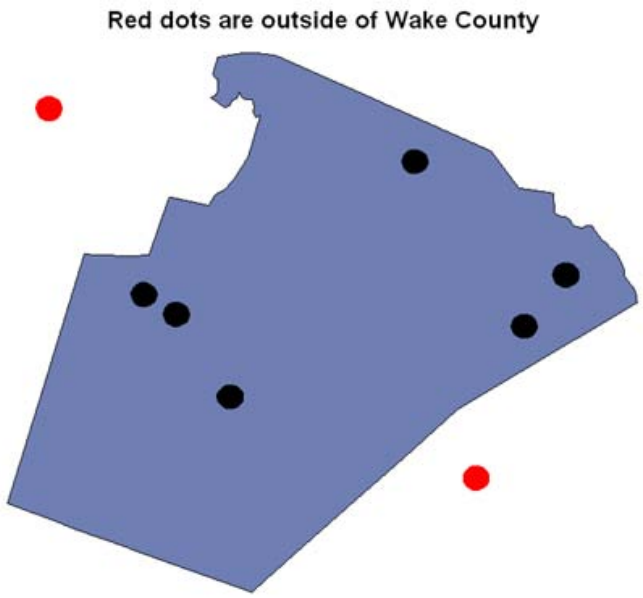
Output 41.1 Proc PRINT Results of Output Data Set

site	STATE	COUNTY	x	y
a	42	133	1.34451	0.69892
b	54	27	1.36910	0.68351
c	54	27	1.36854	0.68723
d	42	21	1.37470	0.70922

Example 2: Mapping and Annotating Values from the GINSIDE Procedure

Procedure features:
ID statement

Sample library member: GINSIDE



The following example determines which customers are inside Wake County in the state of North Carolina. It then draws a map and colors the dots (representing customers) to distinguish customers inside the county from customers outside the county. This example is featured in the SAS Sample Library under the name GINSIDE.

Set the graphics environment.

```
goptions reset=global border;
```

Create the customer data.

```
data customer;
  length city $20;
  input lastname$ zip x y city $;

cards;
Smith    27611 1.374164 0.623436 Raleigh
Jones    27560 1.375948 0.625278 Morrisville
Doe      27513 1.375279 0.624922 Cary
Patel    27520 1.369120 0.621970 Clayton
White    27705 1.377910 0.628629 Durham
Short    27587 1.370373 0.627680 WakeForest
Phillips 27591 1.368124 0.624705 Wendell
Jackson  27597 1.367264 0.625629 Zebulon
;
```

Create a map data set of Wake County in North Carolina.

```
data states;
  set maps.counties(where=(fipstate(state)="NC" and county=183));
run;
```

Combine the CUSTOMER and STATES data sets.

```
data combined;
  set customer states;
run;
```

Project the map and points data sets to use the same projection.

```
proc gproject data=work.combined out=work.combined dupok;
  id state county;
run;

/*split the data*/
data work.states customer;
  set work.combined;
  if missing(zip) = 1 then output work.states;
  else output customer;
run;
```

Determine which customer points fall inside or outside which county.

```
proc ginside map=work.states data=customer out=mapout;
  id state county;
run;
/*see the resulting data*/
proc print;
run;
```

Create an annotate dataset from the points data and color the points black if inside the map, and color them red if outside.

```
data points;
   set mapout;
   length function style color $ 8 position $ 1 text $ 20 ;
   retain xsys ysys "2" hsys "3" when "a" text
"";
   retain rotate 360 style "solid" function "pie" position "5";
   color=
"black";
   size=2;
   if missing(county) then color="red";
   output;
run;
title "Red dots are outside of Wake County";
```

Use the GMAP procedure to display the map. The ANNO= option specifies the annotate data set.

```
proc gmap data=states map=states anno=points;
  id county;
  choro county / coutline=black nolegend;
```

```
run;
quit;
```

Output 41.2 shows the results of PROC PRINT. Notice that the last two observations have missing values for COUNTY because they are not in Wake County.

Output 41.2 Proc PRINT Results for Output Data Set

X	Y	SEGMENT	COUNTY	DENSITY	STATE	city	lastname	zip
-.001541860	-.001337843	2	183	.	.	Raleigh	Smith	27611
-.002986277	0.000506523	3	183	.	.	Morrisville	Jones	27560
-.002444482	0.000149461	4	183	.	.	Cary	Doe	27513
0.001531260	0.002906157	5	183	.	.	WakeForest	Short	27587
0.003358790	-.000065627	6	183	.	.	Wendell	Phillips	27591
0.004053658	0.000860235	7	183	.	.	Zebulon	Jackson	27597
0.002555933	-.002802349	8	.	.	.	Clayton	Patel	27520
-.004565994	0.003861845	9	.	.	.	Durham	White	27705

CHAPTER

42

The GKPI Procedure

Overview

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).

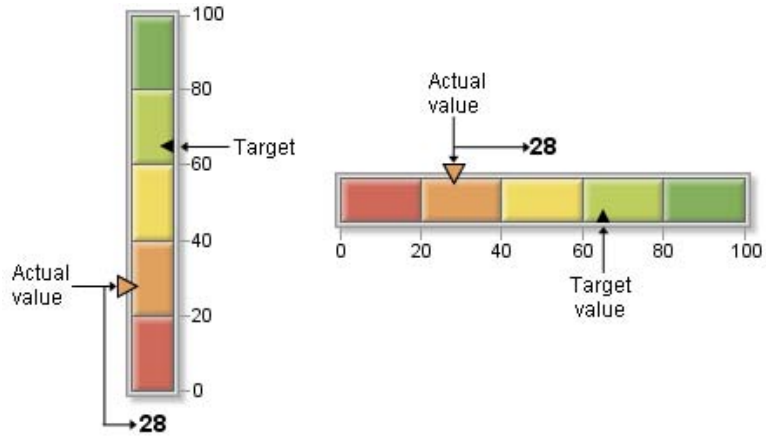
The GKPI procedure produces a two or three-dimensional KPI chart based on a series of segment boundaries and an actual KPI value that you specify. If you specify a target value, the KPI chart also displays the target value. The procedure uses a set of default colors for the KPI chart, but you can specify 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 on the GOPTIONS statement with the GSLIDE or GANNO procedures to generate the GRSEG. △

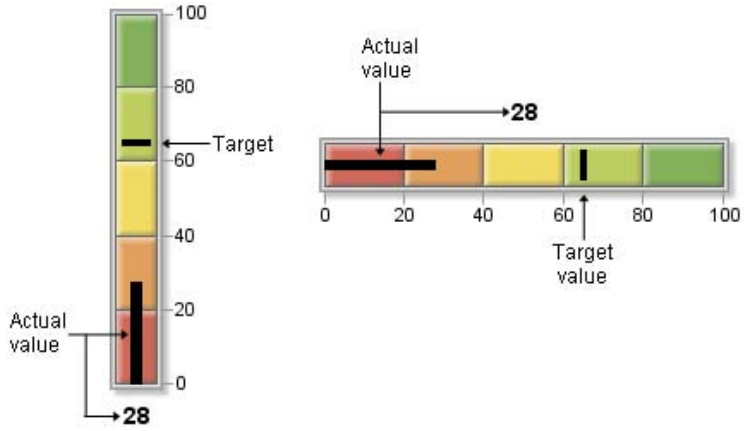
Slider KPI Charts

Slider KPI charts display a bar divided into segments according to the boundary values that you specify. The actual value of the KPI is indicated with a triangle pointer on the top (for a horizontal slider) or the left (for a vertical slider). This actual value indicator is the same color as the segment that contains the actual KPI value. The target value, if it is specified, is displayed as a smaller triangle on the bottom (or right side) of the slider.



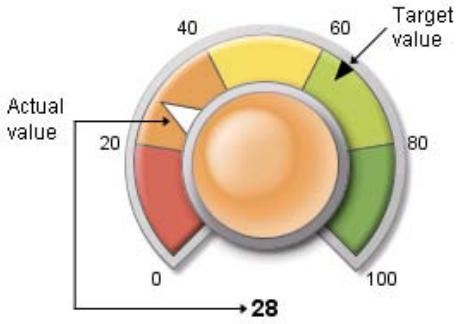
Bullet Graph KPI Charts

Bullet graphs display a bar divided into segments 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.



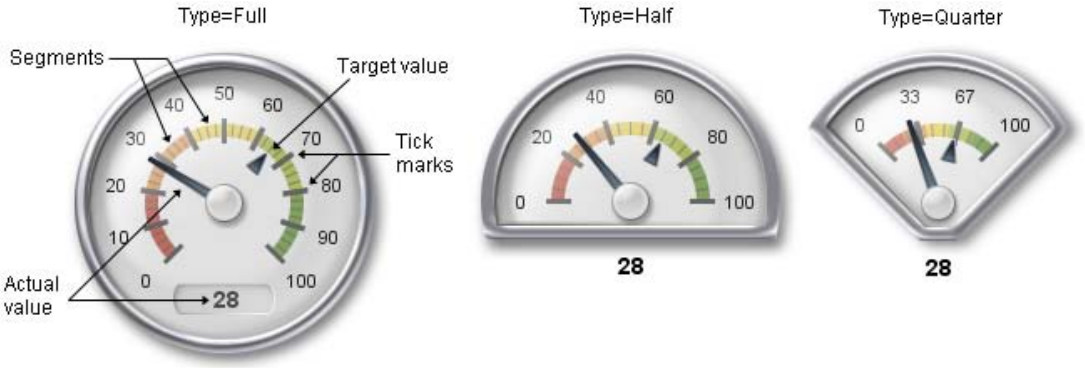
Dial KPI Charts

Dial KPI charts display a dial divided into segments 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 segment that contains the actual KPI value.



Speedometer KPI Charts

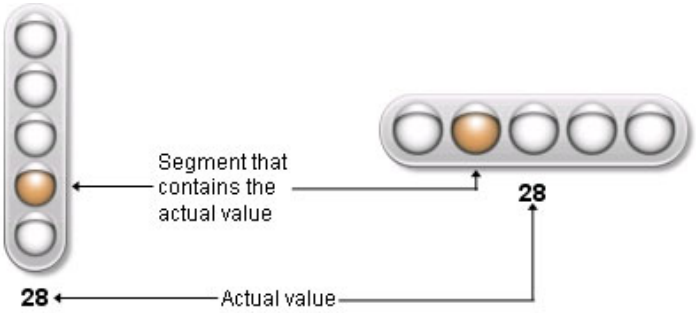
Speedometer KPI charts display a speedometer with the tick marks evenly spaced around the dial and colored segments that correspond to the segment boundaries that you specify. Speedometers can be displayed as a full speedometer, as a half speedometer, or as a quarter speedometer. The actual value of 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 segment 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 segment. The segment that contains the actual value is displayed in color. The remaining segments are gray. In other words, only one "light" is "turned on" at a time. Traffic lights do not display target values.



Concepts

Specifying Basic or Raised Mode

Each KPI chart can be displayed in *basic mode*, which appears flat and two-dimensional, or in *raised mode*, which appears raised and three-dimensional. The default mode is basic mode. You can specify a mode on the PROC GKPI statement:

```
proc gkpi mode=raised;
```

The following figures illustrate the difference in appearance between basic mode and raised mode for each type of KPI chart.

Figure 42.1 Horizontal Slider in Basic and Raised Modes

Bullet graphs appear similar to sliders.

Figure 42.2 Dial in Basic and Raised Modes

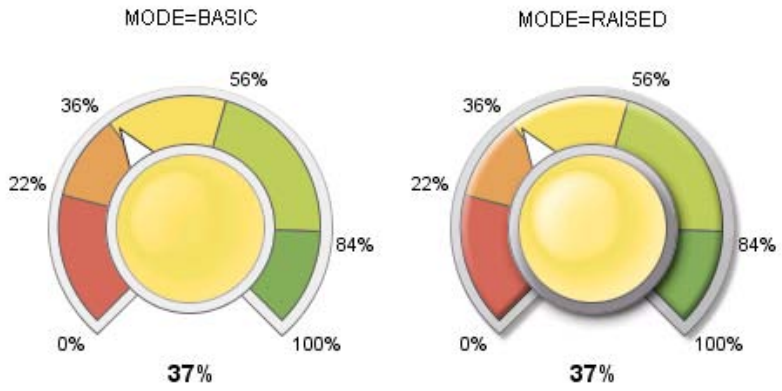


Figure 42.3 Speedometer in Basic and Raised Modes

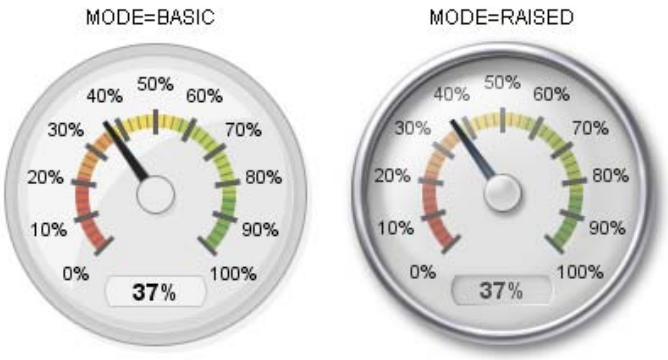
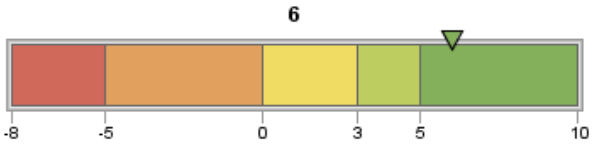


Figure 42.4 Traffic Light in Basic and Raised Modes

Specifying Segment Boundaries and Actual KPI Values

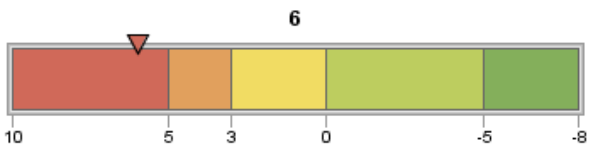
To generate a KPI chart, you must specify a list of segment boundaries using the BOUNDS= option and an actual KPI value using the ACTUAL= option. The values can be positive numbers, negative numbers, or missing (ACTUAL=.), but the BOUNDS= list must be in either ascending or descending order and must contain at least two numbers (in order to define a single segment). For example, the following code defines a horizontal slider with segment boundaries in ascending order from –8 to 10 and an actual KPI value of 6:

```
goptions device=javaimg;
ods html;
proc gkpi;
   hslider actual=6 bounds=(-8 -5 0 3 5 10);
run;
quit;
ods html close;
```



The boundaries can also be specified in desending order, for example:

```
hslider actual=6 bounds=(10 5 3 0 -5 -8)
```



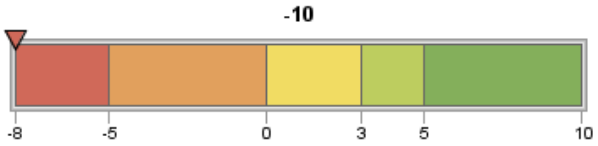
The order in which colors are applied is not affected by whether boundaries are specified in ascending or descending order. See “Defining Active and Inactive Color Lists” on page 1221 for information on controlling segment colors.

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 on 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.

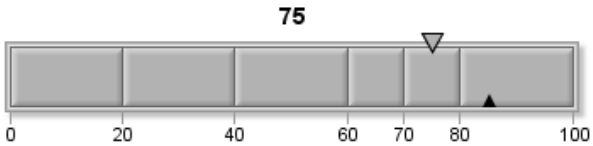
See Example 3 on page 1233 for an example that uses the XPIXELS=, YPIXELS=, BFONT=, and FORMAT= options.

Controlling Segment Colors

Default Colors

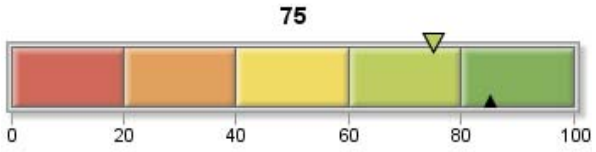
If you define only one segment or more than five segments, the GKPI procedure uses the same value of gray (hexadecimal RGB value B2B2B2) for all segments.

Figure 42.5 GKPI Procedure Gray Scale Default



If you define two to five segments, the GKPI procedure uses some or all of the colors shown in Figure 42.6 on page 1220 as the default colors.

Figure 42.6 GKPI Procedure Default Colors



For example, if you define only three segments, the GKPI procedures uses the red, yellow, and green colors shown in Figure 42.6 on page 1220.

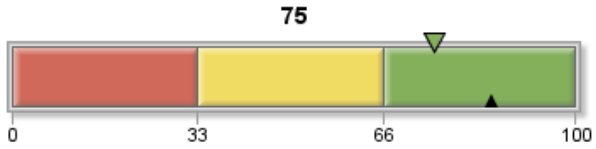


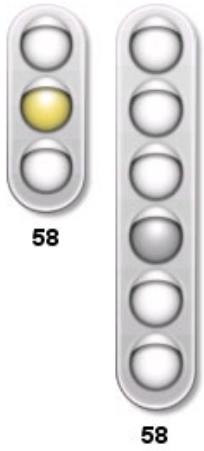
Table 42.1 on page 1220 lists the hexadecimal values for each of these default colors.

Table 42.1 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 segment, then the procedure displays the light for that segment in gray. If you define two to five segments, the traffic light uses the default color listed in Table 42.1 on page 1220 only for the light corresponding to the segment that contains the actual KPI value. All other lights are gray. If you define more than five segments, all lights are displayed in gray, but the light corresponding to the segment 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 42.7 Traffic Light Default Colors



Colors are applied in the same direction, regardless of whether the segment 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 segment in the KPI chart. That is, the first color is applied to the first segment, the second color is applied only to the second segment, and so on. A segment is displayed in its active color if the actual KPI value falls in that segment. All segments 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 segment 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 segments that do not contain the actual KPI value appear gray. △

You can specify colors for the segments using any of the color-naming schemes supported by SAS/GRAPH. See “Specifying Color Names” on page 1223.

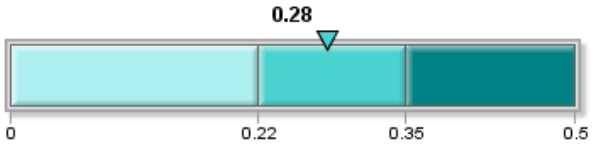
If you specify the COLORS= option, then you must specify a color for each segment. That is, the number of entries in the COLORS= list must be one less than the number

of entries in the BOUNDS= list. If you specify the ACTIVECOLORS= list, you do not have to specify a color for every segment. See “Specifying Active Colors Only for Specific Segments (Using Null Colors)” on page 1223 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 42.6 on page 1220.

```
goptions reset=all device=javaimg;
ods html;
proc gkpi mode=raised;
   hslider actual=0.28
   bounds=(0 .22 .35 .50) /
   colors=(PaleTurquoise MediumTurquoise Teal);
   run;
quit;
ods html close;
```



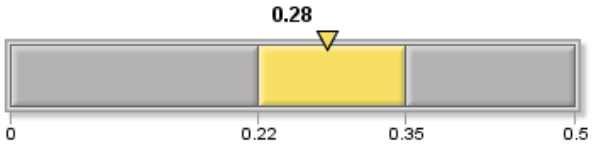
The actual KPI value falls into the second segment, and there are no active colors specified, so the second color in the COLORS= list, MediumTurquoise, is used for the second segment and for the actual KPI value indicator.

Example: Specifying an Active Color List

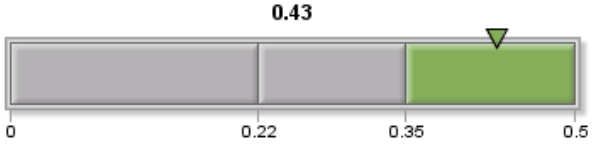
The following example defines the inactive colors for all of the segments to be the same value of gray, cxB2B2B2. For the active colors, it specifies the default values for the red, yellow, and green colors listed in Table 42.1 on page 1220.

```
goptions reset=all device=javaimg;
ods html;
proc gkpi mode=raised;
   hslider actual=0.28
   bounds=(0 .22 .35 .50) /
   colors=(cxB2B2B2 cxB2B2B2 cxB2B2B2)
   activecolors=(cxD06959 cxF1DC63 cx84AF5B);
run;
quit;
ods html close;
```

The actual KPI value is 0.28, which falls into the second segment, so the second color listed in the ACTIVECOLORS= color list, cxF1DC63, which is yellow, is used as the color for the second segment 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 segment instead of gray.



Specifying Active Colors Only for Specific Segments (Using Null Colors)

If you specify a null color for a segment in the ACTIVECOLORS= list, then either the default color or the color in the COLORS= list, if one is specified, is used for that segment 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 segments and you want red to be used only for the lowest and highest segments, 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) will be used for segments two through four even if the actual KPI value falls into one of these segments.

The ACTIVECOLORS= list does not have to specify a color for each segment, but the one-to-one correspondence between the colors that are specified and the segments is maintained. For example, supposed you define five segments 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 segment, and green applies only to the second segment. Default colors (or COLORS= colors) apply to all the other segments.

Specifying Color Names

You can specify colors for the segments using any of the color-naming schemes supported by SAS/GRAPH. For a complete description of these color-naming schemes, see “Color-Naming Schemes” on page 170. The following table shows examples of each of the color naming schemes:

Table 42.2 Examples of Specifying Colors

Color-Naming Scheme	Example
RGB	**COLORS=(cx98FB98 cxDDA0DD cxFFDAB9 cxDB7093 cxB0E0E6)**
CMYK	**COLORS=("FF00FF00" "00FFFF00" "FFFFFF00")**
HLS	**COLORS=(H14055FF H0F060FF H0B485FF H07880FF)**
HSV	**COLORS=(V0F055FF v010FFFF v03BFFFF v12C55E8)**
Gray Scale	**COLORS=(GRAY4F GRAY6D GRAY8A GRAYC3)**

Color-Naming Scheme	Example
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")`

Specifying Fonts

You can control the fonts that are used to display the boundary and tick mark values, the actual KPI values, and the labels with the BFONT=, AFONT=, and LFONT= options, respectively. Each of these options takes a font specification of the following form:

```
<FONT="fontname</BOLD></ITALICS>"> <COLOR=color> <HEIGHT=text-height<units>>
```

FONT= "*fontname*</BOLD></ITALICS>"
specifies the font name. You can specify only system fonts (such as TrueType and UNIX system fonts), not SAS/GRAPH fonts. See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 for more information.

Alias: F=

COLOR=*color*
specifies the text color. You can specify the color in any of the color-naming schemes recognized by SAS/GRAPH.

See "Specifying Color Names" on page 1223 for more information.

Alias: C=

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). The default unit is PCT.

Alias: H=

The LFONT= option also enables you to specify the JUSTIFICATION= option:

```
<FONT="fontname</BOLD></ITALICS>"> <COLOR=color> <HEIGHT=text-height<units>>
<JUSTIFICATION=LEFT|CENTER|RIGHT>
```

JUSTIFICATION=
specifies whether the text is left-justified, centered, or right-justified within the graphics output area. You can specify LEFT, RIGHT, or CENTER. The default is CENTER.

Alias: J=

Procedure Syntax

Type: Template-based (see "Device-Based Graphics and Template-Based Graphics" on page 6)

Requirements:

- You must specify an ODS HTML or ODS RTF statement.
- The only device supported for the GKPI procedure is JAVAIMG. If you do not specify DEVICE=JAVAIMG, then the procedure sets this option automatically.
- At least one DIAL, HSLIDER, HBULLET, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, or VSLIDER statement is required.

Global Statements: FOOTNOTE , GOPTIONS (BORDER, VSIZE, HSIZE, XPIXEL, YPIXEL, IBACK, CBACK, CTEXT, HTEXT, FTEXT only), TITLE

Supports: Run-group processing

PROC GKPI <MODE= BASIC | RAISED>;

DIAL|HBULLET|HSLIDER|HTRAFFICLIGHT|SPEEDOMETER|VTRAFFICLIGHT|VBULLET|VSLIDER ACTUAL=*data-value* BOUNDS=*bound-value-list* </ *options*>;

PROC GKPI Statement

Identifies the mode of the display design.

Syntax

PROC GKPI <MODE= BASIC | RAISED>;

Options

PROC GKPI statement options affect all graphs produced by the procedure.

MODE=BASIC | RAISED
specifies the display mode of KPI chart images. The two choices are as follows:

BASIC	creates a two-dimensional image.
RAISED	creates a three-dimensional image.

The default mode is BASIC. See "Specifying Basic or Raised Mode" on page 1216 examples of each mode.

DIAL, HBULLET, HSLIDER, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, and VSLIDER Statements

Creates a chart in one of seven display types.

Requirements: The ACTUAL= value and the BOUNDS= list are required.

Global statements: TITLE, FOOTNOTE

Supports: Drill-down functionality

Description

The DIAL, HBULLET, HSLIDER, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, and VSLIDER statements specify the type of graphic to be used to display the key performance indicator. One of the following display types is required:

DIAL
specifies a dial.

HBULLET | BULLET
specifies a horizontal bullet graph (a horizontal bar with a horizontal line that represents the actual KPI value).

HSLIDER | SLIDER
specifies a horizontal bar with a triangle that marks the actual KPI value.

HTRAFFICLIGHT
specifies a horizontal traffic light.

SPEEDOMETER
specifies a speedometer.

VTRAFFICLIGHT| TRAFFICLIGHT
specifies a vertical traffic light.

VBULLET
specifies a vertical bullet graph (a vertical bar with a vertical line that represents the actual KPI value).

VSLIDER
specifies a vertical bar with a triangle that marks the actual KPI value.

See “Overview” on page 1213 for examples of each display type.

Syntax

DIAL|HBULLET|HSLIDER|HTRAFFICLIGHT|SPEEDOMETER|VTRAFFICLIGHT|VBULLET|VSLIDER ACTUAL=*data-value*
BOUNDS=(*bounds-list*) </ *options*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options

  ACTIVECOLORS=(*color_1 color_2 color_3 ...color_n-1*)

  AFONT=(<f=“*fontname*</BOLD></ITALICS>”> <c=*color*>
  <h=*text-height*<*units*>>)

AVALUE | NOAVALUE

BFONT=(<f="*fontname*</BOLD></ITALICS>"> <c=*color*> <h=*text-height*<*units*>>)

BVALUE | NOBVALUE

COLORS=(*color_1 color_2 color_3 ...color_n-1*)

FORMAT="*SAS-format*"

LABEL= "*string*"

LFONT=(<f="*fontname*</BOLD></ITALICS>"> <c=*color*> <h=*text-height*<*units*>> <j=*justification*>)

LOWBOUNDARY | NOLOWBOUNDARY

TARGET= *data-value*

TYPE=FULL | HALF | QUARTER

□ output file description options

DESCRIPTION="*description*"

NAME="*name*"

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.

BOUNDS=(*bound_1 bound_2 bound_3 ...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 Segment Boundaries and Actual KPI Values" on page 1218 for more information.

Options

You can specify as many options as needed and list them in any order.

ACTIVECOLORS=(*color_1 color_2 color_3 ...color_n-1*)
specifies the list of active colors for each segment (the colors that you want to use for each segment when that segment contains the actual KPI value). You do not have to specify a color for each segment in the KPI chart. The default colors shown in Figure 42.6 on page 1220 (or the colors specified by the COLORS= option) are used for each segment for which active colors are not specified. The number of entries in the ACTIVECOLORS= list cannot exceed the number of segments 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 Segment Colors" on page 1219 for more information.

AFONT=(<f="*fontname*"</BOLD></ITALICS>> <c=*color*> <h=*text-height*<*units*>>)
specifies the name, color, and text height for the font used for the actual KPI value label. For example `AFONT=(f="Comic Sans MS" c=red h=15pt)`. See "Specifying Fonts" on page 1224 for more information.

Style reference: Font attribute of the GraphLabelText element

AVALUE | NOAVALUE
specifies whether to display the actual KPI value label.

Alias: AVAL | NOAVAL

Default: AVALUE

BFONT=(<f="*fontname*"</BOLD></ITALICS>> <c=*color*> <h=*text-height<units>>*)
specifies the name, color, and text height of the font used for the boundary and tick mark values. For example, **`BFONT=(font="Arial" color=H14055FF height=.25in)`**. See "Specifying Fonts" on page 1224 for more information.

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

BVALUE | NOBVALUE
specifies whether to display the boundary values.

Alias: BVAL | NOBVAL

Default: BVALUE

COLORS=(*color_1 color_2 color_3 ...color_n-1*)
specifies the list of inactive colors for each segment (the colors that you want to be used for each segment when that segment does not contain the actual KPI value). You must specify a color for each segment 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 segments, the GKPI procedure applies a default set of colors ranging from red to green. If you define only one or more than five segments, the default color for all segments is gray.

For traffic lights, the default color for all segments is gray. This option is ignored by the TRAFFICLIGHT statement if the ACTIVECOLORS= option is specified.

See "Controlling Segment Colors" on page 1219 for more information.

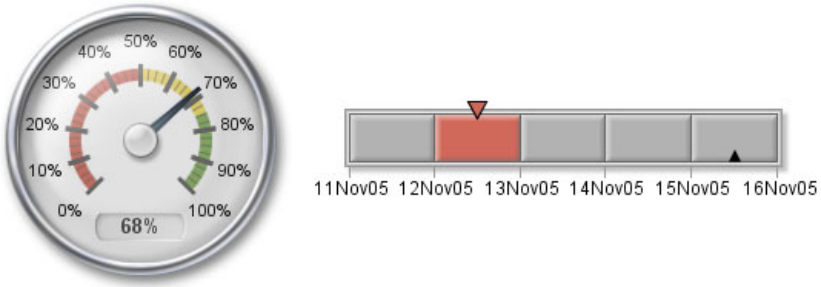
DESCRIPTION= "*description*"
specifies the description of the output. The maximum length for the description is 256 characters. The description does not appear on the graph. The default is "Key performance indicator".

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 on an ODS HTML statement.
- the chart description for Web output. See "Chart Descriptions for Web Presentations" on page 596 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`**.



See "SAS Formats Supported for Java" on page 472 for more information.

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.

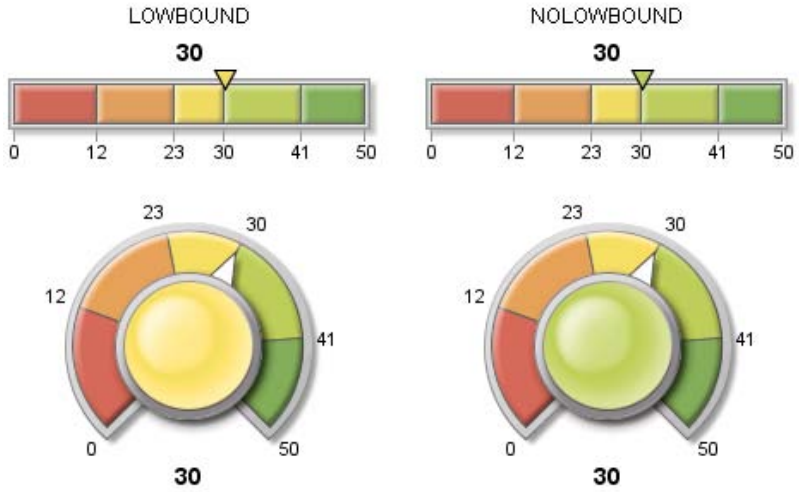
Note: By default, labels are displayed at the top center of the graphics output area, and the KPI chart is displayed in the center of the output area. To reduce the space between labels and the KPI chart, reduce the size of the graphics output area by specifying the XPIXELS=/YPIXELS= or HSIZE=/VSIZE= options on the GOPTIONS statement. See "The Graphics Output and Device Display Areas" on page 59 for more information. △

LFONT=(<f="*fontname*"</BOLD></ITALICS>> <c=*color*> <h=*text-height<units>*> <j=*justification*>)
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. For example: `LFONT=(f="Albany AMT/italics" c=cornflowerblue height=.25cm j=right)`. See the description of LABEL= and "Specifying Fonts" on page 1224 for more information.

Style reference: Font attribute of the GraphLabelText element

LOWBOUNDARY | NOLOWBOUNDARY
specifies whether the KPI chart displays as if the KPI value falls in the lower range segment or the upper range segment when the actual KPI value falls directly on a segment boundary. This option controls the color that is used for dial centers, traffic lights, and actual value indicators. It also controls which segment 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 segment. If you specify NOLOWBOUNDARY, then PROC GKPI uses the color of the higher range segment. Figure 42.8 on page 1230 illustrates the effect of this option on dial centers and on the actual KPI value indicator in a horizontal slider when both a segment boundary and the actual KPI value is 30.

Figure 42.8 LOWBOUND And NOLOWBOUND Effect on Indicator Colors



Alias: LOWBOUND | NOLOWBOUND

Default: LOWBOUNDARY

NAME="*name*"
specifies the name of the graphics output file. The name can be up to 256 characters long, and uppercase characters are converted to lowercase. The default name is **`graph.png`**. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name–for example, **`graph1.png`**.

See also: "About Filename Indexing" on page 99

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

TYPE=FULL | HALF | QUARTER
specifies the size of the display for speedometers. See "Speedometer KPI Charts" on page 1215 for more information.

Restriction: Valid for SPEEDOMETER statement only

Default: FULL

Examples

The following examples illustrate major features of the GKPI procedure. Each of these examples uses the XPIXELS and YPIXELS options on the GOPTIONS statement to scale the KPI charts to a size that would be appropriate for use in a dashboard. These examples also specify STYLE=LISTING option on the ODS HTML statement, which produces a white background instead of the default gray background normally used by the HTML destination.

Note: 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 HTML;** ***procedure-code;*** **ODS HTML CLOSE;**). See "RUN-Group Processing" on page 56 for more information. △

Example 1: Using the Default Colors as the Active Colors

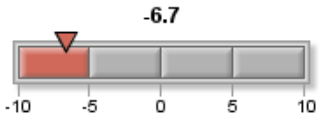
Procedure features:

PROC GKPI statement option: MODE=RAISED

HSLIDER statement options:

ACTUAL=
BOUNDS=
COLORS=
ACTIVECOLORS=

Sample Library Member: GKPGRSLD



The default colors described in "Default Colors" on page 1219 can be used as the active colors instead of the inactive colors. This example uses the same value of gray for all segments for the inactive color. It uses the red, orange, yellow-green, and green colors shown in Figure 42.6 on page 1220 as the active colors.

Set the graphics environment.

```
goptions reset=all device=javaimg xpixels=180 ypixels=110;
```

Close the LISTING destination, and open the HTML destination. Closing the LISTING destination conserves resources. Specifying STYLE=LISTING produces a white background.

```
ods listing close;
ods html style=listing;
```

Generate the KPI chart. Specify the segment 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 segments. The default colors listed in Table 42.1 on page 1220 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, and close the HTML destination. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure. You must close the destination to generate output.

```
quit;
ods html close;
```

Example 2: Creating a Gray Scale Bullet Graph

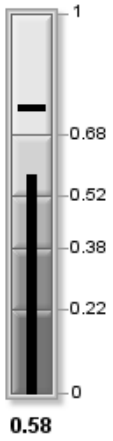
Procedure features:

PROC GKPI statement option: MODE=RAISED

VBULLET statement options:

ACTUAL=
BOUNDS=
COLORS=
TARGET=

Sample Library Member: GKPGRBUL



This example creates a vertical bullet graph. It uses a gray scale color scheme that provides a good contrast between segments. This color scheme can be used in output that is included in publications that are not in color.

Set the graphics environment.

```
goptions reset=all device=javaimg xpixels=130 ypixels=250;
```

Close the LISTING destination, and open the HTML destination. Closing the LISTING destination conserves resources. Specifying STYLE=LISTING produces a white background.

```
ods listing close;
ods html style=listing;
```

Generate the KPI chart. Specify the segment 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, and close the HTML destination. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure. You must close the destination to generate output.

```
quit;
ods html close;
```

Example 3: Creating a Dial KPI Chart

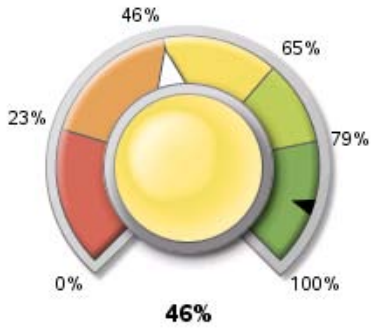
Procedure features:

PROC GKPI statement option: MODE=RAISED

DIAL statement options:

ACTUAL=
AFONT=
BFONT=
BOUNDS=
FORMAT=
NOLOWBOUND
TARGET=

Sample Library Member: GKPDIAL



Set the graphics environment.

```
goptions reset=all device=javaimg xpixels=240 ypixels=200;
```

Close the LISTING destination, and open the HTML destination. Closing the LISTING destination conserves resources. Specifying STYLE=LISTING produces a white background.

```
ods listing close;
ods html style=listing;
```

Generate the KPI chart. Specify the segment boundaries, actual KPI value, and target value. In this case, the target value falls on a segment boundary. The NOLOWBOUNDARY option specifies that the KPI chart behaves as if the actual KPI value falls in the higher range segment. The AFONT= and BFONT= options specify the fonts for the actual value and the boundary segment 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, and close the HTML destination. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure. You must close the destination to generate output.

```
quit;
ods html close;
```

Example 4: Defining a Speedometer

Procedure features:

PROC GKPI statement option: MODE=RAISED

SPEEDOMETER statement options:

ACTUAL=
BOUNDS=
COLORS=
FORMAT=
LABEL=
LFONT=
TARGET=

Sample Library Member: GKPSPD

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.

```
goptions reset=all device=javaimg xpixels=210 ypixels=200;
```

Close the LISTING destination, and open the HTML destination. Closing the LISTING destination conserves resources. Specifying STYLE=LISTING produces a white background.

```
ods listing close;
ods html style=listing;
```

Generate the KPI chart. Specify the segment 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, and close the HTML destination. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure. You must close the destination to generate output.

```
quit;
ods html close;
```

Example 5: Defining a Speedometer with Reversed Colors

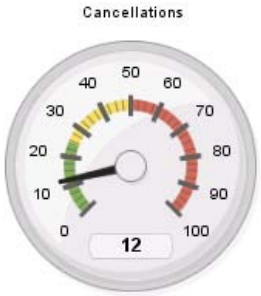
Procedure features:

PROC GKPI statement option: MODE=BASIC

SPEEDOMETER statement options:

ACTUAL=
BOUNDS=
COLORS=
LABEL=

Sample Library Member: GKPSPCLR



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.

```
goptions reset=all device=javaimg xpixels=210 ypixels=200;
```

Close the LISTING destination, and open the HTML destination. Closing the LISTING destination conserves resources. Specifying STYLE=LISTING produces a white background.

```
ods listing close;
ods html style=listing;
```

Generate the KPI chart. Specify the segment boundaries, actual KPI value, target value, and colors. The green, yellow, and red colors listed in Table 42.1 on page 1220 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, and close the HTML destination. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure. You must close the destination to generate output.

```
quit;
ods html close;
```

Example 6: Creating a Traffic Light

Procedure features:

PROC GKPI statement option: MODE=RAISED

TRAFFICLIGHT statement options:

ACTUAL=
BOUNDS=
COLORS=
LABEL=

NOAVALUE

Sample Library Member: GKPTRAFF

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.

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.

```
goptions reset=all device=javaimg xpixels=120 ypixels=210;
```

Close the LISTING destination, and open the HTML destination. Closing the LISTING destination conserves resources. Specifying STYLE=LISTING produces a white background.

```
ods listing close;
ods html style=listing;
```

Generate the KPI chart. Specify the segment 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, and close the HTML destination. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure. You must close the destination to generate output.

```
quit;
ods html close;
```

CHAPTER

43

The GMAP Procedure

Overview

The GMAP procedure produces two-dimensional (choropleth) or three-dimensional (block, prism, and surface) maps that show variations of a variable value with respect to an area. A wide assortment of map data sets is available with SAS/GRAPH software.

Use the GMAP procedure to perform the following tasks:

- produce maps
- summarize data that vary by physical area
- show trends and variations of data between geographic areas
- highlight regional differences or extremes

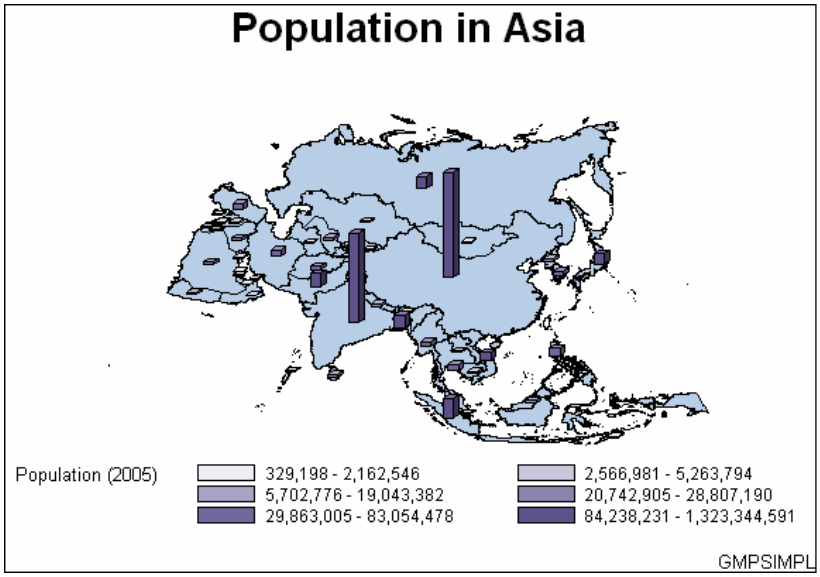
About Block Maps

Block maps display a block at the approximate center of each map area to convey information about response variable values. The height of each block is directly proportional to the value of the response variable.

Note: If the map area consist of multiple, noncontiguous areas, then the block is centered over the largest polygon of the set. For example, in the case of Japan the block is centered over the largest island which is Honshu. △

Figure 43.1 on page 1241 shows a simple block map of the populations of countries in Asia. The population of each country (the response value) is represented by the height of the block.

Figure 43.1 Block Map



The program for this map is in Example 1 on page 1301. For more information on producing block maps, see "BLOCK Statement" on page 1259.

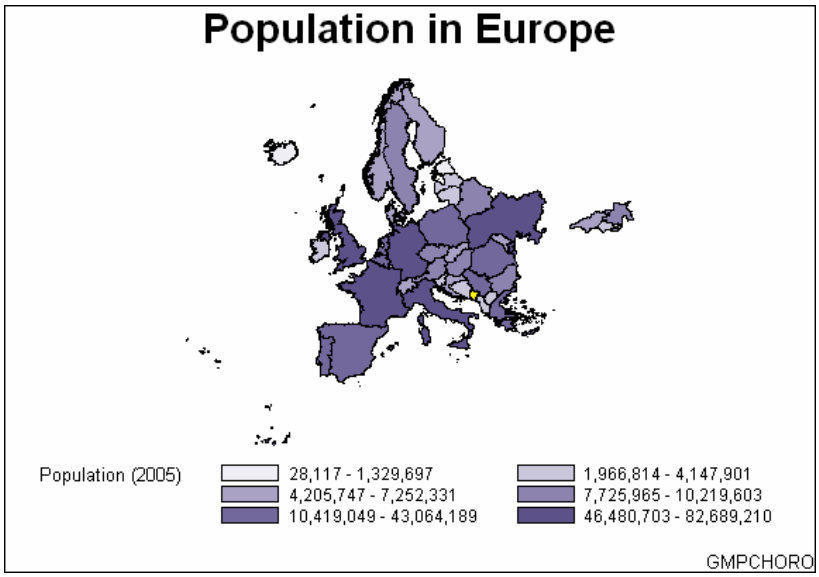
You can assign patterns to the areas in a block map by using the AREA statement. The values of the AREA variable are represented by the pattern of each map area, and the values of the response variable on the BLOCK statement are represented by the height of the blocks. For more information, see "AREA Statement" on page 1255.

About Choropleth Maps

Two-dimensional (choropleth) maps indicate levels of magnitude or response levels of the corresponding response variable by filling map areas with different colors and patterns.

Figure 43.2 on page 1242 shows a choropleth map of the population of countries in Europe. The population of each country (the response value) is represented by the pattern that is assigned to the country.

Figure 43.2 Two-dimensional (Choropleth) Map



The program for this map is in Example 5 on page 1307.

You can also produce a simple choropleth map that shows an outline of a map's areas by specifying your map data set as both the map data set and the response data set in a GMAP statement and adding a PATTERN statement with VALUE=EMPTY. For more information on the PATTERN statement, see "PATTERN Statement" on page 240. For more information on producing choropleth maps, see "CHORO Statement" on page 1269.

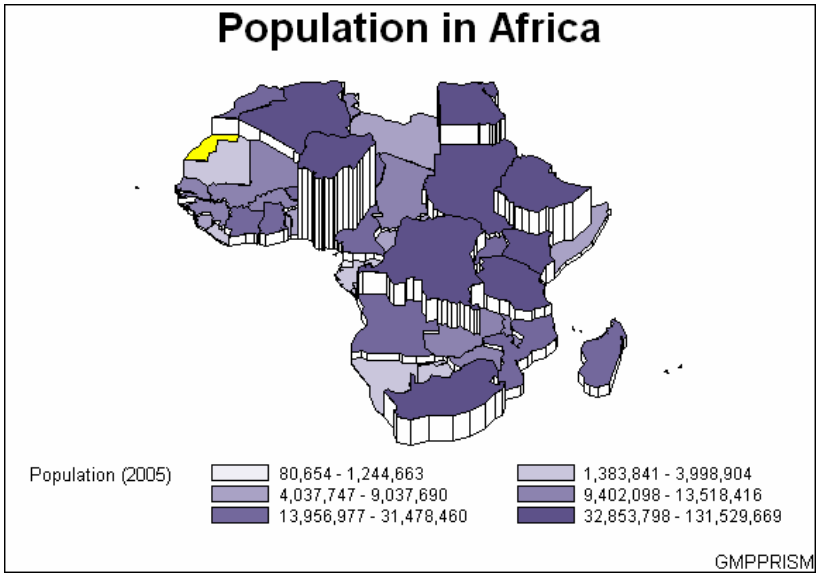
About Prism Maps

Prism maps use polyhedrons (raised polygons) in the shape of each map area to convey information about response variable values. The height of each polyhedron, or prism, is directly proportional to the value of the response variable.

You can alter the perspective of the map by selecting a viewing position (the point in space from which you view the map). You can also change the position of the light source so that the shadowing on the prisms enhances the illusion of height.

Figure 43.3 on page 1243 shows a prism map of the populations of countries in Africa. The population of each country (the response value) is represented by the height of the country and the color of the country's map area.

Figure 43.3 Prism Map



The program for this map is in Example 7 on page 1309. For more information on producing prism maps, see "PRISM Statement" on page 1276.

You can also assign patterns to the areas in a prism map by using the AREA statement. The values of the AREA variable are represented by the pattern of each map area, and the values of the response variable on the PRISM statement are represented by the height of the map areas. For more information, see "AREA Statement" on page 1255.

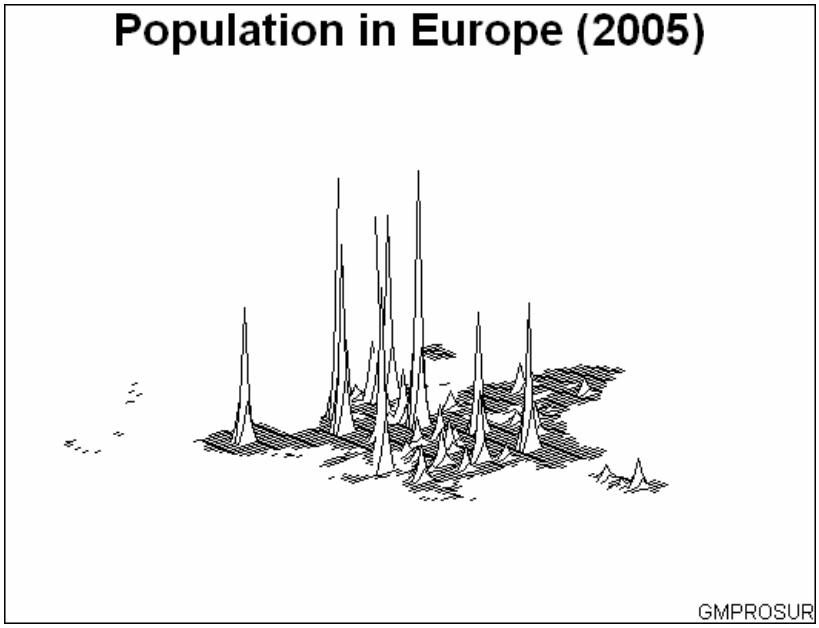
About Surface Maps

Surface maps display a spike at the approximate center of each map area to convey information about response variable values. The height of the spike corresponds to the relative value of the response variable, not to the actual value of the response variable. Thus, a spike that represents a value of 100 might not be exactly 10 times higher than a spike that represents a value of 10. Map area boundaries are not drawn.

Surface maps provide no clear map area boundaries and no legend. Thus, surface maps provide a simple way to judge relative trends in the response data but are an inappropriate way to represent specific response values.

Figure 43.4 on page 1244 shows a surface map of the population growth rates of countries in Europe. The growth rate for each country (the response value) is represented by the height of the spike for that country.

Figure 43.4 Surface Map



The program for this map is in Example 10 on page 1313. For more information on producing surface maps, see "SURFACE Statement" on page 1285.

Concepts

Map data sets and response data sets are used in the GMAP procedure. These data sets must contain the required variables or the procedure stops and you get an error message. The GMAP procedure can take as input a map data set and a response data set, provided that both data sets contain the same ID variable. Alternatively, you can use a single data set as input if it contains either the map data or a variable that references a map data set.

About Map Data Sets

There are two types of data sets that are provided with SAS/GRAPH for mapping: traditional map data sets and feature tables. Much of the map data that is delivered with SAS/GRAPH is available in both the traditional map data set and feature table formats.

SAS/GRAPH software includes a number of predefined map data sets. These data sets are described in "The METAMAPS Data Set" on page 1247.

About Traditional Data Sets

A *traditional map data set* is a SAS data set that contains coordinates that define the boundaries of map areas, such as states or counties.

Required Variables

A traditional map data set must contain at least these variables:

- a numeric variable named X that contains the horizontal coordinates of the boundary points. The value of this variable could be either projected or unprojected. If unprojected, X represents longitude.
- a numeric variable named Y that contains the vertical coordinates of the boundary points. The value of this variable could be either projected or unprojected. If unprojected, Y represents latitude.
- one or more variables that uniquely identify the areas in the map. Map area identification variables can be either character or numeric and are indicated in the ID statement.

The X and Y variable values in the traditional map data set do not have to be in any specific units. They are rescaled by the GMAP procedure based on the minimum and maximum values in the data set. The minimum X and Y values are in the lower-left corner of the map, and the maximum X and Y values are in the upper-right corner.

Map data sets in which the X and Y variables contain longitude and latitude should be projected before you use them with PROC GMAP. See Chapter 46, "The GPROJECT Procedure," on page 1395 for details.

Segment Variable

The traditional map data set can also contain an optional variable named SEGMENT to identify map areas that comprise noncontiguous polygons. Each unique value of the SEGMENT variable within a single map area defines a distinct polygon. If the SEGMENT variable is not present, each map area is drawn as a separate closed polygon that indicates a single segment.

The observations for each segment of a map area in the map data set must occur in the order in which the points are to be joined. The GMAP procedure forms map area outlines by connecting the boundary points of each segment in the order in which they appear in the data set, eventually joining the last point to the first point to complete the polygon. All the segments for each ID value must be contiguous within the map data set.

LONG and LAT Variables

In addition to the variables described in "Required Variables" on page 1244, some of the SAS/GRAPH map data sets also contain the following variables:

- a numeric variable named LONG containing the unprojected longitude (in radians or degrees) of the boundary points
- a numeric variable named LAT containing the unprojected latitude (in radians or degrees) of the boundary points

The GMAP procedure uses the values of the X and Y variables to draw the map. To use the unprojected values to produce a custom map, rename LONG and LAT to X and Y, and then do the following tasks:

1. Rename the LONG and LAT variables to X and Y.
2. Project the coordinates by using the GPROJECT procedure.
3. Use the output data set from GPROJECT as your map data set.

Traditional Map Data Sets Containing X, Y, LONG, and LAT

Most of the traditional map data sets that are provided with SAS/GRAPH software contain four coordinate variables (X, Y, LONG, and LAT). In that case, X and Y are always projected values that are used by the SAS/GRAPH procedures (by default). If you need to use the unprojected values that are contained in the LONG and LAT variables, then do the following tasks:

1 Drop the existing X and Y variables.
2 Rename the LONG and LAT variables to X and Y.

The MAP= value in the GMAP procedure automatically uses X and Y. See "Input Map Data Sets that Contain Both Projected and Unprojected Values" on page 1398 for more details.

Traditional Map Data Sets Containing Only X and Y

The traditional map data sets that contain X and Y variables (and no LONG and LAT variables), are usually projected maps. However, there are a few traditional map data sets for the US and Canada that contain X and Y values that are unprojected longitude and latitude. In this case, you need to use the GPROJECT procedure to project the map (see Chapter 46, "The GPROJECT Procedure," on page 1395).

Note: You can determine whether a SAS traditional map data set is projected or unprojected by looking at the description of each variable that is displayed when you use the CONTENTS procedure or by browsing the MAPS.METAMAPS data set. △

About Feature Tables

An alternative to using the traditional map data set is the feature table. While the traditional map data set stores the spatial information across multiple observations, the feature table uses a data arrangement to store a reference to the spatial information in a single variable value. The feature table's data arrangement uses the $GEOREF SAS/GRAPH format.

$GEOREF format

The $GEOREF format stores spatial information in binary data streams, making it possible to store as a single variable value all the information needed to draw a map area. Thus, the feature tables use only a single observation for each map area, and they treat a field of spatial information just like any other information that can be added to a data set. Each $GEOREF value contains the name of the map data set and the ID variable for that map. The traditional map data set associated with the feature table must be located in the SAS library with the feature table for GMAP to proceed correctly.

The names of the feature tables that are supplied by SAS usually end with the number 2. For example, the feature table for MAPS.AFRICA is MAPS.AFRICA2. You can also determine the feature table for your map data set by referring to the MAPS.METAMAPS data set.

To locate the variable that contains the spatial information, run PROC CONTENTS on a feature table. In the Output window, the variable containing the spatial information has $GEOREF as the value in the column labeled Format.

Note: Some feature tables, like MAPS.CANCENS, have more than one $GEOREF format variable. △

Merging Feature Tables with Response Data Sets

To display response data with a feature table, the feature table must be merged with a response data set. The merged data set is then specified by the DATA= option in the PROC GMAP statement.

First, a PROC SORT must be used to sort the response and feature tables by a variable that is present within both the data sets. Once sorted, the data sets can then be merged with an SQL or DATA step MERGE with the BY variable being the variable

used to sort the data sets. Once the data set is merged, the $GEOREF formatted variable from the feature table becomes the new data set's identification variable to be used in the GMAP procedure. See Example 11 on page 1314 for more details.

The METAMAPS Data Set

In the MAPS library, there is a data set named METAMAPS, which contains metadata about all of the data sets that are delivered in the library. Among the metadata in MAPS.METAMAPS are the following four variables, which you can use to determine which feature table corresponds to a particular geometry table:

Variable	Description
MEMNAME	Identifies the names of all of the data sets that are delivered in the MAPS library.
MEMCODE	Indicates whether a data set represents a feature table (F) or a geometry table (G).
F_TABLE	Indicates the corresponding feature table for a geometry table. This variable is blank for rows that contain metadata about a feature table.
F_GEOCOL	Indicates the variable, in the feature table, whose values encapsulate the geometry object.

For example, consider the data sets MAPS.ASIA, MAPS.STATES, and MAPS.US. Each of these represents a geometry table, and to locate the corresponding feature tables, you would look in MAPS.METAMAPS to find the MEMNAME values ASIA, STATES, and US. Here are the relevant values on those rows:

Table 43.1 Values from the METAMAPS Data Set

MEMNAME	MEMCODE	F_TABLE	F_GEOCOL
Asia	G	ASIA2	CONT95_GEO
STATES	G	US2	GEO_STATE
US	G	US2	_MAP_GEOMETRY_

From these values, you can see that the data sets that are named ASIA, STATES, and US all represent geometry tables because their MEMCODE values are G. The feature table corresponding to the ASIA data set is the data set ASIA2, which stores the spatial information in the variable CONT95_GEO. The feature tables corresponding to STATES and US are both in the data set US2. The spatial information corresponding to STATES is stored in the variable GEO_STATE, and the spatial information corresponding to US is stored in the variable _MAP_GEOMETRY_.

Special Data Sets for Annotating Maps

There are several data sets in the MAPS library that enable you to easily label maps. These data sets contain coordinates for map features such as cities, but cannot be used as map data sets.

MAPS.USCENTER
: contains the coordinates of the visual center of each state in the U.S. and Washington, D.C., as well as coordinates in the ocean for states that are too small

to contain a label. There are two pairs of variables for locating labels using Annotate data sets. The X and Y variables are projected and can be used with the MAPS.US and MAPS.USCOUNTY data sets. The LONG and LAT variables are unprojected longitude and latitude in radians and can be used with the MAPS.STATES, MAPS.COUNTIES, and MAPS.COUNTY data sets.

MAPS.USCITY
contains the locations of selected cities in the U.S. Many city names occur in more than one state, so you might have to subset by state to avoid duplication. There are two pairs of variables for locating labels using Annotate data sets. The X and Y variables contain projected coordinates and can be used with the MAPS.US and MAPS.USCOUNTY data sets. The LONG and LAT variables contain the unprojected longitude and latitude in radians. These can be used to place labels on the MAPS.STATES, MAPS.COUNTIES, or MAPS.COUNTY data sets.

For details on each of these data sets, see the MAPS.METAMAPS data set.

About Response Data Sets

A *response data set* is a SAS data set that contains the following variables:

- one or more response variables that contain data values that are associated with map areas. Each value of the response variable is associated with a map area in the map data set.
- identification variables that identify the map area to which a response value belongs. These variables must be the same as those that are contained in the map data set.

The response data set can contain other variables in addition to these required variables.

Using the Response Data Set with the Map Data Sets

The traditional map data set and the response data set must be used independently in the PROC GMAP statement, where the response data set is specified by the DATA= option and the traditional map data set is specified by the MAP= option. The values of the map area ID variables in the response data set determine the map areas to be included on the map. Unless the ALL option is used in the PROC GMAP statement, only the map areas with response values are shown on the map. As a result, you do not need to subset your map data set if you are mapping only a small section of the map. However, if you map the same small section frequently, then create a subset of the map data set for efficiency.

If you have a response data set named WORK.SITES, then the syntax for using GMAP might resemble the following:

```
proc gmap map=maps.us data=work.sites;
   id state;
   choro region/discrete;
run;
quit;
```

To use data from both a feature table and response data set, merge the two data sets by using a variable that is contained in both data sets. The new combined data set becomes the DATA= value in the PROC GMAP statement. When the response data set and the feature table are merged into one, do not use MAP=*map-data-set* in the PROC GMAP statement. The $GEOREF formatted variable is the ID variable for the combined data set. See Example 11 on page 1314 for more details.

Note: Response data that does not correspond to a map feature is included in the legend. △

About Response Variables

The GMAP procedure can produce block, choropleth, prism, and surface maps for both numeric and character response variables. Numeric variables fall into two categories: discrete and continuous.

- *Discrete variables* contain a finite number of specific numeric values that are to be represented on the map. For example, a variable that contains only the values 1989 or 1990 is a discrete variable.
- *Continuous variables* contain a range of numeric values that are to be represented on the map. For example, a variable that contains any real value between 0 and 100 is a continuous variable.

Numeric response variables are treated as continuous variables unless the DISCRETE option is used in the action statement.

About Response Levels

Response levels are the values that identify categories of data on the graph. The categories that are shown on the graph are based on the values of the response variable. Based on the type of the response variable, a response level can be determined by any of the following:

- a character value
- the MIDPOINTS= option
- a range of numeric values
- a specific numeric value

When response levels are determined by a character value, the GMAP procedure treats each unique value as a response level. For example, if the response variable contains the names of ten regions, each region is a response level, resulting in ten response levels.

When character response levels are determined by the MIDPOINTS= option, any response variable values that do not match one of the specified response level values are ignored.

When response levels are determined by a range of numeric values, each response level has a similar number of observations. These options are exceptions to this:

- The LEVELS= option specifies the number of response levels to be used on the map.
- The DISCRETE option causes the numeric variable to be treated as a discrete variable.
- The MIDPOINTS= option chooses specific response level values as medians of the value ranges.

If the response variable values are continuous, then the GMAP procedure assigns response level intervals automatically unless you specify otherwise. The response levels represent a range of values rather than a single value.

When response levels are determined by specific numeric values, and the DISCRETE option is specified, one level is created for each value. If the response variable has an associated format, then each formatted value is represented by a different response level.

The AREA, BLOCK, CHORO, and PRISM statements assign patterns to response levels. In CHORO and PRISM maps, response levels are shown as map areas.

However, in BLOCK maps, response levels are shown as blocks. If you specify the AREA statement on a BLOCK map, then the response levels for AREA variable are shown as map areas. The default fill pattern for the response level is solid.

PATTERN statements can define the fill patterns and colors for both blocks and map areas. PATTERN definitions that define valid block patterns are applied to the blocks (response levels), and PATTERN definitions that define valid map patterns are applied to map areas.

See "PATTERN Statement" on page 240 for more information on fill pattern values and default pattern rotation.

About Identification Variables

For traditional map data sets and response data sets, *id-variables* identify the map areas (for example, counties, states, or provinces) that make up the map. A *unit area* or *map area* is a group of observations with the same ID value. The GMAP procedure matches the value of the response variables for each map area in the response data set to the corresponding map area in the traditional map data set in order to create the output graphs.

With feature tables, the *geo-variable*, or $GEOREF formatted variable containing the spatial information, is the identification variable. Each observation in a feature table has a unique $GEOREF formatted variable value. When merging the feature table with the response data set using an SQL or DATA step statement, the identification variable can be any variable that is contained within both data sets. Once the merged data set has been created, the *geo-variable* is used in the PROC GMAP ID statement for the merged feature table and response data set. See Example 11 on page 1314 for more details.

Displaying Map Areas and Response Data

Whether the GMAP procedure draws a map area and whether it displays patterns for response values depends on the contents of the response data set and on the ALL and MISSING options. The following table describes the conditions under which the procedure does or does not display map areas and response data.

Table 43.2 Displaying Map Areas and Response Data

If the response data set . . .	*And if . . .*	*Then the procedure . . .*
includes the map area	the map area has a response value	draws the map area and displays the response data
includes the map area	the response value for the map area is a missing value	draws the map area but leaves it empty
includes the map area	the response value for the map area is a missing value and the MISSING option is used in the map statement	draws the map area and displays a response level for the missing value
does not include the map area	the ALL option is used in the PROC GMAP statement	draws the map area but leaves it empty
does not include the map area	the ALL option is not used	does not draw the map area

Summary of Use

To use the GMAP procedure, you must do the following:

1. If using a traditional map data set, determine what processing needs to be done to the map data set before it is displayed. Use the GPROJECT, GREDUCE, and GREMOVE procedures or a DATA step to perform the necessary processing.
2. Issue a LIBNAME statement for the SAS data set that contains the response data set, or use a DATA step to create a response data set.
3. If using a traditional map data set, use the PROC GMAP statement to identify the map data set as the MAP= value and response data set as the DATA= value.
4. If using a feature table, use PROC SORT to individually sort the feature table and response data set by a variable common to both data sets. Next, use SQL or the DATA step MERGE to merge the feature table with the response data set by using a variable common to both data sets. Use the combined data set as the DATA= value in the PROC GMAP statement (do not include MAP= in the PROC GMAP statement).
5. Use the ID statement to specify the *id-variables* or, if you are using a feature table, specify the *geo-variable*.
6. Use a BLOCK, CHORO, PRISM, or SURFACE statement to identify the response variable and generate the map.

Accessing SAS Maps Online

Visit SAS Maps Online to download data updates, sample SAS/GRAPH programs that use the map data sets delivered with SAS/GRAPH, and GIF images of maps. SAS Maps Online is located at the following URL:

`http://www.sas.com/mapsonline`

After downloading and unzipping map data sets, you must take them out of transport format by running the CIMPORT procedure using your current version of SAS. For more information, see Appendix 4, "Transporting and Converting Graphics Output," on page 1659.

Importing Maps from ESRI Shapefiles

You can import ESRI shapefiles as traditional map data sets by using the MAPIMPORT procedure. Depending on the type of coordinates that are in your shapefile, you might want to perform additional processing. For example, you might want to project the map with the GPROJECT procedure, or use the GREDUCE procedure to create a DENSITY variable for reducing your data.

For more information, see Chapter 55, "The MAPIMPORT Procedure," on page 1593.

Procedure Syntax

Requirements: One ID statement, and at least one CHORO, BLOCK, PRISM, or SURFACE statement is required.

Global statements: FOOTNOTE, LEGEND, PATTERN, and TITLE

Reminder: The GMAP procedure can include the BY, FORMAT, LABEL, and WHERE statements as well as the TITLE, NOTE, and FOOTNOTE statements.

Supports: RUN-group processing

PROC GMAP <MAP=*map-data-set*>
DATA=*response-data-set* | *feature-table*
<ALL>
<ANNOTATE=*Annotate-data-set*>
<DENSITY=0...6 | LOW | MEDIUM | HIGH>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>
<STRETCH>
<UNIFORM>;
ID *id-variable(s)* | *geo-variable*;
AREA *response-variable* </ *option(s)*>;
BLOCK *response-variable(s)* </ *option(s)*>;
CHORO *response-variable(s)* </ *option(s)*>;
PRISM *response-variable(s)*</ *option(s)*>;
SURFACE *response-variable(s)* </ *option(s)*>;

PROC GMAP Statement

Identifies the map data set and the response data set that contain the variables associated with the map. If the response data set and the feature table have been merged, the statement's DATA= option identifies the merged data set. The statement also provides the option to display all map areas and to specify annotation and an output catalog.

Requirements: Both a map data set and a response data set are required. This can include a traditional map data set and response data set or a merged response data set and feature table.

PROC GMAP <MAP=*map-data-set*>
DATA=*response-data-set* | *feature-table*
<ALL>
<ANNOTATE=*Annotate-data-set*>
<DENSITY=0...6 | LOW | MEDIUM | HIGH>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>
<STRETCH>
<UNIFORM>;

Required Argument

DATA=*response-data-set* | *feature-table*
identifies the SAS data set that contains the response values or the response values and the spatial information that are evaluated and represented on the map. If a response data set is specified, it must contain the same identification variable or variables as the map data set, along with the values of the response variable. If a feature table is specified, it must contain response data information and spatial geometry information. By default, the GMAP procedure uses the most recently created SAS data set.

See Also: "Concepts" on page 1244, "SAS Data Sets" on page 54, and "About Feature Tables" on page 1246.

Options

PROC GMAP statement options affect all of the graphs that are produced by the procedure.

ALL
specifies that the maps generated by the procedure should include all of the map areas from the map data set, even if the response data set does not include an observation for the map area.

When you use the ALL option and a BY statement in a RUN group, the maps generated for each BY group include every map area from the map data set.

See also: "Displaying Map Areas and Response Data" on page 1250.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate all of the maps that are produced by the GMAP procedure. To annotate individual maps, use the ANNOTATE= option in the action statement.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

DENSITY=0...6 | LOW | MEDIUM | HIGH
for maps that have a DENSITY variable, specifies the density of map observations that are used. The value that you specify indicates the maximum value that the DENSITY variable can have for the observation to be displayed. For example, if you specify DENSITY=5, then only observations in the map data set whose DENSITY value is less than or equal to 5 are displayed.

Intuitively, the DENSITY variable specifies how close a map point is to other map points. If there are many map points in close proximity (high density), then it is possible to eliminate a number of them without seriously degrading the quality of the map. Many map data sets supplied by SAS contain a DENSITY variable. For map data sets that do not contain a DENSITY variable, you can add and populate the variable using the GREDUCE procedure.

You can specify an integer from 0 to 6 for the DENSITY option, or you can specify one of the following: LOW = 1, MEDIUM = 3, HIGH = 6.

If you do not specify the DENSITY option, then all the observations in a map data set are displayed, regardless of whether the data set contains a DENSITY variable or not. This is equivalent to specifying DENSITY=6.

Alias: RESOLUTION=, RES=

Restriction: If the map data set does not contain a column of DENSITY values, then a warning is issued and the option is ignored.

See also: Chapter 48, "The GREDUCE Procedure," on page 1447 for information on the DENSITY variable

GOUT=<*libref.*>*output-catalog*
specifies the SAS catalog in which to save the graphics output that is produced by the GMAP procedure for later replay. You can use the GREPLAY procedure to view the graphs stored in the catalog. If you do not use the GOUT= option, catalog entries are written to the default catalog WORK.GSEG, which is erased at the end of your session.

Restriction: Not supported by Java and ActiveX

See also: "Specifying the Catalog Name and Entry Name for Your GRSEGs" on page 100

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that contains information about the graph that is replayed from the graphics catalog. The information in the image map data set

includes the shape and coordinates of the elements in the graph, along with values that were associated with those elements in variables that were identified for that purpose in the HTML= or HTML_LEGEND= options. The image map data set can be used to generate an HTML image map in an HTML output file using the IMAGEMAP macro. The IMAGEMAP macro takes two arguments, the name of the image map data set and the name or fileref of an HTML output file, as shown in the following example:

```
%imagemap(imgmapds, myimgmap.html);
```

Restriction: Not supported by Java and ActiveX.

MAP=*map-data-set*
names a SAS traditional map data set that contains the X and Y coordinates for the boundary points of each map area. The traditional map data set must contain the same identification variable or variables as the response data set being used. This statement is required if a feature table is not being used.

See also: "About Traditional Data Sets" on page 1244.

STRETCH
stretches map extents to cover all available space in the device. This might cause the map to be distorted. When this option is applied to the PROC GMAP statement, it applies to all statements. If applied to a single statement, it applies only to that statement.

Restriction: Not supported by Java and ActiveX.

UNIFORM
causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option pre-scans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in a static legend and color distribution across all maps such that a single value always has the same color in multiple maps.

When specified on a PROC GMAP statement, UNIFORM applies to all BLOCK, CHORO, AREA, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual BLOCK, AREA, CHORO, or PRISM statement, UNIFORM applies only to the maps produced by that statement.

Restriction: Not supported by Java.

ID Statement

Identifies the variable or variables in the input data set(s) that define map areas.

Requirements: At least one *id-variable* or *geo-variable* is required.

ID *id-variable(s)* | *geo-variable*;

Required Arguments

id-variable(s)
identifies one or more variables in the map and response data sets that define map area. This argument is used only when map and response data sets are specified. If a feature table is specified, then specify a *geo-variable* instead.

Every variable that is listed in the ID statement must appear in both the map and response data sets. The variable identified by the *id-variable(s)* argument can be of type numeric or character and should have the same name, type, and length in both the response and map data sets.

Note: If the ID variables in the response data set and map data set do not have the same length, then your map areas might not be drawn correctly. △

Featured in: Example 1 on page 1301, Example 3 on page 1304, and Example 5 on page 1307

See also: "About Identification Variables" on page 1250

geo-variable
identifies the $GEOREF formatted variable in the feature table containing the spatial geometry information for the map. The variable identified by the *geo-variable* argument must be of character type.

Featured in: Example 11 on page 1314

See also: "About Identification Variables" on page 1250

AREA Statement

Applies color to the regions in BLOCK and PRISM maps based on values of a specified response variable.

Requirements: The response variable is required. The AREA statement must be used in conjunction with either a BLOCK or PRISM statement.

Description

In the case of BLOCK: whereas the BLOCK statement controls the color and appearance of the blocks, the AREA statement controls the color and appearance of the regions under the block.

In the case of PRISM: whereas the PRISM statement controls the height of the prism, the AREA statement controls the color of the region. If you specify an AREA statement, the PRISM statement controls both the color and height.

AREA *response-variable* </ *option(s)*>;

The *option(s)* argument can be one or more of the following:

DISCRETE

LEGEND=*LEGEND<1...99>*

LEVELS=*number-of-response-levels* | ALL

MIDPOINTS=*value-list* | OLD

MISSING

NOLEGEND

PERCENT

RANGE

STATISTIC=FIRST | SUM | FREQUENCY | MEAN

STATFMT= *format-specification*

UNIFORM

Required Arguments

response-variable
specifies the variable in the response data set or in the merged response and feature table if they contain response values that are to be represented on the map. Areas that correspond to response variables with missing values are not colored unless you use the MISSING option in the AREA statement. This variable is represented in all BLOCK and PRISM maps in the same RUN group.

See also: "About Response Variables" on page 1249.

Options

Options in an AREA statement affect all of the maps that are produced by that statement. You can specify as many options as you want and list them in any order.

All of these options are the same as the normal GMAP options except that they apply to the areas of regions only, and not to the bar heights. Here is an example:

```
BLOCK ELECT / LEVELS=5;
AREA CANDIDATE / DISCRETE NOLEGEND;
```

This produces a block map where there are five levels categorized for the blocks. Regions under the blocks are colored by DISCRETE CANDIDATE. No legend is shown for the CANDIDATE values, but a legend is shown for the ELECT values.

DISCRETE
generates a separate response level (color and surface pattern) for each different value of the formatted response variable. The LEVELS= option is ignored when you use the DISCRETE option.

If you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values. If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme.

Note: If the data does not contain a value in a particular range of the format, that formatted range is not displayed in the legend. △

LEGEND=LEGEND<1...99>
specifies the LEGEND statement to associate with the map. The LEGEND= option is ignored if the specified LEGEND definition is not currently in effect. In the GMAP procedure, the BLOCK statement produces a legend unless you use the NOLEGEND option. If you use the SHAPE= option in a LEGEND statement, only the value BAR is valid. Most of the LEGEND options described in "LEGEND Statement" on page 225 are supported by both Java and ActiveX. If a LEGEND option is not supported by Java or ActiveX, it is noted in the LEGEND option definition.

Restriction: Partially supported by Java and ActiveX

See also: "LEGEND Statement" on page 225

LEVELS=*number-of-response-levels* | ALL
specifies the number of response levels to be graphed when the response variables are numeric and the DISCRETE and MIDPOINTS= options are not specified. Each response level is assigned a different surface pattern and color combination. The prism and block heights are based on the data value of the corresponding response variable.

If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme. The response values are assigned lighter and darker values of a color scheme to express lower and higher response values. If you specify the DISCRETE option, then distinct, non-continuous colors are used are used for the response values.

If neither the LEVELS= option nor the DISCRETE option is used, then the GMAP procedure determines the number of response levels by using the formula FLOOR(1+3.3 log(n)), where n is the number of response variable values.

By default, an equal-distribution (quantizing) algorithm is used to determine each level.

The LEVELS= option is ignored when you use the DISCRETE or MIDPOINTS=*value-list* option. When MIDPOINTS=OLD is used with the LEVELS= option, default midpoints are generated using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

MIDPOINTS=*value-list* | OLD
specifies the response levels for the range of response values that are represented by each level (pattern and color combination).

For numeric response variables, *value-list* is either an explicit list of values or a starting 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*>>

By default, the increment value is 1. You can specify discrete numeric values in any order. In all forms, *n* can be separated by blanks or commas. For example,

```
midpoints=(2 4 6)
midpoints=(2,4,6)
midpoints=(2 to 10 by 2)
```

If a numeric variable has an associated format, the specified values must be the *unformatted* values. With numeric response values, DEVICE=JAVA uses only midpoints that fall in the range of the data being used. Thus, if your data ranged from 30–80, but midpoints were specified at 25, 50, 75, and 100, only 50 and 75 are used.

For character response variables, *value-list* is a list of unique character values enclosed in quotes and separated by blanks:

'*value-1*' <...'*value-n*'>

```
midpoints="Midwest" "Northeast" "Northwest"
```

Specify the values in any order. If a character variable has an associated format, the specified values must be the *formatted* values. Character response values specified with the MIDPOINTS= option are not supported by DEVICE=JAVA.

You can selectively exclude some response variable values from the map, as shown here:

```
midpoints="Midwest"
```

Only those observations for which the response variable exactly matches one of the values listed in the MIDPOINTS= option are shown on the map. As a result,

observations might be excluded inadvertently if values in the list are misspelled or if the case does not match exactly.

Specifying MIDPOINTS=OLD generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Restriction: Partially supported by Java

See also: The RANGE option

MISSING
accepts a missing value as a valid level for the response variable.

NOLEGEND
suppresses the legend for the areas.

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTICS= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or MEAN. If you do not use the STATISTICS= option, then STATISTICS=FIRST is assumed and the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias: PERCENTAGE

See also: The STATFMT= option on page 1258, and the STATISTIC= option on page 1258

RANGE
causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints). For example, if MIDPOINTS=15 25 35, then the legend could show 10-20, 20-30, 30-40.

Restriction The MIDPOINTS= option must be specified for the RANGE option to have any effect.

Not supported by ActiveX.

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias: SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart. For nonnumeric variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST	GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.
SUM	All observations matching a given ID value are added together and the summed value is charted.
FREQUENCY	A count of all rows with non-missing values is charted unless you specify the MISSING option.
MEAN	All observations matching a given ID value are added together and then divided by the number of non-missing observations

matched. This value is then charted unless you specify the MISSING option.

Alias: STAT=

UNIFORM
causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option prescans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in a static legend and color distribution across all maps such that a single value always has the same color in multiple maps.

When specified on a PROC GMAP statement, the UNIFORM option applies to all AREA, BLOCK, CHORO, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual AREA, BLOCK, CHORO, or PRISM statement, the UNIFORM option applies only to the maps produced by that statement.

Restriction: Not supported by Java.

BLOCK Statement

Creates three-dimensional block maps on which levels of magnitude of the specified response variables are represented by blocks (bars) of varying height, pattern, and color.

Requirements: At least one response variable is required. The ID statement must be used in conjunction with the BLOCK statement.

Global statements: FOOTNOTE, LEGEND, PATTERN, TITLE

Description

The BLOCK statement specifies the variable or variables that contain the data that are represented on the map by blocks of varying height, pattern, and color. This statement automatically performs the following operations:

- determines the midpoints ranges.
- scales the blocks.
- assigns patterns to the block faces and map areas. (See "About Block Maps and Patterns" on page 1268 for more information.)

You can use statement options to enhance the appearance of the map. For example, you can specify the width and shape of the blocks, the outline colors for the blocks and the map areas, and the angle of view. Other statement options control the response levels.

In addition, you can use global statements to modify the block patterns, the map patterns, and the legend, as well as to add titles and footnotes to the map. You can also use an Annotate data set to enhance the map.

BLOCK *response-variable(s)* </ *option(s)*>;

The *option(s)* argument can be one or more of the following:

- appearance options:

  ANNOTATE=*Annotate-data-set*

BLOCKSIZE=*size*
CBLKOUT=*block-outline-color* | SAME
CDEFAULT=*empty-area-fill-color*
CEMPTY=*empty-area-outline-color*
COUTLINE=*area-outline-color* | SAME
SHAPE=*3D-block-shape*
STRETCH
UNIFORM
WOUTLINE=*block-outline-width*
XSIZE=*map-width <units>*
YSIZE=*map-height <units>*
XVIEW=*x*
YVIEW=*y*
ZVIEW=*z*

- mapping options:
  AREA=*n* | *column-name*
  DISCRETE
  LEVELS=*number-of-response-levels* | ALL
  MIDPOINTS=*value-list* | OLD
  MISSING
  PERCENT | PERCENTAGE
  RANGE
  RELZERO
  STATISTIC=FIRST | SUM | FREQUENCY | MEAN
  STATFMT=*format-specification*
- legend options:
  CTEXT=*text-color*
  LEGEND=LEGEND<1...99>
  NOLEGEND
- description options:
  DESCRIPTION='*description*'
  NAME='*name*'
- ODS options
  HTML=*variable*
  HTML_LEGEND=*variable*

Required Arguments

response-variable(s)
specifies one or more variables in the response data set, or in the merged response and feature table, that contain response values that are to be represented on the map. Each response variable produces a separate map. All variables must be in the input data set. Multiple response variables are separated with blanks. Blocks are not drawn for the response variable with missing values unless you use the MISSING option in the BLOCK statement.

See also: "About Response Variables" on page 1249.

Options

Options in a BLOCK statement affect all of the maps 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 onto maps that are produced by the BLOCK statement. Annotate coordinate systems 1, 2, 7, and 8 are not valid with block maps.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641.

AREA=*n* | *column-name*
specifies that a different map pattern be used for the surface of each map area or group of map areas on the map.

You can specify pattern fills or colors or both with PATTERN statements that specify map/plot patterns. A separate PATTERN definition is needed for each specified area.

AREA=*n*	The value of *n* indicates which variable in the ID statement determines the groups that are distinguished by a surface pattern. By default, all map unit areas are drawn using the same surface fill pattern. If your ID statement has only one map area identification variable, then use AREA=1 to indicate that each map area surface uses a different pattern. If you have more than one variable in your ID statement, then use *n* to indicate the position of the variable that defines groups that share a pattern. When you use the AREA= option, the map data set should be sorted in order of the variables in the ID statement.
AREA=*column-name*	A column name defined in either the MAP= or DATA= data sets might be indicated with the *column-name* value. If the column name exists in both the MAP= and DATA= data sets, the column in the MAP= data set is used. When *column-name* is used, the areas are colored based on the AREA= value. Duplicate AREA= values might have different patterns assigned.

See also: "AREA Statement" on page 1255, "PATTERN Statement" on page 240.

BLOCKSIZE=*size*
specifies the width of the blocks. The unit of *size* is the character cell width for the selected output device. By default, BLOCKSIZE=2.

Alias: BS=

CBLKOUT=*block-outline-color* | SAME
outlines all blocks in the specified color. The SAME value specifies that the outline color of a block, a block segment, or a legend is the same as the interior pattern color.

The default outline color is determined by the current style. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

The CBLKOUT= option is not valid when SHAPE=CYLINDER.

Note: If you specify empty block patterns (VALUE=EMPTY in a PATTERN statement), you should not change the outline color from the default value, SAME, to a single color. Otherwise all the outlines are one color and you can distinguish between empty areas only by their size. Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA. △

Alias: CBLOCK=

Style reference: The Color attribute of the GraphOutlines style element

Restriction: Partially supported by Java

CDEFAULT=*empty-area-fill-color*
fills empty map areas in the specified color. This option affects only map areas that are empty. Empty map areas are generated in block maps only when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default is NONE, which draws the polygon empty, showing the background in the fill area of the polygon.

Alias: CDEF=, DEFCLR=

Restriction: Not supported by Java

See also: The CEMPTY option, the ALL on page 1253 option, and "Displaying Map Areas and Response Data" on page 1250

CEMPTY=*empty-area-outline-color*
outlines empty map areas in the specified color. This option affects only map areas that are empty. Empty map areas are generated in block maps only when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default outline color is the same as the default COUTLINE= color.

Alias: CE=

Restriction: Not supported by Java

See also: The ALL option on page 1253 and "Displaying Map Areas and Response Data" on page 1250

COUTLINE=*area-outline-color* | SAME
outlines non-empty map areas in the specified color. When COUTLINE=*area-outline-color* and DEVICE=JAVA or ACTIVEX, both empty and nonempty map areas are outlined. The SAME value specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color is determined by the current style. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

Note: If you specify empty map patterns (VALUE=EMPTY in a PATTERN statement), then you should not change the outline color from the default value SAME. Otherwise all the outlines are one color and you cannot distinguish between the empty areas. Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA. △

Alias: CO=

Style reference: The Color attribute of the GraphOutlines style element

Restriction: Partially supported by Java

CTEXT=*text-color*
specifies a color for the text in the legend. If you omit the CTEXT= option, a color specification is searched for in this order:

1. the CTEXT= option in a GOPTIONS statement.
2. the default, the text color that is specified in the current style.
3. if you specify NOGSTYLE, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

The CTEXT= color specification is overridden if you also use the COLOR= suboption of a LABEL= or VALUE= option in a LEGEND definition that is assigned

to the map legend. The COLOR= suboption determines the color of the legend label or the color of the legend value descriptions, respectively.

Alias: CT=

Style reference: The Color attribute of the GraphValueText style element

DESCRIPTION='*description*'
specifies a descriptive string up to 256 characters long, that appears in the description field of the catalog entry for the map. The description does not appear on the map. By default, the GMAP procedure assigns a description of the form BLOCK MAP OF *variable*, where *variable* is the name of the map variable.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window
- the catalog-entry properties that you can view from the Explorer window
- the Table of Contents that is generated when you use CONTENTS= on an ODS HTML statement, assuming that the procedure output is generated while the contents page is open
- the Description field of the PROC GREPLAY window
- the chart description for Web output (depending on the device driver). For more information, see "PROC GANNO Statement" on page 914.

Alias: DES=

DISCRETE
generates a separate response level (color and surface pattern) for each different value of the formatted response variable. The LEVELS= option is ignored when you use the DISCRETE option.

If you specify the DISCRETE option, then distinct, non-continuous colors are used are used for the response values. If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme.

Note: If the data does not contain a value in a particular range of the format, that formatted range is not displayed in the legend. △

HTML=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with an area of the map and point to the data or graph you want to display when the user drills down on the area.

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with a legend value and point to the data or graph you want to display in response to drill-down input from the user.

Restriction: Not supported by Java and ActiveX

LEGEND=LEGEND<1...99>
specifies the LEGEND statement to associate with the map. The LEGEND= option is ignored if the specified LEGEND definition is not currently in effect. In the GMAP procedure, the BLOCK statement produces a legend unless you use the NOLEGEND option. If you use the SHAPE= option in a LEGEND statement, only the value BAR is valid. Most of the LEGEND options described in "LEGEND Statement" on page 225 are supported by both Java and ActiveX. If a LEGEND option is not supported by Java or ActiveX, it is noted in the LEGEND option definition.

Restriction: Partially supported by Java and ActiveX

See also: "LEGEND Statement" on page 225

LEVELS=*number-of-response-levels* | ALL
specifies the number of response levels to be graphed when the response variables are numeric and the DISCRETE and MIDPOINTS= options are not specified. Each response level is assigned a different surface pattern and color combination. The block height is based on the data value of the corresponding response variable.

If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme. The response values are assigned lighter and darker values of a color scheme to express lower and higher response values. If you specify the DISCRETE option, then distinct, non-continuous colors are used are used for the response values.

Note: If you specified the NOGSTYLE system option, then non-continuous colors are used by default. △

If neither the LEVELS= option nor the DISCRETE option is used, then the GMAP procedure determines the number of response levels by using the formula FLOOR(1+3.3 log(*n*)), where *n* is the number of response variable values.

By default, an equal-distribution (quantizing) algorithm is used to determine each level.

The LEVELS= option is ignored when you use the DISCRETE or MIDPOINTS=*value-list* option. When MIDPOINTS=OLD is used with the LEVELS= option, default midpoints are generated using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

MIDPOINTS=*value-list* | OLD
specifies the response levels for the range of response values that are represented by each level (pattern and color combination).

For numeric response variables, *value-list* is either an explicit list of values or a starting 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*>>

By default, the increment value is 1. You can specify discrete numeric values in any order. In all forms, *n* can be separated by blanks or commas. For example:

```
midpoints=(2 4 6)
midpoints=(2,4,6)
midpoints=(2 to 10 by 2)
```

If a numeric variable has an associated format, the specified values must be the *unformatted* values. With numeric response values, DEVICE=JAVA uses only midpoints that fall in the range of the data being used. Thus, if your data ranged from 30–80, but midpoints were specified at 25, 50, 75, and 100, only 50 and 75 are used.

For character response variables, *value-list* is a list of unique character values enclosed in quotes and separated by blanks:

'*value-1*' <...'*value-n*'>

```
midpoints="Midwest" "Northeast" "Northwest"
```

Specify the values in any order. If a character variable has an associated format, the specified values must be the *formatted* values. Character response values specified with the MIDPOINTS= option are not supported by DEVICE=JAVA.

You can selectively exclude some response variable values from the map, as shown here:

```
midpoints="Midwest"
```

Only those observations for which the response variable exactly matches one of the values listed in the MIDPOINTS= option are shown on the map. As a result, observations might be excluded inadvertently if values in the list are misspelled or if the case does not match exactly.

Specifying MIDPOINTS=OLD generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Featured in: Example 8 on page 1311

Restriction: Partially supported by Java

See also: The RANGE option

MISSING
accepts a missing value as a valid level for the response variable.

See also: "Displaying Map Areas and Response Data" on page 1250.

NAME='*name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default GRSEG name is GMAP. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, GMAP1.

See also: "About Filename Indexing" on page 99

NOLEGEND
suppresses the legend.

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTIC= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or MEAN. If you do not use the STATISTIC= option, then STATISTIC=FIRST is assumed and the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias: PERCENTAGE

See also: The STATFMT= option on page 1266, and the STATISTIC= option on page 1266

RANGE
causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints). For example, if MIDPOINTS=15 25 35, then the legend could show 10-20, 20-30, 30-40.

Restriction MIDPOINTS= must be specified for the RANGE option to have any effect. Not supported by ActiveX.

RELZERO
creates bars and regions that are relative to a zero value. By default, GMAP creates heights that are relative to the minimum value, which might or might not be zero. With the RELZERO option, zero value bars have no height.

Alias: REL0, RELATIVETOZERO

Restriction This option works only for variables that have no negative values.

SHAPE=*3D-block-shape*
specifies the shape of the blocks. Use this option to enhance the look of the block shape, or to specify a different shape. Unless you specify SHAPE=OLD, only solid fill patterns are used. The value of *3D-block-shape* can be one of the following:

- BLOCK | B
- CYLINDER | C
- HEXAGON | H
- OLDBLOCK | OLD
- PRISM | P
- STAR | S

SHAPE=BLOCK is the default. OLDBLOCK is the same as BLOCK except that with OLDBLOCK the tops and sides of blocks are colored the same as the background, as was the case before SAS 9.2.

The CBLKOUT= option is not valid when SHAPE=CYLINDER.

Default: BLOCK

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias: SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart. For character variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST	GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.
SUM	All observations matching a given ID value are added together and the summed value is charted.
FREQUENCY	A count of all rows with nonmissing values is charted unless you specify the MISSING option.
MEAN	All observations matching a given ID value are added together and then divided by the number of nonmissing observations matched. This value is then charted unless you specify the MISSING option.

Alias: STAT=

Featured in: Example 4 on page 1306

STRETCH
stretches map extents to cover all available space in the device. This might cause the map to be distorted. When this option is applied to the PROC GMAP statement, it applies to all statements. If applied to a single statement, it applies only to that statement.

Alias: STRETCHTOFIT, STR2FIT

Restriction: Not supported by Java and ActiveX

UNIFORM
causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option prescans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in a static legend and color distribution across all maps such that a single value always has the same color in multiple maps.

When specified on a PROC GMAP statement, the UNIFORM option applies to all AREA, BLOCK, CHORO, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual AREA, BLOCK, CHORO, or PRISM statement, the UNIFORM option applies only to the maps produced by that statement.

Restriction: Not supported by Java.

WOUTLINE=*block-outline-width*
specifies the width, in pixels, of the outline for all outlined blocks and for the outline of the map areas.

Default: 1

XSIZE=*map-width <units>*
YSIZE=*map-height <units>*
specify the physical dimensions of the map to be drawn. By default, the map uses the entire procedure output area.

Valid *units* are CELLS (character cells), CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). The default unit is CELLS.

If you specify values for *map-width* or *map-height* that are greater than the dimensions of the procedure output area, the map is drawn using the default size.

Restriction: Not supported by Java and ActiveX

XVIEW=*x*
YVIEW=*y*
ZVIEW=*z*
specify coordinates of the viewing position in the reference coordinate system. In this system, the four corners of the map lie on the X-Y plane at coordinates (0,0,0), (0,1,0), (1,1,0), and (1,0,0). No axes are actually drawn on the maps that are produced by PROC GMAP. Your viewing position cannot coincide with the viewing reference point at coordinates (0.5,0.5,0), the center of the map. The value for *z* cannot be negative.

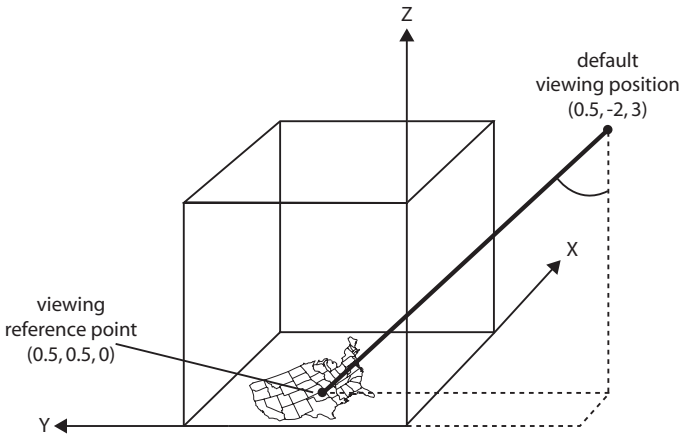
If you omit the XVIEW=, YVIEW=, and ZVIEW= options, the default coordinates are (0.5, –2, 3). This viewing position is well above and to the south of the center of the map. You can specify one, two, or all three of the view coordinates; any that you do not specify are assigned the default values. While you can use the XVIEW= and YVIEW= options with DEVICE=JAVA, ZVIEW= cannot be used with DEVICE=JAVA.

Alias: XV=, YV=, ZV=

Restriction: Partially supported by Java

Figure 43.5 on page 1268 shows the position of the viewing reference point, as well as the default viewing position.

Figure 43.5 Viewing Position and Viewing Reference Point



About Block Maps and Patterns

Block maps are different from other maps in that they display two different types of areas that use patterns:

- the blocks themselves, which represent the response levels
- the map areas from which the blocks rise

By default, block patterns are determined by the current style. If you specify the AREA statement or the AREA= option, then the map area colors are determined by the current style and the block colors are determined by the attributes that you specified.

Note: If you specified the NOGSTYLE system option, then solid patterns are used for blocks and hatch patterns are used for the map areas. The map areas and their outlines use the first color in the color list. △

The BLOCK statement has the following options that explicitly control the outline colors used by the blocks and the map areas:

- CBLKOUT=
- CEMPTY=
- COUTLINE=

In addition the AREA= option and AREA statement control how the map areas are patterned.

When you use PATTERN statements to define the patterns for the map, you must specify the correct type of pattern for the area. The blocks use bar/block patterns and the map areas use map/plot patterns. See “PATTERN Statement” on page 240 for more information on specifying patterns.

Note: If you specify only one PATTERN statement and include only the COLOR= option, that color is used for both the blocks and the map areas. For example, this statement makes the blocks solid blue and the map areas blue hatch. △

```
pattern1 color=blue;
```

Note: Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA. △

CHORO Statement

Creates two-dimensional maps in which values of the specified response variables are represented by varying patterns and colors.

Requirements: At least one response variable is required. The ID statement must be used in conjunction with the CHORO statement

Global statements: FOOTNOTE, LEGEND, PATTERN, TITLE

Description

The CHORO statement specifies the variable or variables that contain the data represented on the map by patterns that fill the map areas. This statement automatically

- determines the midpoints
- assigns patterns to the map areas

You can use statement options to enhance the appearance of the map, for example, by selecting the colors and patterns that fill the map areas. Other statement options control the selection of ranges for the response variable.

In addition, you can use global statements to modify the map area patterns and legend, as well as add titles and footnotes to the map. You can also use an Annotate data set to enhance the map.

CHORO *response-variable(s)* </ *option(s)*>;

option(s) can be one or more from any of the following categories:

- appearance options:
  - ANNOTATE=*Annotate-data-set*
  - CDEFAULT=*empty-area-fill-color*
  - CEMPTY=*empty-area-outline-color*
  - COUTLINE=*area-outline-color* | SAME
  - STRETCH
  - UNIFORM
  - WOUTLINE=*area-outline-width*
  - XSIZE=*map-width<units>*
  - YSIZE=*map-height <units>*
- mapping options:
  - DISCRETE
  - LEVELS=*number-of-response-levels* | ALL
  - MIDPOINTS=*value-list* | OLD
  - MISSING
  - PERCENT | PERCENTAGE
  - RANGE
  - STATISTIC=FIRST | SUM | FREQUENCY | MEAN
  - STATFMT=*format-specification*
- legend options:
  - CTEXT=*text-color*

LEGEND=LEGEND<1...99>

NOLEGEND

- description options:

  DESCRIPTION='*description*'

  NAME='*name*'

- ODS options

  HTML=*variable*

  HTML_LEGEND=*variable*

Required Arguments

response-variable(s)
specifies one or more variables in the response data set or in the merged response and feature table if they contain response values that are represented on the map. Each response variable produces a separate map. All variables must be in the input data set. Multiple response variables are separated with blanks.

Missing values for the response variable are not considered valid response values unless you use the MISSING option in the CHORO statement.

Response variables can be either numeric or character in type. Numeric response variables are normally grouped into ranges, or response levels, as determined by the MIDPOINTS= or LEVELS= options. Each response level is assigned a different combination of pattern and color. Character response variables are assigned unique response levels, as are numeric variables when the DISCRETE option is specified.

See also: "About Response Variables" on page 1249.

Options

Options in a CHORO 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 onto maps that are produced by the CHORO statement.

Alias: ANNO=

Featured in: Example 6 on page 1308.

See also: Chapter 29, "Using Annotate Data Sets," on page 641.

CDEFAULT=*empty-area-fill-color*
fills empty map areas in the specified color. This option affects only map areas that are empty. Empty map areas are generated in choro maps only when there is no response value for a map area and the MISSING option is not used, or when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default is NONE, which draws the polygon empty, showing the background in the fill area of the polygon.

Alias: CDEF=, DEFCLR=

Restriction: Not supported by Java

See also: The CEMPTY option, the ALL option on page 1253, and "Displaying Map Areas and Response Data" on page 1250

CEMPTY=*empty-area-outline-color*
outlines empty map areas in the specified color. This option affects only the empty map areas, which are generated in choro maps when either of the following is true:

- There is no response value for a map area and the MISSING option is not used.
- A map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default outline color is the same as the default COUTLINE= color.

Alias: CE=

Restriction: Not supported by Java

See also: The ALL option on page 1253 and "Displaying Map Areas and Response Data" on page 1250

COUTLINE=*area-outline-color* | SAME
outlines non-empty map areas in the specified color. When COUTLINE=*area-outline-color* and DEVICE=JAVA or ACTIVEX, both empty and non-empty map areas are outlined. The value SAME specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color is determined by the current style. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

Note: If you specify empty map patterns (VALUE=EMPTY in a PATTERN statement), then you should not change the outline color from the default value SAME to a single color. Otherwise all the outlines are one color and you cannot distinguish between the empty areas. △

Alias: CO=

Style reference: The Color attribute of the GraphOutlines style element

CTEXT=*text-color*
specifies a color for the text in the legend. If you omit the CTEXT= option, a color specification is searched for in this order:

1. the CTEXT= option in a GOPTIONS statement.
2. the default, the text color that is specified in the current style.
3. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

The CTEXT= color specification is overridden if you also use the COLOR= suboption of a LABEL= or VALUE= option in a LEGEND definition that is assigned to the map legend. The COLOR= suboption determines the color of the legend label or the color of the legend value descriptions, respectively.

Alias: CT=

Style reference: The Color attribute of the GraphValueText style element

DESCRIPTION='*description*'
specifies a descriptive string up to 256 characters long that appears in the description field of the catalog entry for the map. The description does not appear on the map. By default, the GMAP procedure assigns a description of the form CHOROPLETH MAP OF *map_variable*.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window
- the catalog-entry properties that you can view from the Explorer window
- the Table of Contents that is generated when you use CONTENTS= on an ODS HTML statement, assuming that the procedure output is generated while the contents page is open

- the Description field of the PROC GREPLAY window
- the chart description for Web output (depending on the device driver). For more information, see "PROC GANNO Statement" on page 914.

Alias: DES=

DISCRETE
generates a separate response level (color and surface pattern) for each different value of the formatted response variable. The LEVELS= option is ignored when you use the DISCRETE option.

If you specify the DISCRETE option, then distinct, non-continuous colors are used are used for the response values. If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme.

Note: If the data does not contain a value in a particular range of the format, that formatted range is not displayed in the legend. △

Featured in: Example 11 on page 1314

HTML=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with an area of the map and point to the data or graph you want to display when you drill down on the area.

See also: "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with a legend value and point to the data or graph you want to display when you drill down on the value.

Restriction: Not supported by Java and ActiveX

See also: "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601

LEGEND=LEGEND<1...99>
assigns the specified LEGEND statement that is to be applied to the map. The LEGEND= option is ignored if the specified LEGEND definition is not currently in effect. In the GMAP procedure, the CHORO statement produces a legend by default unless you specify the NOLEGEND option. If you use the SHAPE= option in a LEGEND statement, then only the value BAR is valid. Most of the LEGEND options described in "LEGEND Statement" on page 225 are supported by both Java and ActiveX. If a LEGEND option is not supported by Java or ActiveX, it is noted in the LEGEND option definition.

Featured in: Example 3 on page 1304

Restriction: Partially supported by Java and ActiveX

See also: "LEGEND Statement" on page 225.

LEVELS=*number-of-response-levels* | ALL
specifies the number of response levels to be graphed when the response variables are numeric and the DISCRETE and MIDPOINTS= options are not specified. Each response level is assigned a different surface pattern and color combination.

If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme. The response values are assigned lighter and darker values of a color scheme to express lower and higher response values. If

you specify the DISCRETE option, then distinct, non-continuous colors are used are used for the response values.

Note: If you specified the NOGSTYLE system option, then non-continuous colors are used by default. △

If neither the LEVELS= option nor the DISCRETE option is used, then the GMAP procedure determines the number of response levels by using the formula FLOOR(1+3.3 log(*n*)), where *n* is the number of response variable values.

By default, an equal-distribution (quantizing) algorithm is used to determine each level.

The LEVELS= option is ignored when you use the DISCRETE or MIDPOINTS=*value-list* option. When MIDPOINTS=OLD is used with the LEVELS= option, default midpoints are generated using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Featured in: Example 2 on page 1302

MIDPOINTS=*value-list* | OLD

specifies the response levels for the range of response values that are represented by each level (pattern and color combination).

For numeric response variables, the *value-list* argument is either an explicit list of values, a starting 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*>

By default the increment value is 1. You can specify discrete numeric values in any order. In all forms, *n* can be separated by blanks or commas. For example:

```
midpoints=(2 4 6)
midpoints=(2,4,6)
midpoints=(2 to 10 by 2)
```

If a numeric variable has an associated format, the specified values must be the *unformatted* values. With numeric response values, DEVICE=JAVA uses only midpoints that fall in the range of the data being used. Thus, if your data ranged from 30–80, but midpoints were specified at 25, 50, 75,and 100, only 50 and 75 are used.

For character response variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:

'*value-1*' <...'*value-n*'>

The values are character strings enclosed in single quotation marks and separated by blanks. For example:

```
midpoints="Midwest" "Northeast" "Northwest"
```

Specify the values in any order. If a character variable has an associated format, the specified values must be the *formatted* values. Character response values specified with the MIDPOINTS= option are not supported by DEVICE=JAVA.

You can selectively exclude some response variable values from the map, as shown here:

```
midpoints="Midwest"
```

The only observations that are shown on the map are those observations for which the response variable exactly matches one of the values that are listed in the MIDPOINTS= option. As a result, observations might be excluded inadvertently if values in the list are misspelled or if the case does not match exactly.

Specifying MIDPOINTS=OLD generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Featured in: Example 8 on page 1311

Restriction: Partially supported by Java

See also: The RANGE option

MISSING
accepts a missing value as a valid level for the response variable.

See also: "Displaying Map Areas and Response Data" on page 1250

NAME='*name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default GRSEG name is GMAP. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, GMAP1.

See also: "About Filename Indexing" on page 99

NOLEGEND
suppresses the legend.

Featured in: Example 6 on page 1308

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTIC= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or MEAN. If you do not use the STATISTIC= option, then STATISTIC=FIRST is assumed—the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias: PERCENTAGE

See also: The STATFMT= option on page 1274 and the STATISTIC= option on page 1274.

RANGE
causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints). For example, if MIDPOINTS=15 25 35, then the legend could show 10-20, 20-30, 30-40.

Restriction: The MIDPOINTS= option must be specified for the RANGE option to have any effect.
Not supported by ActiveX.

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias: SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart. For character variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST	GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.
SUM	All observations matching a given ID value are added together and the summed value is charted.
FREQUENCY	A count of all rows with nonmissing values is charted unless you specify the MISSING option.
MEAN	All observations matching a given ID value are added together and then divided by the number of non-missing observations matched. This value is then charted unless you specify the MISSING option.

Alias: STAT=

STRETCH
stretches map extents to cover all available space in the device. This might cause the map to be distorted. When this option is applied to the PROC GMAP statement, it applies to all statements. If applied to a single statement, it applies only to that statement.

Alias: STRETCHTOFIT, STR2FIT

Restriction: Not supported by Java and ActiveX

UNIFORM
causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option prescans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in a static legend and color distribution across all maps such that a single value always has the same color in multiple maps.

When specified on a PROC GMAP statement, UNIFORM applies to all AREA, BLOCK, CHORO, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual AREA, BLOCK, CHORO, or PRISM statement, UNIFORM applies only to the maps produced by that statement.

Restriction: Not supported by Java

WOUTLINE=*area-outline-width*
specifies the width of all map area outlines, in pixels.

Default: 1

XSIZE=*map-width <units>*
YSIZE=*map-height <units>*
specify the physical dimensions of the map. By default, the map uses the entire procedure output area.

Valid *units* are CELLS (character cells), CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). The default unit is CELLS.

If you specify values for *units* that are greater than the dimensions of the procedure output area, the map is drawn using the default size.

If you specify either the XSIZE= or YSIZE= option without specifying the other option, the GMAP procedure scales the dimension for the option that was not specified to retain the original shape of the map.

Restriction: Not supported by Java and ActiveX

PRISM Statement

Creates three-dimensional prism maps in which levels of magnitude of the specified response variables are represented by polyhedrons (raised polygons) of varying height, pattern, and color.

Requirements: At least one response variable is required. You must use the ID statement in conjunction with the PRISM statement.

Global statements: FOOTNOTE, LEGEND, PATTERN, TITLE

Description

The PRISM statement specifies the variable or variables that contain the data that are represented on the map by raised map areas. This statement automatically performs the following operations:

- determines the midpoints ranges or midpoints
- assigns patterns to the map areas

You can use statement options to control the ranges of the response values, specify the angle of view, and enhance the appearance of the map.

In addition, you can use global statements to modify the map area patterns and the legend, as well as add titles and footnotes to the map. You can also use an Annotate data set to enhance the map.

Note: PRISM maps do not work well with polygons within polygons (holes). It is recommended that a CHORO or BLOCK map be created for these maps instead. △

PRISM *response-variable(s)* </ *option(s)*>;

The *option(s)* can be one or more options from any or all of the following categories:

- appearance options:

  ANNOTATE=*Annotate-data-set*
  CDEFAULT=*empty-area-fill-color*
  CEMPTY=*empty-area-outline-color*
  COUTLINE=*area-outline-color* | SAME
  STRETCH
  UNIFORM
  WOUTLINE=*area-outline-width*
  XLIGHT=*x*
  YLIGHT=*y*
  XSIZE=*map-width* <*units*>
  YSIZE=*map-height* <*units*>
  XVIEW=*x*
  YVIEW=*y*
  ZVIEW=*x*

- mapping options:

  AREA=*n* | *column-name* | (*area-options*)
  DISCRETE
  LEVELS=*number-of-response-levels* | ALL
  MIDPOINTS=*value-list* | OLD

MISSING
PERCENT | PERCENTAGE
RANGE
RELZERO
STATISTIC=FIRST | SUM | FREQUENCY | MEAN
STATFMT=*format-specification*

- legend options:
  CTEXT=*text-color*
  LEGEND=LEGEND<1...99>
  NOLEGEND
- description options:
  DESCRIPTION='*description*'
  NAME='*name*'
- ODS options
  HTML=*variable*
  HTML_LEGEND=*variable*

Required Arguments

response-variable(s)
specifies one or more variables in the response data set, or in the merged response and feature table, that contain response values that are to be represented on the map. Each response variable produces a separate map. All variables must be in the input data set. Multiple response variables are separated with blanks.

Missing values for the response variable are not considered valid unless you use the MISSING option.

Response variables can be either numeric or character. By default, and as determined by the LEVELS= or MIDPOINTS= values, numeric response variables are grouped into ranges, or response levels. Each response level is assigned a different prism height and a different pattern and color combination.

Character variables and numeric variables (when you use the DISCRETE option) have a unique response level for each unique response variable value.

See also: "About Response Variables" on page 1249.

Options

Options in a PRISM 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.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate onto the maps that are produced by the PRISM statement. Annotate coordinate systems 1, 2, 7, and 8 are not valid with Prism maps.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

AREA=*n* | *column-name*
specifies that a different map pattern be used for the surface of each map area or group of map areas on the map.

Note: The AREA statement provides a greater amount of control than the AREA= option. △

You can specify pattern fills or colors or both with PATTERN statements that specify map/plot patterns. A separate PATTERN definition is needed for each specified area.

AREA=*n* The value of *n* indicates which variable in the ID statement determines the groups that are distinguished by a surface pattern. By default, all map unit areas are drawn using the same surface fill pattern. If your ID statement has only one map area identification variable, then use AREA=1 to indicate that each map area surface uses a different pattern. If you have more than one variable in your ID statement, then use *n* to indicate the position of the variable that defines groups that share a pattern. When you use the AREA= option, the map data set should be sorted in order of the variables in the ID statement.

AREA=*column-name* A column name defined in either the MAP= or DATA= data sets can be indicated with the *column-name* value. If the column name exists in both the MAP= and DATA= data sets, the column in the map= data set is used. When *column-name* is used, the areas are colored based on the AREA= value. Duplicate AREA= values might have different patterns assigned

See also: "AREA Statement" on page 1255, "PATTERN Statement" on page 240.

CDEFAULT=*empty-area-fill-color*

fills empty map areas in the specified color. This option affects only map areas that are empty. Empty map areas are generated in prism maps only when there is no response value for a map area and the MISSING option is not used, or when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default is NONE, which draws the polygon empty, showing the background in the fill area of the polygon.

Alias: CDEF=, DEFCLR=

Restriction: Not supported by Java

See also: The CEMPTY option, the ALL option on page 1253, and "Displaying Map Areas and Response Data" on page 1250

CEMPTY=*empty-area-outline-color*

outlines empty map areas in the specified color. Empty map areas are generated in prism maps either

- when there is no response value for a map area and the MISSING option is not used, or
- when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default outline color is the same as the default COUTLINE= color.

Alias: CE=

Restriction: Not supported by Java

See also: ALL on page 1253 and "Displaying Map Areas and Response Data" on page 1250

COUTLINE=*area-outline-color* | SAME

outlines nonempty map areas in the specified color. SAME specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color is determined by the current style. If you specified the NOGSTYLE system option, then the default color is the first color in the color list.

Note: If you specify empty map patterns (VALUE=EMPTY in a PATTERN statement), you should not change the outline color from the default value SAME to a single color. Otherwise, all the outlines are one color and you cannot distinguish between the empty areas. Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA. △

Alias: CO=

Style reference: The Color attribute of the GraphOutlines style element.

CTEXT=*text-color*
specifies a color for the text in the legend. If you omit the CTEXT= option, a color specification is searched for in this order:

1. the CTEXT= option in a GOPTIONS statement.
2. the default, the text color that is specified in the current style.
3. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

The CTEXT= color specification is overridden if you also use the COLOR= suboption of a LABEL= or VALUE= option in a LEGEND definition assigned to the map legend. The COLOR= suboption determines the color of the legend label or the color of the legend value descriptions, respectively.

Alias: CT=

Style reference: The Color attribute of the GraphValueText style element

DESCRIPTION='*description*'
specifies the description of the catalog entry for the map. The maximum length for *description* is 256 characters. By default, the GMAP procedure assigns a description of the form PRISM MAP OF *map_variable*.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window
- the catalog-entry properties that you can view from the Explorer window
- the Table of Contents that is generated when you use CONTENTS= on an ODS HTML statement, assuming that the procedure output is generated while the contents page is open
- the Description field of the PROC GREPLAY window
- the chart description for Web output (depending on the device driver). For more information, see "PROC GANNO Statement" on page 914.

Alias: DES=

DISCRETE
generates a separate response level (color and surface pattern) for each different value of the formatted response variable. The LEVELS= option is ignored when you use the DISCRETE option.

If you specify the DISCRETE option, then distinct, non-continuous colors are used are used for the response values. If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme.

Note: If the data does not contain a value in a particular range of the format, that formatted range is not displayed in the legend. △

Featured in: Example 11 on page 1314

HTML=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with an area of

the map and point to the data or graph that are displayed in response to drill-down input.

See also: "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with legend values and point to the data or graphs that are displayed in response to drill-down input.

Restriction: Not supported by Java and ActiveX

See also: "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601

LEGEND=LEGEND<1...99>
specifies the LEGEND definition to associate with the map. LEGEND= is ignored if the specified LEGEND definition is not currently in effect. In the GMAP procedure, the PRISM statement produces a legend unless you use the NOLEGEND option. If you use the SHAPE= option in a LEGEND statement, only the value BAR is valid. Most of the LEGEND options described in "LEGEND Statement" on page 225 are supported by both Java and ActiveX. If a LEGEND option is not supported by Java or ActiveX, it is noted in the LEGEND option definition.

Featured in: Example 8 on page 1311

Restriction: Partially supported by Java and ActiveX

See also: "LEGEND Statement" on page 225

LEVELS=*number-of-response-levels* | ALL
specifies the number of response levels to be graphed when the response variables are numeric and the DISCRETE and MIDPOINTS= options are not specified. Each response level is assigned a different surface pattern and color combination. The prism height is based on the data value of the corresponding response variable.

If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme. The response values are assigned lighter and darker values of a color scheme to express lower and higher response values. If you specify the DISCRETE option, then distinct, non-continuous colors are used are used for the response values.

If neither the LEVELS= option nor the DISCRETE option is used, then the GMAP procedure determines the number of response levels by using the formula FLOOR(1+3.3 log(*n*)), where *n* is the number of response variable values.

By default, an equal-distribution (quantizing) algorithm is used to determine each level.

The LEVELS= option is ignored when you use the DISCRETE or MIDPOINTS=*value-list* option. When MIDPOINTS=OLD is used with the LEVELS= option, default midpoints are generated using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Featured in: Example 2 on page 1302

MIDPOINTS=*value-list* | OLD
specifies the response levels for the range of response values that are represented by each level (prism height, pattern, and color combination).

For numeric response variables, *value-list* is either an explicit list of values, or a starting 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*>

By default the increment value is 1. You can specify discrete numeric values in any order. In all forms, *n* can be separated by blanks or commas. For example,

```
midpoints=(2 4 6)
midpoints=(2,4,6)
midpoints=(2 to 10 by 2)
```

If a numeric variable has an associated format, the specified values must be the *unformatted* values. With numeric response values, DEVICE=JAVA uses only midpoints that fall in the range of the data being used. Thus, if your data ranged from 30–80, but midpoints were specified at 25, 50, 75,and 100, only 50 and 75 are used.

For character response variables, *value-list* has this form:

'*value-1*' <...'*value-n*'>

The values are character strings enclosed in single quotation marks and separated by blanks. For example,

```
midpoints="Midwest" "Northeast" "Northwest"
```

Specify the values in any order. If a character variable has an associated format, the specified values must be the *formatted* values. Character response values specified with the MIDPOINTS= option are not supported by DEVICE=JAVA.

You can selectively exclude some response variable values from the map, as shown here:

```
midpoints="Midwest"
```

Only those observations for which the response variable exactly matches one of the values listed in the MIDPOINTS= option are shown on the map. As a result, observations might be inadvertently excluded if values in the list are misspelled or if the case does not match exactly.

Specifying MIDPOINTS=OLD generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Featured in: Example 8 on page 1311

Restriction: Partially supported by Java

See also: The RANGE option

MISSING

accepts a missing value as a valid level for the response variable.

See also: "Displaying Map Areas and Response Data" on page 1250

NAME='*name*'

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default GRSEG name is GMAP. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, GMAP1.

See also: "About Filename Indexing" on page 99

NOLEGEND

suppresses the legend.

PERCENT

causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTIC= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or

MEAN. If you do not use the STATISTIC= option, then STATISTIC=FIRST is assumed and the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias: PERCENTAGE

See also: The STATFMT= option on page 1282, and the STATISTIC= option on page 1282

RANGE

causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints). For example, if MIDPOINTS=15 25 35, then the legend could show 10-20, 20-30, 30-40.

Restriction MIDPOINTS= must be specified for the RANGE option to have any effect. Not supported by ActiveX.

RELZERO

creates area heights that are relative to a zero value. By default, GMAP creates heights that are relative to the minimum value, which might or might not be zero. With the RELZERO option, zero value areas have no height.

Alias: REL0, RELATIVETOZERO

Restriction This option works only for variables that have no negative values.

STATFMT=*format-specification*

overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias: SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN

specifies the statistic for GMAP to chart. For character variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST	GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.
SUM	All observations matching a given ID value are added together and the summed value is charted.
FREQUENCY	A count of all rows with nonmissing values is charted unless you specify the MISSING option.
MEAN	All observations matching a given ID value are added together and then divided by the number of nonmissing observations matched. This value is then charted unless you specify the MISSING option.

Alias: STAT=

STRETCH

stretches map extents to cover all available space in the device. This might cause the map to be distorted. When this option is applied to the PROC GMAP statement, it applies to all statements. If applied to a single statement, it applies only to that statement.

Alias: STRETCHTOFIT, STR2FIT

Restriction: Not supported by Java and ActiveX

UNIFORM
causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option prescans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in a static legend and color distribution across all maps such that a single value always has the same color in multiple maps.

When specified on a PROC GMAP statement, the UNIFORM option applies to all AREA, BLOCK, CHORO, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual AREA, BLOCK, CHORO, or PRISM statement, the UNIFORM option applies only to the maps produced by that statement.

Restriction: Not supported by Java

WOUTLINE=*area-outline-width*
specifies the width, in pixels, of all map area outlines.

Default: 1

XLIGHT=*x*
YLIGHT=*y*
specify the coordinates of the imaginary light source in the map coordinate system. The position of the light source affects the way the sides of the map polygons are shaded. Although you can specify any point for the light source using the XLIGHT= and YLIGHT= options, the light source is actually placed in one of only four positions.

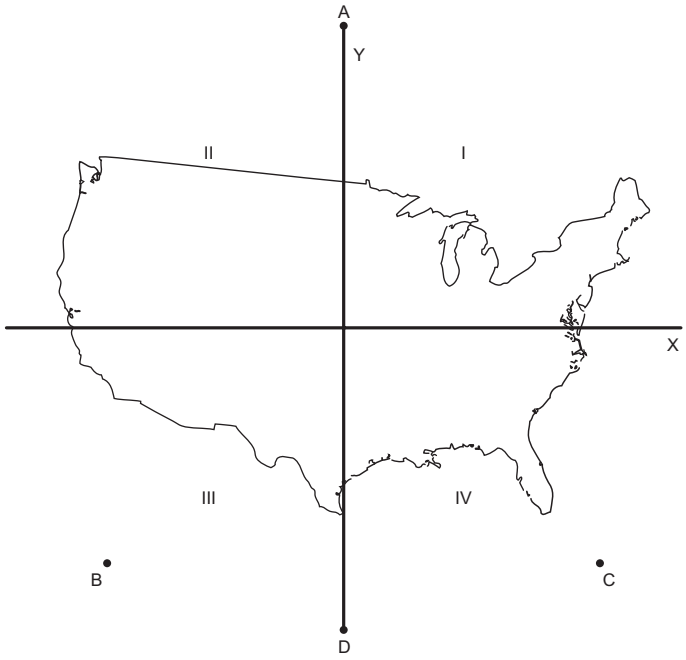
Table 43.3 on page 1283 shows how the point you specify is positioned.

Table 43.3 Light Source Coordinates

Specified Light Source	Light Source Position
in quadrants I or II, or on the X or +Y axis	behind the map (point A), and all side polygons are shadowed
on or within approximately 10 degrees of the Y axis	the viewing position (point D), and none of the side polygons are shadowed
in quadrant III (except within 10 degrees of the Y axis)	to the left of the map (point B), and the right-facing sides of polygons are shadowed
in quadrant IV (except within 10 degrees of the Y axis)	to the right of the map (point C), and the left-facing side polygons are shadowed

Figure 43.6 on page 1284 illustrates the light source positions. Assume that your viewing position, selected by the XVIEW=, YVIEW=, and ZVIEW= options, is point D.

Figure 43.6 Coordinates of Imagined Light Source in a Map Coordinate System



By default, the light source position is the same as the viewing position specified by the XVIEW=, YVIEW=, and ZVIEW= options. The light source position cannot coincide with the viewing reference point (0.5,0.5), which corresponds with the position directly above the center of the map.

Restriction: Not supported by Java and ActiveX

See also: XVIEW= on page 1284

XSIZE=*map-width <units>*
YSIZE=*map-height <units>*

specify the dimensions of the map that you are drawing. By default, the map uses the entire procedure output area.

Valid *units* are CELLS (character cells), CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). The default unit is CELLS.

If you specify values for *map-width* and *map height* that are greater than the dimensions of the procedure output area, the map is drawn using the default size. If you specify one value and not the other, the dimension is adjusted to maintain the correct aspect ratio.

Restriction: Not supported by Java and ActiveX

XVIEW=*x*
YVIEW=*y*
ZVIEW=*z*

specify the viewing position coordinates for the map. In this system, the four corners of the map lie on the X–Y plane at coordinates (0, 0, 0), (0, 1, 0), (1, 1, 0), and (1, 0, 0).

The viewing position cannot coincide with the viewing reference point at coordinates (0.5, 0.5, 0).

The value for *z* cannot be negative.

If you omit the XVIEW=, YVIEW=, and ZVIEW= options, the default coordinates are (0.5, –2,3). This viewing position is well above and to the south of the center of the map. One, two, or all three view coordinates can be specified; any that are not specified are assigned the default values.

Figure 43.5 on page 1268 shows the position of the viewing reference point, as well as the default viewing position.

To ensure that the polygon edges are distinguishable, the angle from vertical must be less than or equal to 45 degrees. If you specify a ZVIEW= value such that this condition cannot be satisfied (that is, a very small value), PROC GMAP increases the ZVIEW= value automatically so that the angle is 45 degrees or less. While you can use the XVIEW= and YVIEW= options with DEVICE=JAVA, ZVIEW= cannot be used with DEVICE=JAVA.

Alias: XV=, YV=, ZV=

Restriction: Partially supported by Java

SURFACE Statement

Creates three-dimensional surface maps in which levels of magnitude of the specified response variables are represented by spikes of varying height.

Requirements: At least one response variable is required and must be numeric. The ID statement must be used in conjunction with the SURFACE statement.

Global statements: FOOTNOTE, TITLE

Restriction: Not supported by Java and ActiveX

Description

The SURFACE statement specifies the variable or variables that contain the data that are represented on the map by raised map areas. This statement automatically determines the midpoints. You can use statement options to control spike proportions, specify the angle of view, and modify the general appearance of the map. For example, you can select the color and number of lines for the representation of the surface area. You can control the selection of spike heights and base widths.

In addition, you can use global statements to add titles and footnotes to the map. You can also enhance the map with an Annotate data set.

SURFACE *response-variable(s)* </ *option(s)*>;

option(s) can be one or more of the following:

- appearance options:
  - ANNOTATE=*Annotate-data-set*
  - CBODY=*surface-map-color*
  - CONSTANT=*n*
  - NLINES=*number-of-lines*
  - ROTATE=*degrees*
  - TILT=*degrees*
  - XSIZE=*map-width <units>*
  - YSIZE=*map-height <units>*
- mapping options:
  - PERCENT | PERCENTAGE
  - STATISTIC=FIRST | SUM | FREQUENCY | MEAN
  - STATFMT=*format-specification*

□ description options:
DESCRIPTION='*description*'
NAME='*name*'

Required Arguments

response-variable(s)
specifies one or more variables in the response data set, or in the merged response and feature table, that contain response values that are to be represented on the map. The *response-variable* must be numeric and must contain only positive values. Each response variable produces a separate map. All variables must be in the input data set. Multiple response variables are separated with blanks.

The GMAP procedure scales response variables for presentation on the map. The height of the spikes on the map correspond to the relative value of the response variable, not to the actual value of the response variable. However, when the viewing angle is changed, the spikes might not appear this way. The spikes in the front might appear to be higher than the spikes in the back, which represent greater values.

See also: "About Response Variables" on page 1249.

Options

SURFACE statement options affect all maps that are produced by that statement.

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate onto maps that are produced by the SURFACE statement. Annotate coordinate systems 1, 2, 7, and 8 are not valid with surface maps.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

CBODY=*surface-map-color*
specifies the color that is used to draw the surface map. Regardless of the current ODS style, the default color is the first color in the current color list.

Alias: CB=

CONSTANT=*n*
specifies a denominator to use in the distance decay function. This function determines the base width of the spike that is drawn at each map area center.

By default, CONSTANT=10. Values greater than 10 yield spikes that are wider at the base. Values less than 10 yield spikes that are narrower at the base.

Let x_k and y_k represent the coordinates, and z_k represent the function value at the center of each map area. The z_k values are scaled from 1 to 11. A square grid of x by y points (where the size of the grid is the NLINES= option value) and the associated function value *f(x,y)* are generated from the map area center value using this formula:

$$f(x,y) = \sum^{k} \left(1 - 1.5^{k} + .5\mathrm{D}^{3k}\right) \triangle^{k\mathrm{zk}}$$

where

$$\mathrm{D}^{k} = \left(x - x^{k}\right)^{2} + \left(y - y^{k}\right)^{k}$$

and

$$\triangle^k = \left[\text{martix cdelim} = \text{XXXXXXXXXXXXXXX}\, 1 \text{ if } \mathrm{D}^k < 1,\ 0 \text{ otherwise.}\right]$$

Alias: CON=

Featured in: Example 10 on page 1313

DESCRIPTION='*description*'
specifies the description of the catalog entry for the map. The maximum length for *description* is 256 characters. By default, the GMAP procedure assigns a description of the form SURFACE MAP OF *variable*, where *variable* is the name of the map variable.

The descriptive text is shown in each of the following:

- the "description" portion of the Results window
- the catalog-entry properties that you can view from the Explorer window
- the Table of Contents that is generated when you use CONTENTS= on an ODS HTML statement, assuming that the procedure output is generated while the contents page is open
- the Description field of the PROC GREPLAY window
- the chart description for Web output (depending on the device driver). For more information, see "PROC GANNO Statement" on page 914.

Alias: DES=

NAME='*name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default GRSEG name is GMAP. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, GMAP1.

See also: "About Filename Indexing" on page 99

NLINES=*number-of-lines*
specifies the number of lines used to draw the surface map. Values can range from 50 to 100; the higher the value, the more solid the map appears and the more resources used. By default, NLINES=50.

Alias: N=

Featured in: Example 10 on page 1313

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTIC= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or MEAN. If you do not use the STATISTIC= option, then STATISTIC=FIRST is assumed and the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias: PERCENTAGE

See also: The STATFMT= option on page 1288, and the STATISTIC= option on page 1288

ROTATE=*degrees*
specifies the degrees of the angle at which to rotate the map about the Z axis in the map coordinate system. The *degrees* argument can be any angle. Positive values indicate rotation in the counterclockwise direction. By default, ROTATE=70. The ROTATE= option also affects the direction of the lines that are used to draw the surface map.

Featured in: Example 10 on page 1313

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias: SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart. For character variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST	GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.
SUM	All observations matching a given ID value are added together and the summed value is charted.
FREQUENCY	A count of all rows with non-missing values is charted unless you specify the MISSING option.
MEAN	All observations matching a given ID value are added together and then divided by the number of non-missing observations matched. This value is then charted unless you specify the MISSING option.

Alias: STAT=

TILT=*degrees*
specifies the degrees of the angle at which to tilt the map about the X axis in the map coordinate system. The value of *degrees* can be 0 to 90. Increasing values cause the map to tilt backward and makes the spikes more prominent. Decreasing values make the map shape more distinguishable and the spikes less prominent. TILT=90 corresponds to viewing the map edge-on, while TILT=0 corresponds to viewing the map from directly overhead. By default, TILT=70.

Featured in: Example 10 on page 1313

XSIZE=*map-width <units>*
YSIZE=*map-height <units>*
specify the physical dimensions of the map. By default, the map uses the entire procedure output area.

Valid *units* are CELLS (character cells), CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). The default unit is CELLS.

If you specify values for *map-width* and *map-height* that are greater than the dimensions of the procedure output area, the map is drawn using the default size. And if you specify only one dimension, the other is scaled to maintain the aspect ratio.

Using FIPS Codes and Province Codes

The map area identification variable in some SAS/GRAPH map data sets contain standardized numeric codes. The data sets for the United States contain a variable whose values are FIPS (Federal Information Processing Standards) codes. The data sets for Canada contain standard province codes or census division codes. When you use the GMAP procedure with a traditional map data set, the variables that identify map areas in your response data set must have the same values as the map area identification variables in the traditional map data set.

If both a feature table and a response data set contain FIPS Codes or Province Codes, then once both data sets have been sorted, an SQL or DATA step MERGE can be used to merge the two data sets using the variable containing the codes. However, with the merged response and feature table, the identification variable used in the GMAP procedure must be the $GEOREF formatted variable that contains the spatial information. See "$GEOREF format" on page 1246 for more information.

If the map area identification variables in your response data set are state or province names or abbreviations, convert them to FIPS codes or province codes before using the response data set with one of the map data sets supplied by SAS. Table 43.4 on page 1289 lists the FIPS codes for the United States and Table 43.5 on page 1290 lists the standard codes for Canadian provinces.

Note: Alternatively, you can convert the FIPS code or province codes in your map data set to match the names in your response data. △

Table 43.4 U.S. FIPS Codes

FIPS Code	State	FIPS Code	State
01	Alabama	30	Montana
02	Alaska	31	Nebraska
04	Arizona	32	Nevada
05	Arkansas	33	New Hampshire
06	California	34	New Jersey
08	Colorado	35	New Mexico
09	Connecticut	36	New York
10	Delaware	37	North Carolina
11	District of Columbia	38	North Dakota
12	Florida	39	Ohio
13	Georgia	40	Oklahoma
15	Hawaii	41	Oregon
16	Idaho	42	Pennsylvania
17	Illinois	44	Rhode Island
18	Indiana	45	South Carolina
19	Iowa	46	South Dakota
20	Kansas	47	Tennessee
21	Kentucky	48	Texas

FIPS Code	State	FIPS Code	State
22	Louisiana	49	Utah
23	Maine	50	Vermont
24	Maryland	51	Virginia
25	Massachusetts	53	Washington
26	Michigan	54	West Virginia
27	Minnesota	55	Wisconsin
28	Mississippi	56	Wyoming
29	Missouri	72	Puerto Rico

Table 43.5 Canadian Province Codes

Province Code	Province
10	Newfoundland
11	Prince Edward Island
12	Nova Scotia
13	New Brunswick
24	Quebec
35	Ontario
46	Manitoba
47	Saskatchewan
48	Alberta
59	British Columbia
60	Yukon
61	Northwest Territories

Note: The ID variables in Canadian maps are character. △

The MAPS.CNTYNAME data set contains a cross-reference of names and FIPS codes for all counties in the United States. The MAPS.CANCENS data set contains a cross-reference of census district names and codes for Canadian provinces.

Base SAS software provides several functions that convert state names to FIPS codes and vice versa. The following table lists these functions and a brief description of each. See *SAS Language Reference: Dictionary* for more information.

Table 43.6 FIPS and Postal Code Functions

Function	Description
STFIPS	converts state postal code to FIPS state code
STNAME	converts state postal code to state name in upper case
STNAMEL	converts state postal code to state name in mixed case
FIPNAME	converts FIPS code to state name in upper case

FIPNAMEL	converts FIPS code to state name in mixed case
FIPSTATE	converts FIPS code to state postal code

Using Formats for Map Variables

You can specify an output map area name or numeric value using one of the predefined formats for maps. The following prefixes are used in the names of the formats for maps:

CONT	Continent
CNTRY	Country
GLC	Geographic Location Code, distributed by Government Services Administration, USA
ISO	International Standard Organization

The formats for maps are located in the SASHELP.MAPFMTS catalog. See the MAPS.NAMES table to view all the continent and country names and corresponding GLC, ISO, and numeric representation for the continent values.

To use one of the formats for maps, you must specify the SASHELP.MAPFMTS catalog on the FMTSEARCH= option on a SAS OPTIONS statement:

```
options fmtsearch=(sashelp.mapfmts);
```

In addition to using the PUT statement (as shown in the examples in the following table), the formats can also be invoked using a FORMAT statement.

Note: If the input to a format is invalid, the format is "**" or "***" . △

Table 43.7 Formats for Maps

FORMAT	DESCRIPTION	EXAMPLE	OUTPUT
contfmt	use a continent's numeric value to output the continent's name	cont= 91 **put(cont,contfmt.);**	North America
glcna	use the country's GLC numeric code to output the country's GLC alpha code	id=460 **put(id,glcna.);**	IR
glcnlu	use the GLC numeric code to output the country's long name in uppercase	id=460 **put(id,glcnlu.);**	IRAN, ISLAMIC REPUBLIC OF
glcnsu	use the GLC numeric code to output the country's short name in uppercase	id=460 **put(id,glcnsu.);**	IRAN
glcnsm	use the GLC numeric code to output the country's name in mixed case	id=460 **put(id,glcnsm.);**	Iran

FORMAT	DESCRIPTION	EXAMPLE	OUTPUT
ison2a	use the country's ISO numeric code to output the country's ISO alpha2 code	iso=364 **put(iso,ison2a.);**	IR
ison3a	use the country's ISO numeric code to output the country's ISO alpha3 code	iso=364 **put(iso,ison3a.);**	IRN
isonlu	use the country's ISO numeric code to output the country's long name in uppercase	iso=364 **put(iso,isonlu.);**	IRAN, ISLAMIC REPUBLIC OF
isonsu	use the country's ISO numeric code to output the country's short name in uppercase	iso=364 **put(iso,isonsu.);**	IRAN
$cntrysl	use a country's short name in uppercase to output the country's long name in uppercase	name='IRAN' **put(name,$cntrysl.);**	IRAN, ISLAMIC REPUBLIC OF
$glcalu	use the GLC alpha code to output the country's long name in uppercase	country='IR' **put(country,$glcalu.);**	IRAN, ISLAMIC REPUBLIC OF
$glcan	use the country's GLC alpha code to output the country's GLC numeric code	country='IR' **put(country,$glcan.);**	460
$glcsua	use the country's short name in uppercase to output the GLC alpha code name	name='IRAN' **put(name,$glcsua.);**	IR
$glcsun	use the country's short name in uppercase to output the country's GLC numeric code	name='IRAN' **put(name,$glcsun.);**	460
$glcsma	use the country's short name in mixed-case to output the country's GLC alpha code	mixname='Iran' **put(mixname,$glcsma.);**	IR
$glcsmn	use the country's short name in mixed-case to output the country's GLC numeric code	mixname='Iran' **put(mixname,$glcsmn.);**	460

FORMAT	DESCRIPTION	EXAMPLE	OUTPUT
$glcprov	use a province/city name appended by || as a delimiter, followed by the country's GLC alpha code to output a province||country code, the province/city code, and the country's GLC alpha numeric code	provname='TEHRAN||IR' **put(provname,$glcprov.);**	8250460 8250 — province/ city code 460 — country GLC numeric code
$isosu2a	use the country's short name in uppercase to output the country's ISO alpha2 code	name='IRAN' **put(name,$isosu2a.);**	IR
$isosu3a	use the country's name in uppercase to output the country's ISO alpha3 code	name='IRAN' **put(name,$isosu3a.);**	IRN
$isosun	use the country's short name in uppercase to output the country's ISO numeric code	name='IRAN' **put(name,$isosun.);**	364
$isoa2lu	use the country's ISO alpha2 code to output the country's long name in uppercase	alpha2='IR' **put(alpha2,$isoa2su.);**	IRAN, ISLAMIC REPUBLIC OF
$isoa2n	use the country's ISO alpha2 code to output the country's ISO numeric code	alpha2='IR' **put(alpha2,$isoa2n.);**	364
$isoa2su	use the country's ISO alpha2 code to output the country's short name in uppercase	alpha2='IR' **put(alpha2,$isoa2lu.);**	IRAN
$isoa3lu	use the country's ISO alpha3 code to output the country's long name in uppercase	alpha3='IRN' **put(alpha3,$isoa3lu.);**	IRAN, ISLAMIC REPUBLIC OF
$isoa3n	use the country's ISO alpha3 code to output the country's ISO numeric code	alpha3='IRN' **put(alpha3,$isoa3n.);**	364
$isoa3su	use the country's ISO alpha3 code to output the country's short name in uppercase	alpha3='IRN' **put(alpha3,$isoa3su.);**	IRAN

FORMAT	DESCRIPTION	EXAMPLE	OUTPUT
$isosm2a	use the country's short name in mixed-case to output the country's ISO alpha2 code	mixname='Iran' **put(mixname,$isosm2a.);**	IR
$isosm3a	use the country's short name in mixed-case to output the country's ISO alpha3 code	mixname='Iran' **put(mixname,$isosm3a.);**	IRN
$isosmn	use the country's short name in mixed-case to output the country's ISO numeric code	mixname='Iran' **put(mixname,$isosmn.);**	364

Using SAS/GRAPH Map Data Sets

Accessing Detailed Descriptions of Map Data Sets

You might need detailed information on the map data sets in order to determine their type, size, the variables they contain, or, in the case of traditional data sets, whether they are projected or unprojected. You can get this information by using the CONTENTS or DATASETS procedure, or browsing the MAPS.METAMAPS (see "The METAMAPS Data Set" on page 1247) data set in the MAPS library (or the library where your SAS-supplied map data sets reside). If the libref MAPS has automatically been assigned, you can see a complete list of map data sets by viewing the MAPS.METAMAPS data set.

These statements list the map data sets in the SAS library that is assigned to the libref MAPS:

```
proc datasets lib=maps;
run;
```

The following statements provide detailed information on a traditional map data set, including the number of observations, the variables in each data set, and a description of each variable:

```
proc contents data=maps.canada3;
run;
```

To see the contents and descriptions of all of the map data sets supplied by SAS you can specify DATA=MAPS._ALL_ in the CONTENTS procedure. See the *Base SAS Procedures Guide* for more information on the CONTENTS and DATASETS procedures.

Customizing SAS/GRAPH Map Data Sets

You can customize the area that is displayed on your map by using only part of a particular map data set. There are several ways to accomplish this. You can use WHERE processing or a DATA step to subset the map data to be used by the GMAP procedure.

With the traditional map data set, you can also use the GPROJECT procedure to create a rectangular subset of a map data set by using minimum and maximum

longitude and latitude values. For more information, see Chapter 46, "The GPROJECT Procedure," on page 1395.

You can combine traditional map data sets in either of these situations:

- The map data sets to be combined were originally projected together.
- The map data sets all contain the same type of coordinates. That is, all are in radians or all are in degrees, and the longitude coordinates are measured in the same direction.

Traditional map data sets supplied by SAS that have coordinates expressed only as longitude and latitude, with variable names LONG and LAT, must be renamed X and Y and should be projected before you use them with the GMAP procedure.

Subsetting Traditional Map Data Sets

Some of the SAS/GRAPH map data sets contain a large number of observations. Programs that use only a few states or provinces run faster if you exclude the unused portion of the map data set or use a reduced map data set. SAS provides several ways to accomplish this. One is to use the WHERE statement or WHERE= data set option within the GMAP procedure to select only the states or provinces you want.

The WHERE statement and WHERE= data set option are most useful when you produce a simple map and do not need to make any other changes to the data set. For example, to use only the observations for Quebec in the CANADA traditional map data set, begin the GMAP procedure with this statement:

```
proc gmap map=maps.canada(where=(province="24"));
```

To use only North Carolina in US2MERGED (a data set created by using SQL or DATA step MERGE on the feature table US2 and a response data set also containing the variable STATE) the GMAP procedure would begin with the following statement:

```
proc gmap data=work.us2merged(where=(STATE=37));
```

The WHERE= data set option applies only to the data set that you specify in the argument in which the WHERE= option appears. If you use the WHERE statement, the WHERE condition applies to the traditional map data set and the response data sets or the merged response and feature table.

Another approach is to use a DATA step to create a subset of the larger data set. This code illustrates another way to extract the observations for Quebec from the CANADA traditional map data set:

```
data quebec;
   set maps.canada(where=(province="24"));
```

This code illustrates another way to extract North Carolina data from the US2 feature table:

```
data ncarolina;
   set maps.us2(where=(STATE=37));
```

This approach is most useful when you want to create a permanent subset of a map data set or when you need to make additional changes to the map data set.

Also see Chapter 49, "The GREMOVE Procedure," on page 1459 for an example of how to use GREMOVE to create a regional map from one of the traditional map data sets that are supplied with SAS/GRAPH.

Reducing Traditional Map Data Sets

A *reduced map data set* is one that can be used to draw a map that retains the overall appearance of the original map but that contains fewer points, requires

considerably less storage space, and can be drawn much more quickly. You can improve performance by plotting fewer observations for each map area. You reduce a traditional map data set when you subset it on the variable DENSITY. You can add the variable DENSITY to a map data set by using the GREDUCE procedure. For more information, see Chapter 48, "The GREDUCE Procedure," on page 1447.

Note: Many of the map data sets in the MAPS library are supplied with a DENSITY variable. △

An *unreduced map data set* contains all of the coordinates that were produced when the map was digitized. This type of map data set has more observations than most graphics output devices can accurately plot. Some unreduced map data sets already contain a DENSITY variable like the one calculated by the GREDUCE procedure, so it is not necessary to use the GREDUCE procedure to process these data sets. Values for DENSITY range from 0 through 6 (the lower the density, the coarser the boundary line).

You can set the DENSITY value by using the DENSITY= option on the PROC GMAP statement. For example, the following statement excludes all points with a density level of 2 or greater:

```
proc gmap map=maps.states density=2;
```

The resulting map is much coarser than one drawn by using all of the observations in the data set, but it is drawn much faster.

Another way to create a reduced map data set is to use a DATA step to exclude observations with larger density values:

```
data states;
   set maps.states(where=(density<2));
```

Projecting Traditional Map Data Sets

Map data can be stored as unprojected or projected coordinates. Unprojected map data contains spherical coordinates, that is, longitude and latitude values usually expressed in radians.*

Many of the map data sets in the MAPS library are projected. However, these map data sets contain only unprojected coordinates and should be projected before you use them.

- CANADA3
- CANADA4
- COUNTIES
- COUNTY
- STATES

If the projection supplied with the traditional map data set does not meet your needs, then you can use the GPROJECT procedure (on unprojected map coordinates) to create a different projection. For more information on traditional map data sets with unprojected coordinates, see "Traditional Map Data Sets Containing X, Y, LONG, and LAT" on page 1245. You should select a projection method that least distorts the regions that you are mapping. (All projection methods inherently distort map regions.) See Chapter 46, "The GPROJECT Procedure," on page 1395 for more information.

Note: Using an unprojected traditional map data set with the GMAP procedure can cause your map to be reversed. △

* If your data is in degrees, then it can be converted to radians by multiplying by the degree-to-radian constant [atan(1)/45].

Controlling the Display of Lakes

Some countries contain a lake that is located completely within a single unit area. Occasionally these lakes can be a problem when mapping traditional map data sets. In addition, displaying lakes might not be appropriate for some applications. In these cases, you might want to remove the lakes from the map data set before you proceed.

Traditional map data sets that contain coordinates for a lake that is located within a single internal division are identified by the presence of the numeric variable LAKE. The value of LAKE is 1 for points that correspond to lakes and 0 otherwise. The following statements illustrate how to delete the lakes from your traditional map data sets using WHERE processing:

```
proc gmap map=maps.chile(where=(lake=0))
          data=maps.chile;
   id id;
   choro id / levels=1 nolegend;
   title box=1 f=none h=4
         "Chile with Lakes Removed";
run;
```

You can also create a new traditional map data set that is a subset of the traditional map data set:

```
data nolake;
   set maps.chile(where=(lake=0));
run;
```

Creating Traditional Map Data Sets

In addition to using map data sets that are supplied with SAS/GRAPH software, you can also create your own map data sets. Map data sets are not limited to geographic data; you use them to define other spaces such as floor plans.

A unit area is defined by observations in the map data set that have the same identification (ID) variable value. A unit area might be composed of a single polygon or a collection of polygons. A polygon is defined by all of the observations that have the same SEGMENT variable value within the same unit area.

- If the unit area is a single polygon, then all values of SEGMENT are the same (alternatively, you can omit the SEGMENT variable).
- If the unit area contains multiple polygons, such as islands, then the SEGMENT variable has multiple values. For example, in the MAPS.US data set, the state of Hawaii (a unit area) contains six different values in the SEGMENT variable, one for each island in the state.
- If the unit area contains enclosed polygons (holes), such as lakes, then the SEGMENT variable has one value but the interior polygon is defined by separate boundaries. To separate boundaries, a missing X and Y value must be inserted at the separation point. For example, in the CANADA2 data set supplied with SAS/GRAPH, the map data for the Northwest Territories (a unit area) use enclosed polygons for two lakes.

Creating a Unit Area that is a Single Polygon

This DATA step creates a SAS data set that contains coordinates for a unit area with a single polygon, a square:

```
data square;
   input id x y;
   datalines;
1 0 0
1 0 40
1 40 40
1 40 0
;
```

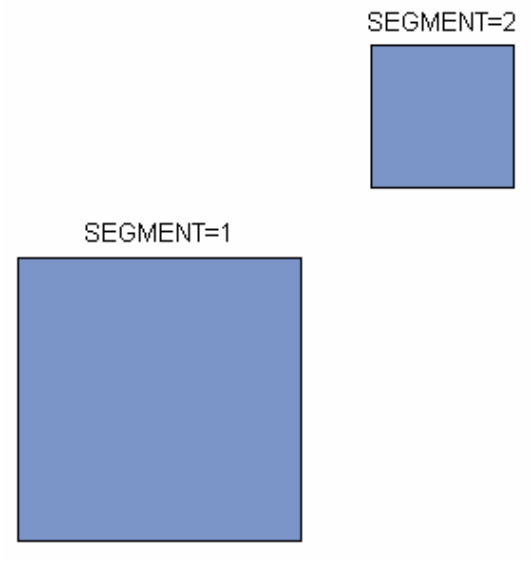
This data set does not have a SEGMENT variable.

Creating a Unit Area that Contains Multiple Polygons

Use different values of the SEGMENT variable to create separate polygons within a single unit area. For example, this DATA step assigns two values to the SEGMENT variable. The resulting data set produces a single unit area that contains two polygons, as shown in Figure 43.7 on page 1299:

```
data map;
   input id $ segment x y;
   datalines;
square   1 0 0
square   1 0 4
square   1 4 4
square   1 4 0
square   2 5 5
square   2 5 7
square   2 7 7
square   2 7 5
;
```

Figure 43.7 Single Unit Area with Two Segments (Polygons)

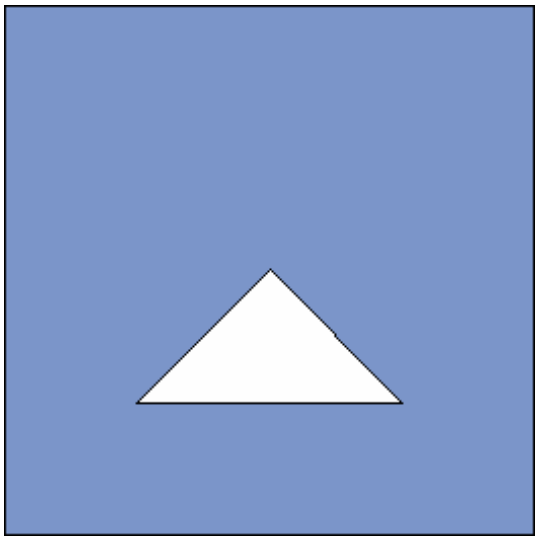


Creating a Unit Area that Contains Enclosed Polygons as Holes

Use separate boundaries to create an enclosed polygon (that is, a polygon that falls within the primary polygon for a single segment). The boundary for the hole is separated from the primary polygon boundary by inserting a missing value for X and Y. For example, the data set that is created by this DATA step produces the map shown in Figure 43.8 on page 1300:

```
data map;
   input id $ segment x y;
   datalines;
square   1 0 0
square   1 0 4
square   1 4 4
square   1 4 0
square   1 . .
square   1 1 1
square   1 2 2
square   1 3 1
;
```

Figure 43.8 Single Unit Area with Hole



Note: A single map segment (a section of a unit area with a single value of the SEGMENT variable) cannot contain multiple polygons without at least one observation with missing values for X and Y. All segments within the map data sets that are supplied by SAS/GRAPH contain a single polygon that can have one or more separate boundaries, each separated by an observation with missing values for X and Y. △

Creating a Unit Area that Contains Another Area

Sometimes rather than a hole or lake, an enclosed polygon represents a separate map area. For example, in MAPS.AFRICA, the country of Lesotho is surrounded by the country of South Africa.

To create an enclosed map area:

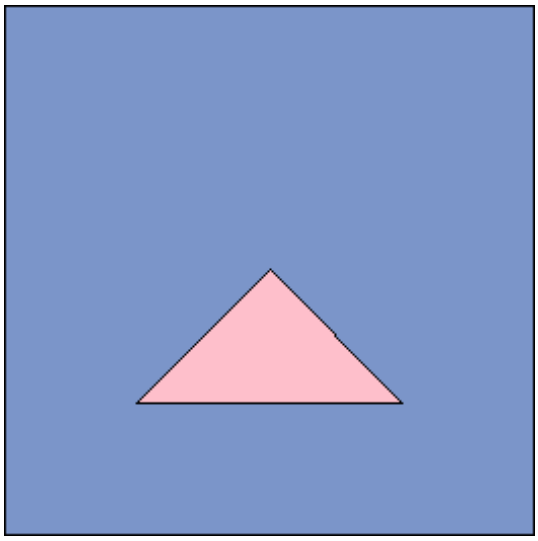
1. Create an observation with missing values for X and Y for the surrounding area.
2. Define the boundary as part of the surrounding area by the using ID value for the surround area.
3. Define the boundary as part of the enclosed area by using the ID value for the enclosed area.

For example, this DATA step creates a data set that produces the map shown in Figure 43.9 on page 1301:

```
data map;
   input id $ segment x y;
   datalines;
square   1 0 0
square   1 0 4
square   1 4 4
square   1 4 0
square   1 . .
square   1 1 1
square   1 2 2
square   1 3 1
triangle 1 1 1
triangle 1 2 2
triangle 1 3 1
```

```
;
```

Figure 43.9 Unit Area within a Unit Area



Examples

The following examples include features from one or more of the GMAP statements.

Note: 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 HTML; *procedure-code*; ODS HTML CLOSE;**). See "RUN-Group Processing" on page 56 for more information. △

Example 1: Producing a Simple Block Map

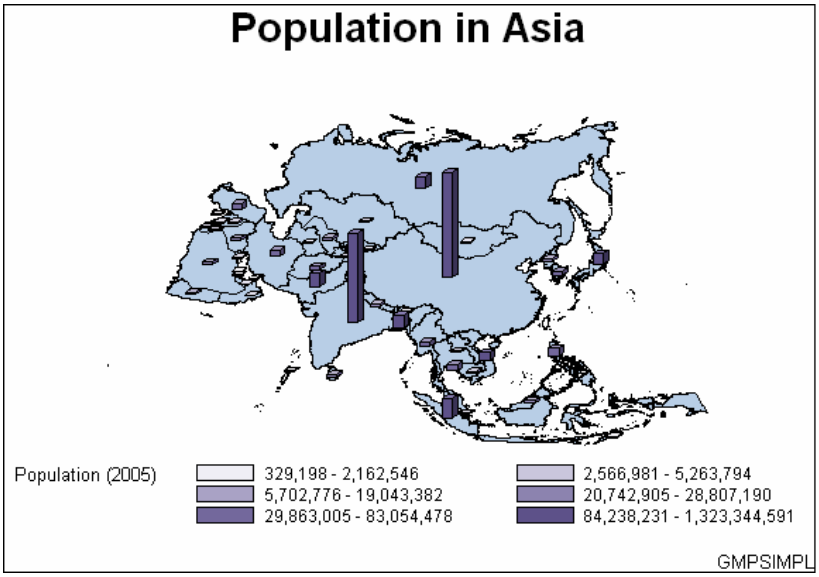
Procedure features:

ID statement

BLOCK statement option:

BLOCKSIZE=
RELZERO

Sample library member: GMPSIMPL



This example produces a block map that shows population of countries in Asia. Since the DISCRETE option is not used, the response variable is assumed to have a continuous range of values. Because neither the LEVELS= nor MIDPOINTS= option is used, the GMAP procedure selects a number of levels based on the number of map areas and then calculates the appropriate response levels.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

```
title1 "Population in Asia";
footnote1 j=r "GMPSIMPL";
```

Produce the block map. The ID statement specifies the variable that is in both the map data set and the response data set and defines map areas. The BLOCK statement specifies the variable in the response data set that contains the response values for each of the map areas. The BLOCKSIZE= option specifies the width of the blocks. The RELZERO option specifies that the block values are relative to zero.

```
proc gmap data=sashelp.demographics(where=(cont=95))
          map=maps.asia all;
   id id;
   block pop / blocksize=1 relzero;
run;
quit;
```

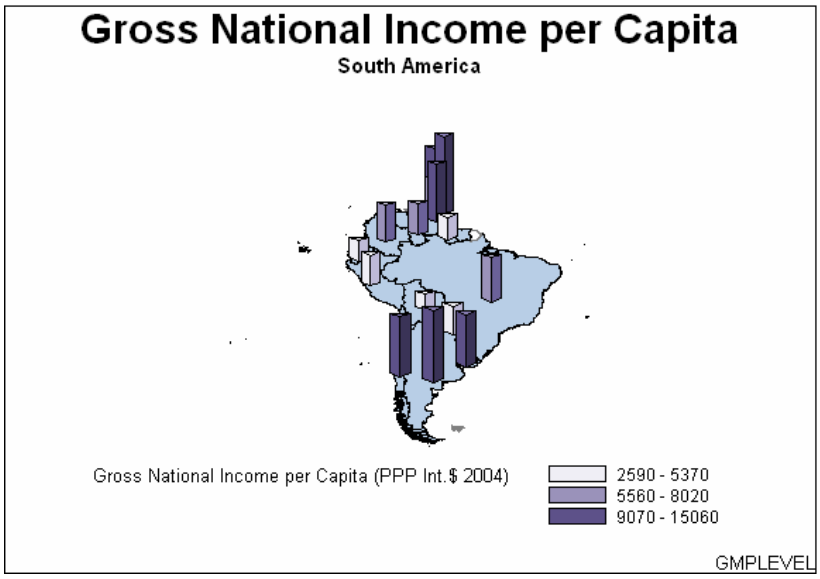
Example 2: Specifying Response Levels in a Block Map

Procedure features:

BLOCK statement options:

CEMPTY=
LEVELS=
SHAPE=
RELZERO

Sample library member: GMPLEVEL



This example uses the LEVELS= option to specify the number of response levels for the blocks. The LEVELS= option tells GMAP how many response levels and the GMAP procedure calculates the quantiles.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

```
title1 "Gross National Income per Capita";
title2 "South America";
footnote1 j=r "GMPLEVEL";
```

Produce the block map. The LEVELS= option specifies the number of response levels for the graph. The SHAPE= option draws the blocks as prisms. The RELZERO option specifies that the block values are relative to zero. The CEMPTY= option specifies the outline color for map areas that have missing data.

```
proc gmap data=sashelp.demographics(where=(cont=92))
          map=maps.samerica all;
   id id;
```

```
   block gni / levels=3 shape=prism
               relzero cempty=gray;
run;
quit;
```

Example 3: Assigning a Format to the Response Variable

Procedure features:

BLOCK statement options:

LEGEND=
RELZERO

AREA statement options:

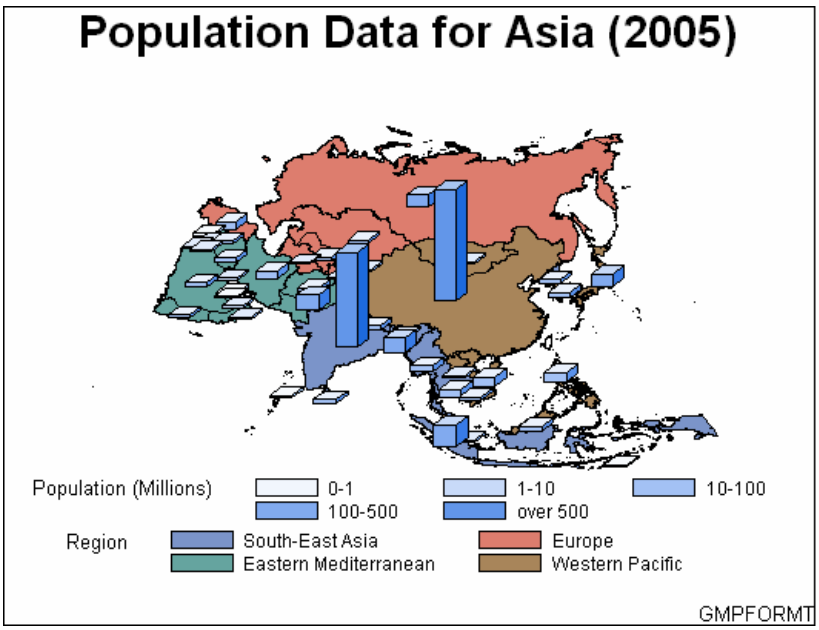
MIDPOINTS=

Other features:

FORMAT statement

LEGEND statement

Sample library member: GMPFORMT



This example creates formats for the response variables. The format for the POP variable defines and labels ranges of values. These ranges appear in the legend and make the map easier to understand.

The example also uses the AREA statement to patterns the map areas by region.

Set the graphics environment.

```
goptions reset=all border;
```

Create a format for POP.POPFMT. defines the ranges of values for POP and labels the values.

```
proc format;
  value popfmt low-1000000="0-1"
               1000001-10000000="1-10"
               10000001-100000000="10-100"
               100000001-500000000="100-500"
               500000001-high="over 500";
run;
```

Create a format for REGION.REGIONFMT. labels the values for REGION.

```
proc format;
  value $ regionfmt "SEAR" = "South-East Asia"
                    "EUR" = "Europe"
                    "EMR" = "Eastern Mediterranean"
                    "WPR" = "Western Pacific";
run;
```

Define the title and footnote for the map.

```
title1 "Population Data for Asia (2005)";
footnote j=r "GMPFORMT";
```

Assign the legend label.

```
legend1 label=("Population (Millions)");
```

Produce the block maps. The FORMAT statements assign POPFMT. to the POP variable and $REGIONFMT. to the REGION variable. The AREA statement assigns patterns to the map areas according to the values of the REGION variable. The RELZERO option specifies that the blocks values are relative to zero.

```
proc gmap data=sashelp.demographics(where=(cont=95))
          map=maps.asia all;
   format pop popfmt.;
   format region $regionfmt.;
   id id;
   area region / midpoints="SEAR" "EUR" "EMR" "WPR";
   block pop / levels=all
               legend=legend1
               relzero;
run;
quit;
```

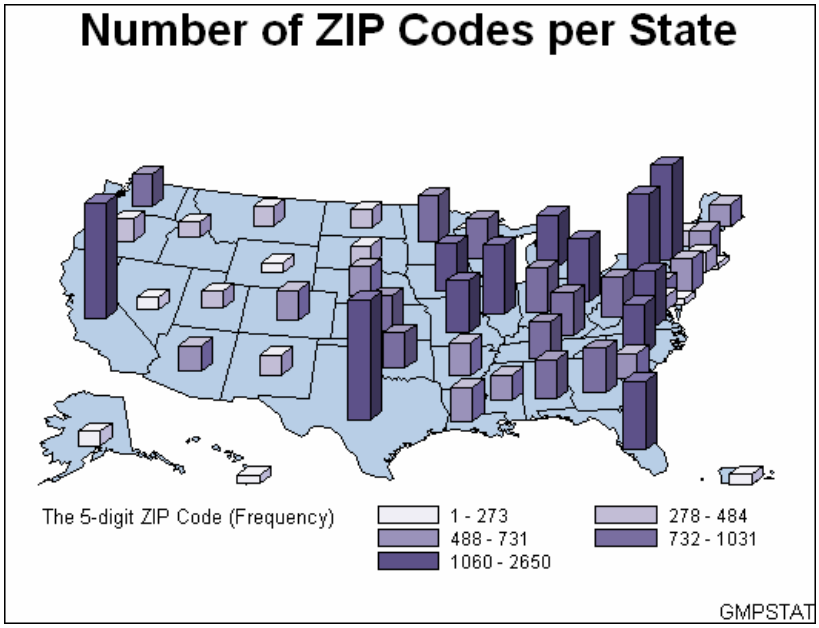
Example 4: Specifying the Statistic for the Response Variable

Procedure features:

BLOCK statement options:

STATISTIC=
LEVELS=
RELZERO

Sample library member: GMPSTAT



This example specifies the statistic for the response variable that is displayed by the block map. The STATISTIC= option specifies that the statistic is frequency rather than the default statistic (sum).

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

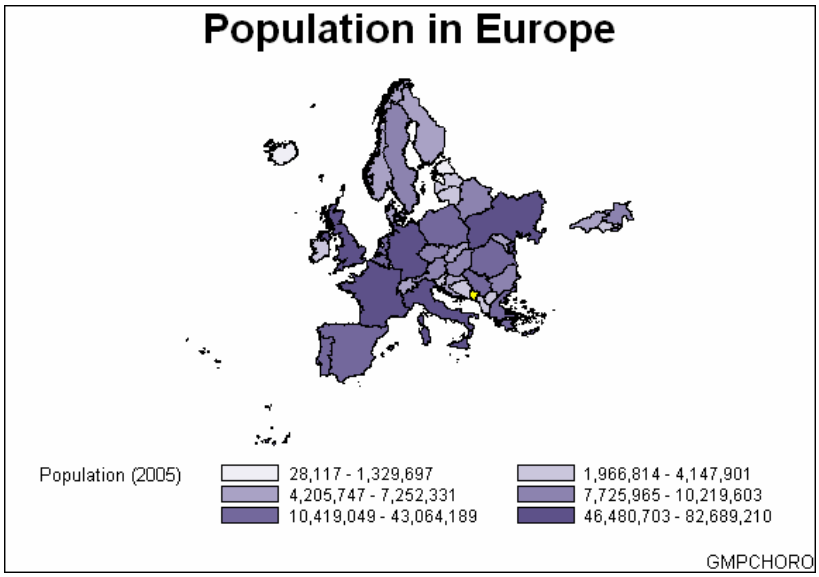
```
title1 "Number of ZIP Codes per State";
footnote j=r "GMPSTAT";
```

Produce the block maps. The FORMAT statements assign POPFMT. to the POP variable and $REGIONFMT. to the REGION variable. The AREA statement assigns patterns to the map areas according to the values of the REGION variable. The RELZERO option specifies that the blocks values are relative to zero.

```
proc gmap map=maps.us data=sashelp.zipcode all;
   id state;
   block zip / statistic=frequency
                    levels=5 relzero;
run;
quit;
```

Example 5: Producing a Simple Choropleth Map

Procedure features:
ID statement
CHORO statement option:
CDEFAULT=
Sample library member: GMPCHORO



This example produces a choropleth (two-dimensional) map that shows the population of countries in Europe. Since the DISCRETE option is not used, the response variable is assumed to have a continuous range of values. Because neither the LEVELS= nor MIDPOINTS= options are used, the GMAP procedure selects a number of levels based on the number of map areas and then calculates the appropriate response levels. The legend shows the range of values for each level.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

```
title1 "Population in Europe";
footnote1 j=r "GMPCHORO";
```

Produce the choropleth map. The ID statement specifies the variable that is in both the map data set and the response data set that defines map areas. CDEFAULT= specifies the color for the map areas that have missing data. The WHERE= clause on the MAP= option excludes the islands of Greenland and Svalbard, which have no data in DEMOGRAPHICS data set.

```
proc gmap map=maps.europe(where=(id ne 405 and id ne 845))
          data=sashelp.demographics(where=(cont=93)) all;
  id id;
  choro pop / cdefault=yellow;
run;
quit;
```

Example 6: Labeling Provinces on a Map

Procedure features:

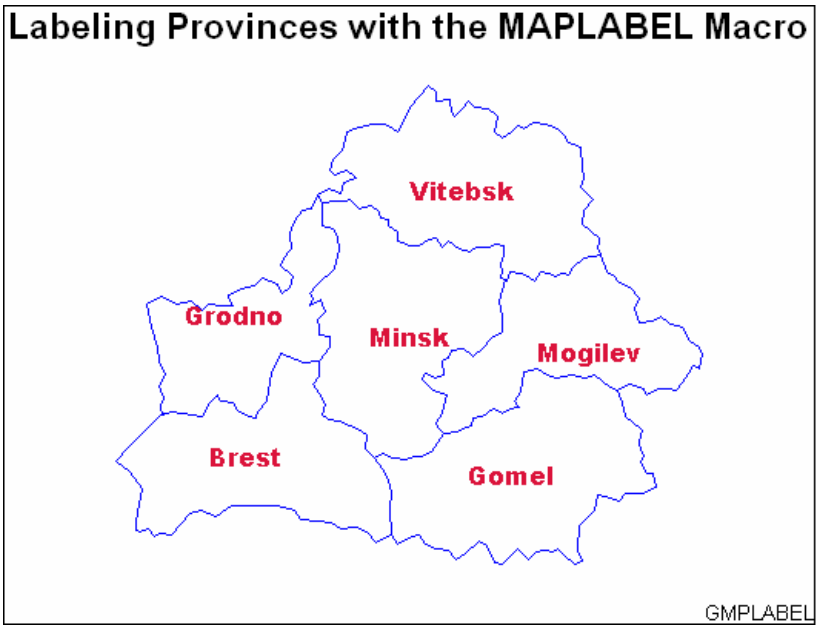
CHORO statement options:

ANNOTATE=
NOLEGEND

Other features:

Annotate Facility

Sample library member: GMPLABEL



This example uses the Annotate facility to add labels to each area in a map of Belarus. The CHORO statement assigns the Annotate data set to the map.

The %MAPLABEL annotate macro is used to create and position the map labels. For more information about this macro, see "%MAPLABEL Macro" on page 747.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

```
title "Labeling Provinces with the MAPLABEL Macro";
footnote j=r "GMPLABEL";
```

Define pattern characteristics. PATTERN1 defines a single map pattern that is repeated for each of the six map areas (provinces). The pattern is an empty fill with a blue border. The VALUE= option defines a map/plot pattern. Specifying a color causes PATTERN1 to generate only one pattern definition. The REPEAT= option specifies the number of times to repeat the pattern definition.

```
pattern1 value=empty color=blue repeat=6;
```

Create the Annotate data set. The %ANNOMAC macro enables the annotate macros. The %MAPLABEL annotate macro creates the annotate data set.

```
%annomac;
%maplabel (maps.belarus, maps.belarus2, work.labelout, idname, id, font=Arial Black,
           color=crimson, size=4, hsys=3);
```

Produce the choropleth map. The NOLEGEND option suppresses the legend. The ANNOTATE= option specifies the data set to annotate the map.

```
proc gmap map=maps.belarus data=maps.belarus;
   id id;
   choro id / nolegend annotate=labelout;
run;
quit;
```

Example 7: Producing a Simple Prism Map

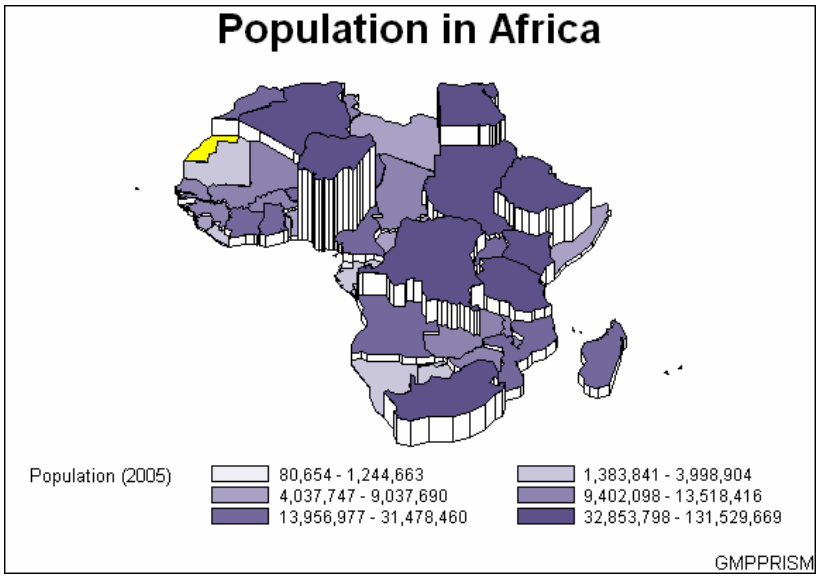
Procedure features:

ID statement

PRISM statement option:

CDEFAULT=
RELZERO

Sample library member: GMPPRISM



This example produces a prism map of the population of countries in Africa. Since the DISCRETE option is not used, the response variable is assumed to have a continuous range of values. Because neither the LEVELS= nor MIDPOINTS= option is used, the GMAP procedure selects a number of levels based on the number of map areas and then calculates the appropriate response levels.

Since the XVIEW=, YVIEW=, and ZVIEW= options are not used, the default viewing position, above and to the east and south of the center of the map, is used. Since the XLIGHT= and YLIGHT= options are not used, none of the side polygons of the prisms are shadowed. The light source is the same as the viewing position.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

```
title1 "Population in Africa";
footnote1 j=r "GMPPRISM";
```

Produce the prism map. The ID statement specifies the variable in the map data set and the response data set that defines map areas. The CDEFAULT= option sets the color of map areas that have missing data. The RELZERO option makes the prism heights relative to zero.

```
proc gmap data=sashelp.demographics(where=(cont=94))
          map=maps.africa density=low all;
   id id;
   prism pop / cdefault=yellow relzero;
run;
quit;
```

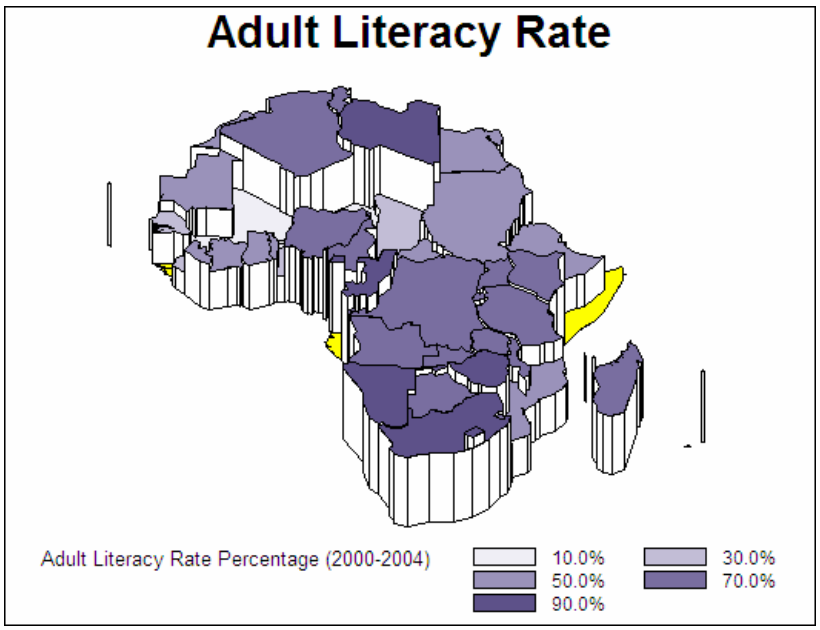
Example 8: Specifying Midpoints in a Prism Map

Procedure features:

PRISM statement options:

MIDPOINTS=
CDEFAULT=

Sample library member: GMPMIDPT



This example specifies a set of midpoints that are used to create the response levels.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the map.

```
title1 "Adult Literacy Rate";
```

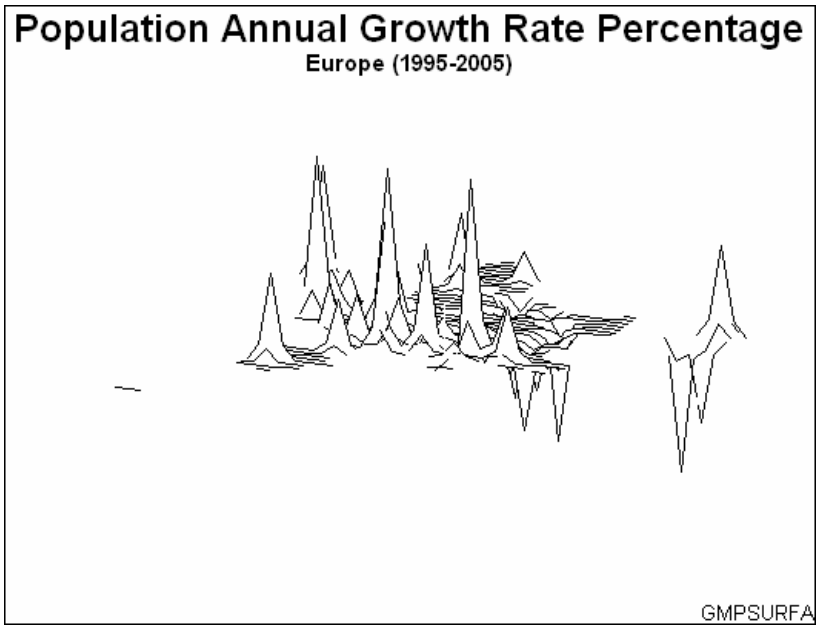
Produce the prism map. The MIDPOINTS= option specifies the response levels for the map. The CDEFAULT= option sets the color of map areas that have missing data.

```
proc gmap data=sashelp.demographics(where=(cont=94))
          map=maps.africa density=low all;
   id id;
   format adultliteracypct percentn7.2;
   prism adultliteracypct / midpoints=10 to 90 by 20
```

```
                                    cdefault=yellow;
run;
quit;
```

Example 9: Producing a Simple Surface Map

Procedure features:
SURFACE statement
Sample library member: GMPSURFA



This example produces a surface map that shows the annual population growth rate of countries in Europe. Because the CONSTANT= and NLINES= options are not used, the GMAP procedure draws a surface that consists of 50 lines and uses the default decay function to calculate spike height and base width. And because the ROTATE= and TILT= options are not used, the map is rotated 70 degrees around the Z axis and tilted 70 degrees with respect to the X axis.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

```
title1 "Population Annual Growth Rate Percentage";
title2 "Europe (1995--2005)";
footnote1 j=r "GMPSURFA";
```

Produce the surface map. The ID statement specifies the variable in the map data set and the response data set that defines the map areas.

```
proc gmap map=maps.europe data=sashelp.demographics;
   id id;
   surface popagr;
run;
quit;
```

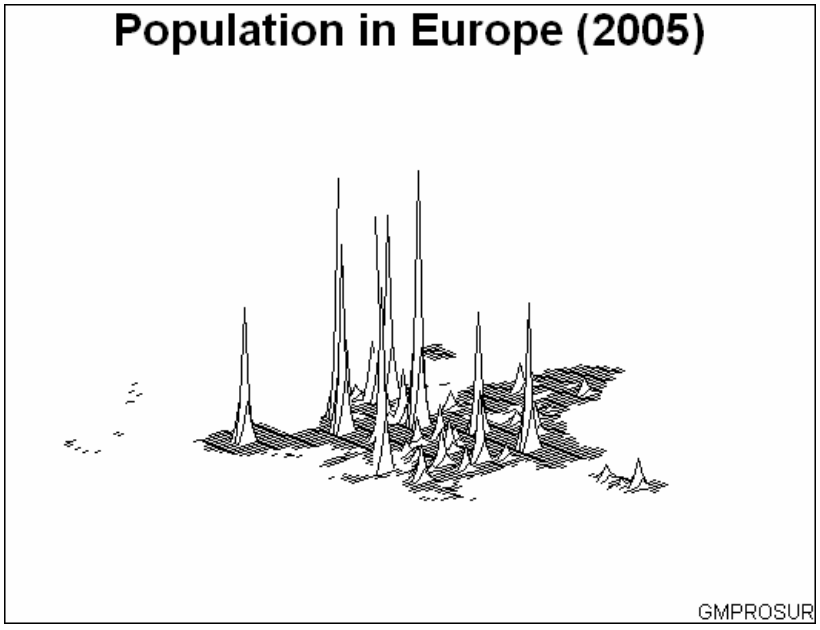
Example 10: Rotating and Tilting a Surface Map

Procedure features:

SURFACE statement options:

CONSTANT=
NLINES=
ROTATE=
TILT=

Sample library member: GMPROSUR



This example tilts and rotates the surface map and uses more lines to draw the surface.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

```
title1 "Population in Europe (2005)";
footnote1 j=r "GMPROSUR";
```

Produce the surface map. The CONSTANT= option specifies a value that is less than the default value so the spikes are narrower at the base. The NLINES= option specifies the maximum number of map lines, which gives the best map shape resolution. The ROTATE= and TILT= options adjust the map orientation to make the crowded spikes in the northeast portion of the map easier to distinguish.

```
proc gmap map=maps.europe data=sashelp.demographics;
   id id;
   surface pop / constant=4
                 nlines=100
                 rotate=40
                 tilt=60;
run;
quit;
```

Example 11: Creating a Map Using the Feature Table

Procedure Features:

ID statement

CHORO statement option:

DISCRETE

ODS Features:

ODS HTML statement:

BODY=

Other Features:

MERGE statement

GOPTIONS statement

Sample library member: GMPSPATL

When you use a feature table with the GMAP procedure, you must merge the feature table with your response data set before generating a map, storing the combined data in a new data set. On PROC GMAP, you use the DATA= option to name the combined data set, and you use the ID statement to identify the variable that contains the spatial information.

To illustrate the use of a feature table, assume you want to generate a map of the United States. Rather than using the traditional map data set MAPS.US, you want to use its corresponding feature table. To determine which feature table corresponds to a traditional map data set, look in the MAPS.METAMAPS data set:

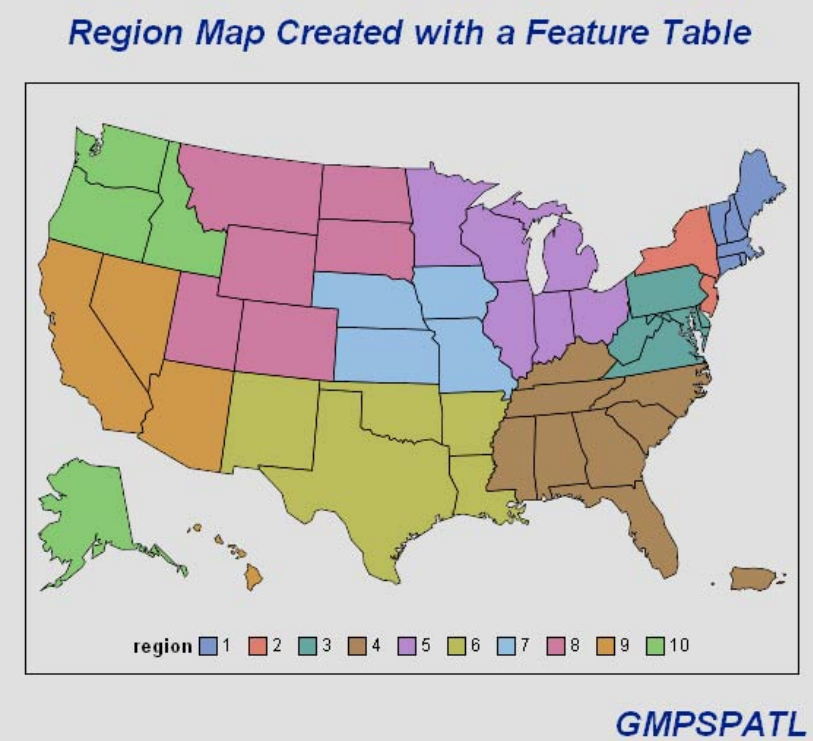
- The feature table MAPS.US2 corresponds to the traditional map data set MAPS.US.

- In MAPS.US2, the values of the variable _MAP_GEOMETRY_ encapsulate the geometry object.

The sample program uses the following procedures and statements:

- PROC SORT sorts WORK.SITES by the values of variable STATE. This prepares SITES for a merge with the feature table MAPS.US2. The variable STATE identifies the map areas in both SITES and MAPS.US2.
- PROC SORT sorts the feature table MAPS.US2. The OUT= option specifies that the sorted data be written to a new data set WORK.MYMAP.
- In the DATA step, the MERGE statement merges the feature table with the response data. The combined data set is saved to a new data set named BOTH. The data set BOTH is stored in WORK, a temporary library. To use the combined data set in other SAS/GRAPH programs, you would need to save the merged data set to a permanent library.
- On the PROC GMAP statement, the DATA= option points to the combined data set, BOTH. The ID statement specifies _MAP_GEOMETRY_ as the variable that contains the spatial data.

The following example creates the response data set SITES and merges it with the feature table US2. It then uses the combined data set to generate a map.



Create the data set SITES with regional data. Sites contains a region number for each state and the total number of hazardous waste sites in each state. The STFIPS function converts the state postal codes to FIPS state codes.

```
data sites;
   length stcode $ 2;
```

```
   input region stcode $ sites @@;
   state=stfips(stcode);
   datalines;
 6 AR  12 10 AK   7  4 AL  12  9 AZ  10
 9 CA  90  8 CO  15  1 CT  15  3 DE  18
 4 FL  52  4 GA  15  9 HI   4  7 IA  16
10 ID   8  5 IL  38  5 IN  30  7 KS  10
 4 KY  16  6 LA  15  1 MA  30  3 MD  13
 1 ME  12  5 MI  72  5 MN  30  7 MO  22
 4 MS   1  8 MT   8  4 NC  22  8 ND   0
 7 NE  10  1 NH  18  2 NJ 105  6 NM   9
 9 NV   1  2 NY  78  5 OH  34  6 OK  10
10 OR  10  3 PA 100  4 PR  56  1 RI  12
 4 SC  26  8 SD   2  4 TN  14  6 TX  26
 8 UT  12  3 VA  25  1 VT   8 10 WA  49
 5 WI  40  3 WV   6  8 WY   3
;
run;
```

Sort the response and the feature tables in the order of the BY variable. By default, the first PROC SORT sorts the response data set created in the code above. Both sorted data sets are stored in the SAS temporary library WORK. To enable the data sets to be merged, the same BY variable is used to sort both the response and feature tables.

```
proc sort data=sites out=sites;
   by state;
run;

proc sort data=maps.us2 out=mymap;
   by state;
run;
```

Merge the data sets.

```
data both;
   merge mymap sites;
   by state;
run;
```

Specify the ACTIVEX driver and HTML output. To conserve system resources, ODS LISTING CLOSE closes the Listing destination for procedure output. In the program's ODS HTML statement, the BODY= option names the file for storing HTML output.

```
goptions reset=all border;
ods listing close;
ods html body="hazmat_sites.html";
```

Define the title and footnote for the map.

```
title1 "Region Map Created with a Feature Table";
footnote1 j=r "GMPSPATL";
```

Generate the choropleth map using the merged response data set and feature table. The ID variable is the $GEOREF formatted variable that associates the feature table data with its map data set (MAPS.US). DISCRETE specifies that each level of REGION is a separate response level.

```
proc gmap data=both;
   id _map_geometry_;
   choro region/discrete;
run;
quit;
```

Close the HTML destination and open the listing destination. The HTML destination must be closed before you can view the output with a browser. ODS LISTING opens the Listing destination again so that the destination is again available for displaying output during this SAS session.

```
ods html close;
ods listing;
```

CHAPTER

44

The GOPTIONS Procedure

Overview

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 220 for a list of the graphics options that you can set with the GOPTIONS statement. See Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 for a complete description of each graphics option. △

The list of graphics options displays 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 44.1 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 44.1 Parital Output from the GOPTIONS Procedure

```
                SAS/GRAPH software options and parameters
            (executing in DMS Programming Environment environment)
NOADMGDF                    GDDM driver output an ADMGDF file
ASPECT=                     Aspect ratio (width/height) for software characters
NOAUTOCOPY                  Automatic hardcopy after display
NOAUTOFEED                  Automatic paper feed after plot
NOAUTOSIZE                  Change character cell size to preserve device
                            catalog rows and columns
BINDING=NOBINDING           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=                   Select 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 milliseconds
DEVADDR=                    IBM Device address, qname, or node name
DEVICE=                     Default device driver
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
FASTTEXT                    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 Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327. △

Procedure Syntax

PROC GOPTIONS <*option(s)*>;

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 <*option(s)*>;

option(s) can be one or more options from the following categories:

- item request options
  - AXIS
  - FOOTNOTE
  - LEGEND
  - OPTION=*graphics-option*
  - PATTERN
  - SYMBOL
  - TITLE
- listing format options
  - CENTIMETERS
  - NOLIST
  - NOLOG
  - SHORT

Options

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.

Note: The CENTIMETERS option does not affect the graphics options that can use unit specifications of CELLS, CM, IN, PCT, and PT. △

Alias: CM

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

Featured in: Example 1 on page 1322

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

Featured in: Example 1 on page 1322

NOLOG
displays the output in the OUTPUT window instead of the Log window.

OPTION=*graphics-option*
requests information on the specified graphics option. For these options, requesting the information on 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.

Featured in: Example 2 on page 1323

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

Example 1: Displaying TITLE and FOOTNOTE Statements

Procedure features:
PROC GOPTIONS statement:

FOOTNOTE
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 44.2 shows the listing that appears in the LOG window.

Output 44.2 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" ;
```

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

Procedure features:

PROC GOPTIONS statement:

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 44.3 shows the listing that appears in the Log window.

Output 44.3 Using the SHORT Option (GOPSHORT)

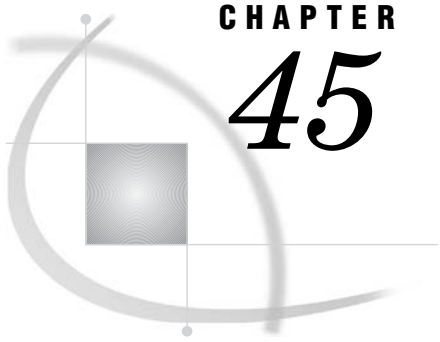
```
                          SAS/GRAPH software options and parameters
                           (executing in DMS Process environment)
NOACCESSIBLE NOADMGDF ALTDESC ASPECT= NOAUTOCOPY NOAUTOFEED AUTOSIZE=  BINDING=DEFAULTEDGE
NOBORDER CBACK= CBY= CELL CHARACTERS CHARTYPE= CIRCLEARC NOCOLLATE COLORS=( ) CPATTERN= CSYMBOL=
CTEXT= CTITLE= DASH DASHSCALE= DELAY= DEVADDR= DEVICE= DEVMAP=DEFAULT DISPLAY DISPOSAL=NONE
DRVINIT= DRVTERM= NODUPLEX NOERASE EXTENSION= FASTTEXT 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 NOV5COMP
NOV6COMP VORIGIN= VPOS= VSIZE= XMAX= XPIXELS= YMAX= YPIXELS=
```

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

45

The GPLOT Procedure

Overview

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 display 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 charts, 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:

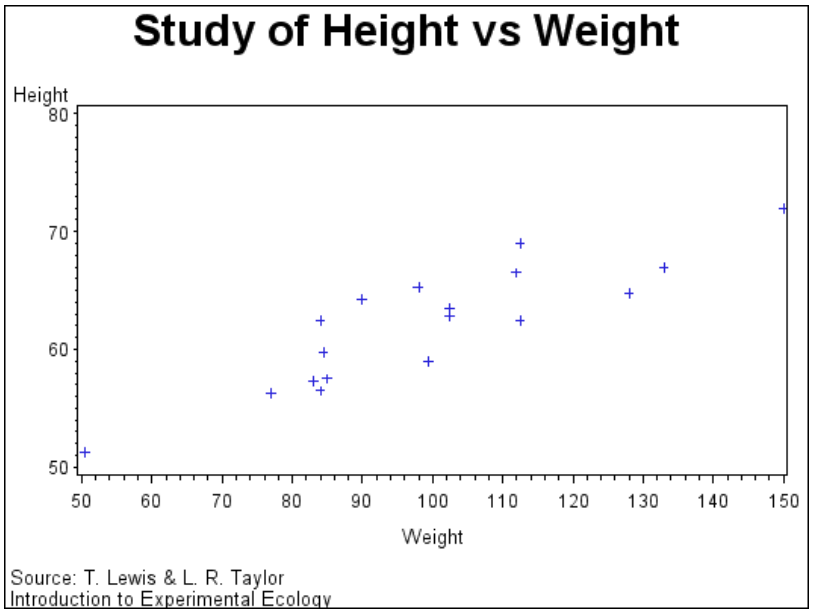
- displaying a long series of data, showing trends and patterns
- interpolating between data points
- extrapolating beyond existing data with the display of regression lines and confidence limits.

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 45.1 Scatter Plot of Two Variables (GPLVRBL1(a))



The program for this plot is in Example 4 on page 1372. For more information on producing scatter plots, see “PLOT Statement” on page 1347.

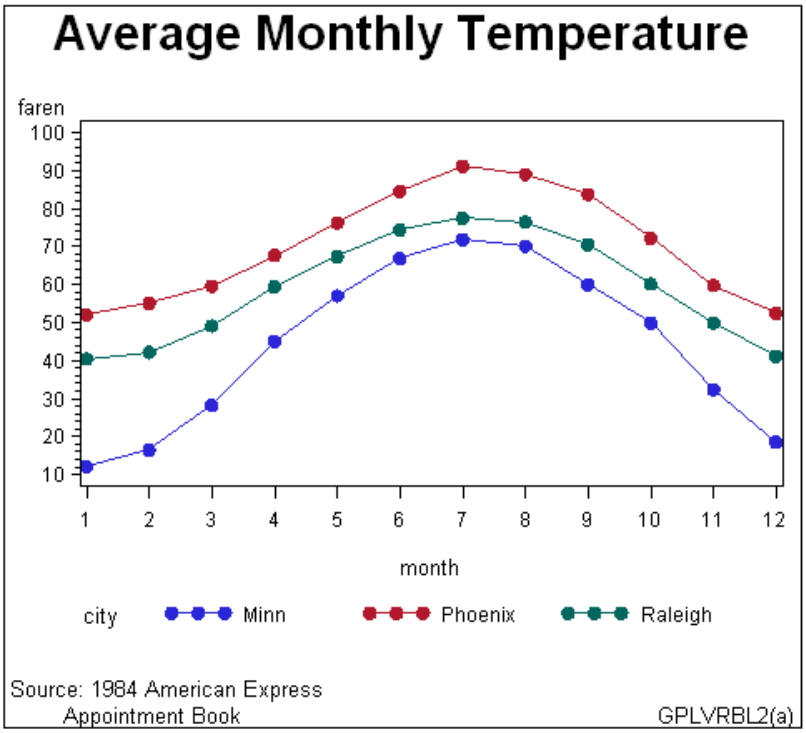
You can also overlay two or more plots (multiple sets of data points) on a single set of axes and you can apply a variety of interpolation techniques to these plots. See "About Interpolation Methods" on page 1329.

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 45.2 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 on page 1383, a SYMBOL statement connects the data points and specifies the plot symbol that is used for each value of the classification variable (CITY). For more information on how to produce plots with a classification variable, see "PLOT Statement" on page 1347.

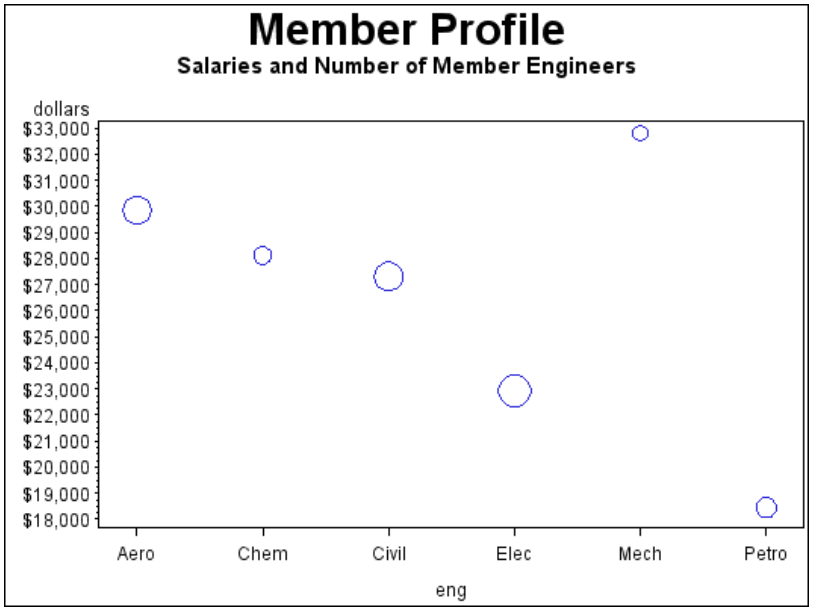
About Bubble Plots

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 45.3 on page 1328 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 45.3 Bubble Plot (GPLBUBL1)



The program for this plot is in Example 1 on page 1367. For more information on producing bubble plots, see "BUBBLE Statement" on page 1333.

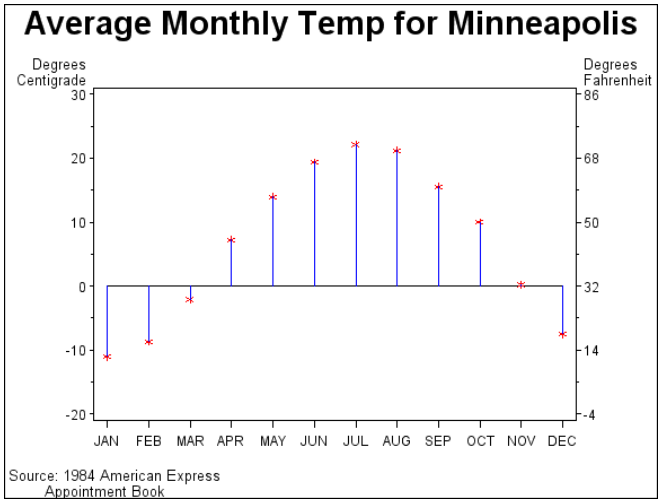
About Plots with Two Vertical Axes

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 45.4 Plot with a Right Vertical Axis (GPLSCVL1)



The program for this plot is in Example 9 on page 1386. For more information on how to produce plots with a right vertical axis, see "PLOT2 Statement" on page 1361 and "BUBBLE2 Statement" on page 1343.

About Interpolation Methods

In addition to these graphs, you can produce other types of plots such as box plots or high-low-close charts 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, 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

"SYMBOL Statement" on page 252 describes all interpolation methods.

Concepts

Parts of a Plot

Some terms used with GPLOT procedure are illustrated in Figure 45.5 on page 1330 and Figure 45.6 on page 1330.

Figure 45.5 GPLOT Procedure Terms

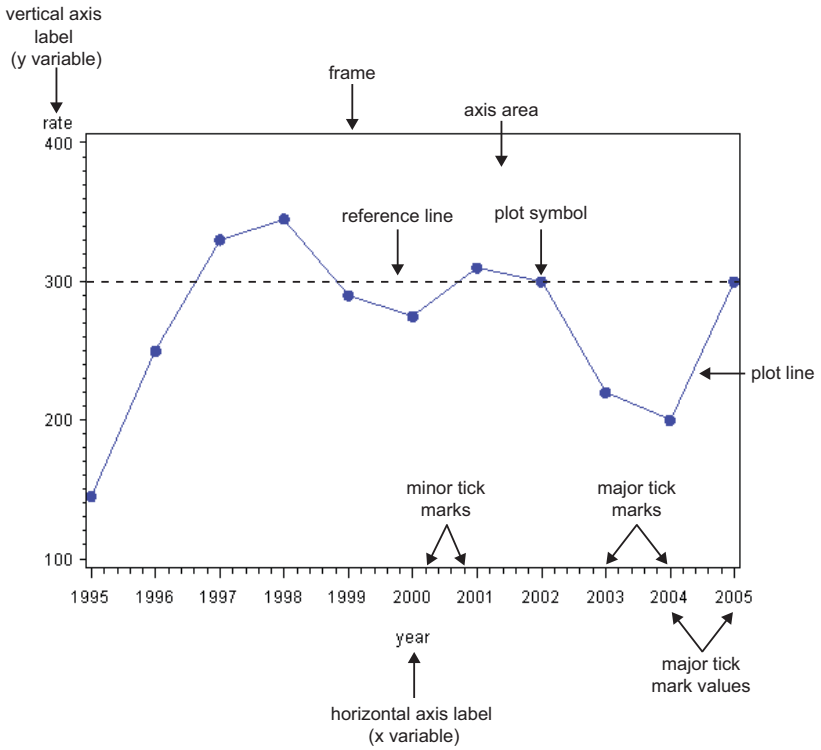
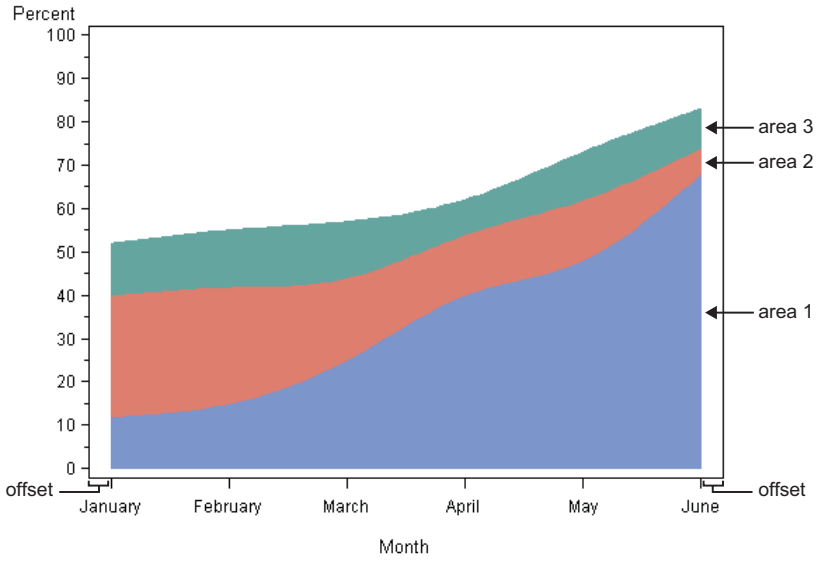


Figure 45.6 Additional GPLOT Procedure Terms



About the Input Data Set

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, but 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 interpolations.

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 and as a result might change interpolated values for the plot. 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 of the SYMBOL statement in "SYMBOL Statement" on page 252 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 "AXIS Statement" on page 198 for a complete discussion.

Procedure Syntax

Requirements: 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.

Global statements: AXIS, FOOTNOTE, LEGEND, PATTERN, TITLE

Reminder: The procedure can include BY, FORMAT, LABEL, WHERE, and NOTE statements.

Supports:
RUN-group processing

PROC GPLOT <DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set* >
<UNIFORM>;
BUBBLE *plot-request(s)* <*/option(s)*>;
BUBBLE2 *plot-request(s)* <*/option(s)*>;
PLOT *plot-request(s)* <*/option(s)*>;
PLOT2 *plot-request(s)* <*/option(s)*>;

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.

Requirements: An input data set is required.

Syntax

PROC GPLOT <DATA=*input-data-set*>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set* >
<UNIFORM>;

Options

ANNOTATE=*Annotate-data-set*
ANNO=*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.

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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 also: "SAS Data Sets" on page 54 and "About the Input Data Set" on page 1331.

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 also: "Specifying the Catalog Name and Entry Name for Your GRSEGs" on page 100.

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map in an HTML output file. The IMAGEMAP= option can be used only if the PLOT or PLOT2 statements are used, and the PLOT or PLOT2 statement must use the HTML= option or the HTML_LEGEND= option or both.

If the HTML= option is used in the PLOT or PLOT2 statement, the plot points are defined as hot zones, unless the AREA= option is also used. In that case there are not plot points and the areas between plot lines are defined as hot zones. If the HTML_LEGEND= option is used, the legend symbols are defined as hot zones. Information for the links is stored in the variables referenced by the HTML= or HTML_LEGEND= options or both.

The %IMAGEMAP macro generates the image map in the HTML output file. The macro takes two arguments, the name of the image map data set and the name or fileref of the HTML output file, as shown in the following example:

```
%imagemap(imgmapds, myimgmap.html);
```

UNIFORM
specifies that the same axis scaling is used for all graphs that are produced by the procedure. By default, the range of axis values for each axis is based on the minimum and maximum values in the data and, therefore, 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. Thus, if the procedure 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, 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.

Requirements: At least one plot request is required.

Global statements: AXIS, FOOTNOTE, TITLE

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, modify the appearance of axes, control the display of the bubbles, specify a backplane 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.

Syntax

BUBBLE *plot-request(s)* </*option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- bubble appearance options:
  - BCOLOR=*bubble-color*
  - BFILL=SOLID | GRADIENT
  - BFONT=*font*
  - BLABEL
  - BSCALE=AREA | RADIUS
  - BSIZE=*multiplier*
- plot appearance options:
  - ANNOTATE=*Annotate-data-set*
  - CAXIS=*axis-color*
  - CFRAME=*background-color*
  - CTEXT=*text-color*
  - FRAME | NOFRAME
  - FRONTREF
  - GRID
  - IFRAME= *fileref* | '*external-file*'
  - IMAGESTYLE = TILE | FIT
  - NOAXIS
- horizontal axis options:
  - AUTOHREF
  - CAUTOHREF=*reference-line-color*
  - CHREF=*reference-line-color* | (*reference-line-color*) | *reference-line-color-list*
  - HAXIS=*value-list* | AXIS<1...99>
  - HMINOR=*number-of-minor-ticks*
  - HREF=*value-list*
  - HREVERSE

HZERO
LAUTOHREF=*reference-line-type*
LHREF=*reference-line-type* | (*reference-line-type*) | *reference-line-type-list*

- vertical axis options:
  AUTOVREF
  CAUTOVREF=*reference-line-color*
  CVREF=*reference-line-color* | (*reference-line-color*) | *reference-line-color-list*
  LAUTOVREF=*reference-line-type*
  LVREF=*reference-line-type* | (*reference-line-type*) | *reference-line-type-list*
  VAXIS=*value-list* | AXIS<1...99>
  VMINOR=*number-of-minor-ticks*
  VREF=*value-list*
  VREVERSE
  VZERO
  WAUTOHREF=
  WHREF=
- catalog entry description options:
  DESCRIPTION='*entry-description*'
  NAME='*entry-name*'

Required Arguments

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.

Note: If you specify the JAVA, JAVAMETA, 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. △

Options

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 plots that are produced by the BUBBLE statement.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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. If you do not specify the BCOLOR= option, then the bubble color becomes the color of the default style (GSTYLE) or the color specified by the current ODS style (if used).

Featured in: Example 2 on page 1368 and Example 3 on page 1370.

Style reference: ContrastColor attribute of the GraphOutline, GraphData1, and TwoColorAltRamp elements.

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 background 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.

Note: The SAS/GRAPH ActiveX control displays negative values as empty circles. △

Restriction: Not supported by Java and ActiveX

BFONT=*font*
specifies the font to use for bubble labels. See Chapter 11, "Specifying Fonts in SAS/GRAPH Programs," on page 155 for details on 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

Featured in: Example 2 on page 1368.

Style reference: Font attribute of the GraphValueText element

Restriction: Not supported by Java and ActiveX

See also: The BLABEL option for information on the location and color of labels

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 display in the color specified by the CTEXT= option. If you omit the CTEXT=option, the default is the color of the current style.

Featured in: Example 2 on page 1368

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. By default, BSIZE=5. If you specify BSIZE=0, then the default size is used instead.

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.

Featured in: Example 2 on page 1368

Restriction: Partially supported by Java and ActiveX

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*)
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 a single color without parentheses applies that color to all reference lines. The CAUTOHREF= option overrides the CHREF= option for lines drawn with the AUTOHREF option. Specifying a single color in parentheses 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 is of the form (*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

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, 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 other devices.

The COLOR= suboption of a LABEL= option in an AXIS definition overrides the CTEXT= option and determines the color of the axis label. Likewise, the COLOR= suboption of a VALUE= option in an AXIS definition overrides the CTEXT= option and determines the color of the tick marks.

or VALUE= option in an AXIS definition, then that COLOR= suboption determines the color of the axis label or the color of the tick mark values, respectively.

Alias: C=

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 a single color without parentheses applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying a single color in parentheses 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 is of the form (*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.

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 the description of the catalog entry for the plot. The maximum length for *entry-description* is 256 characters. The description does not appear on the plot. By default, the procedure assigns a description of the form BUBBLE OF *variable*variable=variable*.

The *entry-description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. For more information, refer to the discussion of "Substituting BY Line Values in a Text String" on page 294. 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.

The descriptive text is shown in each of the following locations:

- in the Results window.
- among the catalog-entry properties that you can view from the Explorer window
- in the Description field of the PROC GREPLAY window.
- the data tip text for Web output (depending on the device driver you are using). See "Data Tips for Web Presentations" on page 598 for details.

Alias: DES=

FRAME | NOFRAME
specifies whether a frame is drawn around the axis area. The default is FRAME. If you also use a BUBBLE2 or PLOT2 statement and your plotting statements have conflicting frame specifications, FRAME is used.

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.

To fill the axis area with a background color, use the CFRAME= option.

To fill the axis area with a background image, use the IFRAME= option.

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.

Alias: FR|NOFR=

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. For a description of *value-list*, see the HAXIS= option on page 1354 for the PLOT statement. To assign labels to horizontal reference lines, specify an axis definition that contains the REFLABEL= option. 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. By default, the procedure scales the axis and provides an appropriate number of tick marks.

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 "AXIS Statement" on page 198.

Featured in: Example 2 on page 1368

Restriction: Partially supported by Java and ActiveX

See also: "About the Input Data Set" on page 1331 for more information on values out of range

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=

Featured in: Example 2 on page 1368

HREF=*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* HAXIS= option on page 1354 HAXIS for the PLOT statement. LHREF=, CHREF=, and WHREF= 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.

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

HZERO
specifies that tick marks on the horizontal axis begin in the first position with a value of zero. The HZERO request is ignored if negative values are present for the horizontal variable or if the horizontal axis has been specified with the HAXIS= option.

IFRAME=*fileref* | '*external-file*'
identifies the image file you want to apply to the backplane of the plot. See also the IMAGESTYLE= option and "Displaying Images on Data Elements" on page 185. The IFRAME= option is overridden by the NOIMAGEPRINT goption. For more information about the NOIMAGEPRINT option, see "IMAGEPRINT" on page 387.

Restriction: Not supported by Java

IMAGESTYLE= TILE | FIT
specifies whether to tile multiple instances of the image to fill the backplane (TILE) or to stretch the image to fit the backplane frame (FIT). The TILE value is the default. For more information see the IFRAME= option.

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, 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

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 a single line type without parentheses applies that line type to all reference lines. The LAUTOHREF= option overrides the LHREF= option for lines drawn with the AUTOHREF option. Specifying a single line type in parentheses 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 is of the form (*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.

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 a single line type without parentheses applies that line type to all reference lines. The LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option. Specifying a single line type in parentheses 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 is of the form (*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.

Alias: LV=

Style reference: LineStyle attribute of the GraphGridLines element

NAME='*entry-name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default name is GPLOT. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name-for example, GPLOT1.

See also: "About Filename Indexing" on page 99

NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias: NOAXES

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 the HAXIS= option on page 1354. 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 "AXIS Statement" on page 198.

Featured in: Example 2 on page 1368 and Example 3 on page 1370

Restriction: Partially supported by Java and ActiveX

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=

Featured in: Example 2 on page 1368

VREF=*value-list*
draws one or more reference lines perpendicular to the vertical axis at points that are specified by *value-list*. For a description of the *value-list*, see the HAXIS= option on page 1354. 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.

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 with a zero. The VZERO request is ignored if the vertical variable either contains negative values or has been ordered with the VAXIS= option or the ORDER= option in an AXIS statement.

WAUTOHREF=*N*
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 CAUTOREF= option.

Style reference: LineThickness attribute of the GraphGridLines element

WAUTOVREF=*N*
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 CAUTOREF= option.

Style reference: LineThickness attribute of the GraphGridLines element

WHREF=*N*
specifies line widths for reference lines as determined by the horizontal axis. Line widths are specified as whole numbers. To specify colors for these reference lines, use the CREF= option.

Style reference: LineThickness attribute of the GraphGridLines element

WVREF=*N*
specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. To specify colors for these reference lines, use the CREF= option.

Style reference: LineThickness attribute of the GraphGridLines element

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.

Requirements: 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.

Global statements: AXIS, FOOTNOTE, TITLE

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, control the display of the bubbles, display a background 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.

Syntax

BUBBLE2 *plot-request(s)* </*option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- bubble appearance options:
  - BCOLOR=*bubble-color*
  - BFILL=SOLID | GRADIENT
  - BFONT=*font*
  - BLABEL
  - BSCALE=AREA | RADIUS
  - BSIZE=*multiplier*
- plot appearance options:
  - ANNOTATE=*Annotate-data-set*
  - CAXIS=*axis-color*
  - CFRAME=*background-color*
  - CTEXT=*text-color*
  - FRAME | NOFRAME
  - GRID
  - NOAXIS | NOAXES
- vertical axis options:
  - AUTOVREF
  - CAUTOVREF=*reference-line-color*
  - CVREF=*reference-line-color* | (*reference-line-color*) | *reference-line-color-list*
  - LAUTOVREF=*reference-line-type*
  - LVREF=*reference-line-type* | (*reference-line-type*) | *reference-line-type-list*
  - VAXIS=*value-list* | AXIS<1...99>
  - VMINOR=*number-of-minor ticks*
  - VREF=*value-list*
  - VREVERSE
  - VZERO

Required Arguments

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.

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 “BUBBLE Statement” on page 1333 for complete descriptions of options used with the BUBBLE2 statement.

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 on 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 each with two vertical axes. The first pair of plot requests (Y*X=S and Y2*X=S) produce one graph in which 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 which 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.

If the scale of values for the left and right vertical axes is the same and you want both axes to represent the same range of values, 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.

Requirements: At least one plot request is required.

Global statements: AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE

Supports: Drill-down functionality

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 on the SYMBOL statement, see "About SYMBOL Definitions" on page 1360.

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.

Syntax

PLOT *plot-request(s)* </*option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- plot options:

  AREAS=*n*
  GRID
  LEGEND | LEGEND=LEGEND<1...99>
  NOLEGEND
  OVERLAY
  REGEQN
  SKIPMISS

- appearance options:

  ANNOTATE=*Annotate-data-set*
  CAXIS=*axis-color*
  CFRAME=*background-color*
  COUTLINE=*outline-color*
  CTEXT=*text-color*

FRAME | NOFRAME
FRONTREF
IFRAME= *fileref* | *'external-file'*
IMAGESTYLE = TILE | FIT
NOAXIS | NOAXES

- horizontal axis options:
  AUTOHREF
  CAUTOHREF=*reference-line-color*
  CHREF=*reference-line-color* | (*reference-line-color*) | *reference-line-color-list*
  HAXIS=*value-list* | AXIS<1...99>
  HMINOR=*number-of-minor-ticks*
  HREF=*value-list*
  HREVERSE
  HZERO
  LAUTOHREF=*reference-line-type*
  LHREF=*reference-line-type* | (*reference-line-type*) | *reference-line-type-list*
- vertical axis options:
  AUTOVREF
  CAUTOVREF=*reference-line-color*
  CVREF=*reference-line-color* | (*reference-line-color*) | *reference-line-color-list*
  LAUTOVREF=*reference-line-type*
  LVREF=*reference-line-type* | (*reference-line-type*) | *reference-line-type-list*
  VAXIS=*value-list* | AXIS<1...99>
  VMINOR=*number-of-minor-ticks*
  VREF=*value-list*
  VREVERSE
  VZERO
  WAUTOVREF
  WVREF
- catalog entry description options:
  DESCRIPTION='*entry-description*'
  NAME='*entry-name*'
- ODS options:
  HTML=*variable*
  HTML_LEGEND=*variable*

Required Arguments

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 1361. △

(*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;* that is, all plots are drawn on the same set of axes, and a legend is automatically generated to show the plot symbol and color for each value of the classification variable.

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. △
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 on page 1383.

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;
```

Options

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 plots that are produced by the PLOT statement.

Alias: ANNO=

See also: Chapter 29, "Using Annotate Data Sets," on page 641.

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.

If you specify a value for the AREAS= option that is greater than the number of bounded areas in the plot, 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 "SYMBOL Statement" on page 252 for details on 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 "PATTERN Statement" on page 240 for more information on map/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 because the INTERPOL= setting does not automatically apply to multiple symbol definitions.

Note: 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 248 for more information. △

You can alter the default pattern behavior by specifying patterns and colors on 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 "PATTERN Statement" on page 240 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, when creating multiple plots in combination with the OVERLAY option, the PLOT statements must be ordered 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 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. Therefore, the data must be recoded so that the lowest value of the third variable produces the lowest plot line, the next lowest value produces the next lowest plot line, and so on.

The AREAS= option works only if all plot lines are generated by the same PLOT or PLOT2 statement.

If you use the VALUE= option in the SYMBOL statement, some symbols might be hidden. If reference lines are also specified with the AREAS= option, they are drawn behind the pattern fill.

Featured in: Example 7 on page 1380.

Restriction: Partially supported by Java

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 CAUVTOREF 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=

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 later in the FRAME option description. If the IFRAME= option is in effect, an image appears in the background instead of the 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 a single color without parentheses applies that color to all reference lines. The CAUTOHREF= option overrides the CHREF= option for reference lines drawn with the AUTOHREF option. Specifying a single color in parentheses 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 is of the form (*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 of 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 the INTERPOL=map/ plot-pattern option or the GOPTIONS statement specifies the INTERPOL= option"INTERPOL" on page 389. The default outline color is specified in the current style. However, if the NOGSTYLE option is specified, then the default color is the first color in the device's color list (the foreground color), and 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.

The COUTLINE= option overrides the SYMBOL statement option CO=.

Restriction: Not supported by Java

Style reference: Color attribute of the GraphOutlines element

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, 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, 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 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.

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. △

Alias: C=

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 a single color without parentheses applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying a single color in parentheses 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 is of the form (*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 of 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 the description of the catalog entry for the plot. The maximum length for *entry-description* is 256 characters. The description does not appear on the plot. By default, the procedure assigns a description of the form PLOT OF *variable*variable=variable*.

The *entry-description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. For more information, refer to the discussion of "Substituting BY Line Values in a Text String" on page 294. 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.

The descriptive text is shown in each of the following locations:

- in the Results window.
- among the catalog-entry properties that you can view from the Explorer window.
- in the Description field of the PROC GREPLAY window.
- the data tip text for Web output (depending on the device driver you are using). See "Data Tips for Web Presentations" on page 598 for details.

Alias: DES=

FRAME | NOFRAME
specifies whether a frame is drawn around the axis area. The default is FRAME. If you also use a BUBBLE2 or PLOT2 statement and your plotting statements have conflicting frame specifications, FRAME is used.

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.

To fill the axis area with a background color, use the CFRAME= option.

To fill the axis area with a background image, use the IFRAME= option.

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.

Alias: FR|NOFR=

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 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 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.

For a complete description of *value-list*, see the ORDER= option on page 205 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 1331 for more information on 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 "AXIS Statement" on page 198.

Featured in: Example 4 on page 1372, Example 5 on page 1375, and Example 9 on page 1386

Restriction: Partially supported by Java and ActiveX

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=

Featured in: Example 2 on page 1368

HREF=*value-list*
draws one or more reference lines perpendicular to the horizontal axis at points that are specified by *value-list*. See the HAXIS= option for a description of *value-list*. 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. LHREF=, CHREF=, and WHREF= 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.

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 in the HTML output file that is generated by ODS. These links are associated with the plot points, or if the AREA= option is used, with the areas between plot lines. The links point to the data or graph that you want to display when the user drills down on the plot point or area. There is no limit on the length of the variable.

Restriction: Partially supported by Java and ActiveX for the PLOT statement and not supported by Java and ActiveX for the PLOT2 statement.

See also: "Overview of Enhancing Web Presentations" on page 596.

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values are used to create links in the HTML output file that is generated by ODS. When the HTML output file is displayed in a Web browser, clicking on an element in the legend displays the URL that was specified for that legend element, based on the value of the variable that is named as the value of the HTML_LEGEND option. The maximum length for the value of this variable is 1024 characters. To see an example that generates a drill-down graph using ODS, see Example 10 on page 1389.

Restriction: Not supported by Java and ActiveX.

See also: "Overview of Enhancing Web Presentations" on page 596.

HZERO
specifies that tick marks on the horizontal axis begin in the first position with a value of zero. The HZERO request is ignored if negative values are present for the horizontal variable or if the horizontal axis has been specified with the HAXIS= option.

IFRAME=*fileref* | '*external-file*'
identifies the image file you want to apply to the backplane frame of the plot. See also the IMAGESTYLE= option and "Displaying an Image in Graph Frame" on page 184. The IFRAME= option is overridden by the NOIMAGEPRINT goption. For more information about the NOIMAGEPRINT option, see "IMAGEPRINT" on page 387.

For Web output that is generated with the ACTIVEX or ACTXIMG device drivers,

Restriction: Not supported by Java

IMAGESTYLE= TILE | FIT
specifies whether to tile multiple instances of the image to fill the backplane frame (TILE) or to stretch a single instance of the image to fill the backplane frame (FIT). The TILE value is the default. See also the IFRAME= option.

Restriction: Not supported by Java

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 legend to use for the plot.

- 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*n* 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 and specify the list of variables in quotes 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;
```

- 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*n* to assign a defined LEGEND statement to the plot.

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.

Featured in: Example 6 on page 1377

See also: "LEGEND Statement" on page 225

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 a

single line type without parentheses applies that line type to all reference lines. The LAUTOHREF= option overrides the LHREF= option for lines drawn with the AUTOHREF option. Specifying a single line type in parentheses 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 is of the form (*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 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 a single line type without parentheses applies that line type to all reference lines. The LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option. Specifying a single line type in parentheses 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 is of the form (*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.

For needle plots that are generated with a Java or ActiveX device driver, the value of the LVREF= option is not applied to the default reference line that is drawn at zero when the minimum value of the vertical axis is less than zero. This line is solid (not dashed).

Alias: LV=

Featured in: Example 5 on page 1375

Style reference: LineStyle attribute of the GraphGridLines element

Restriction: Partially supported by Java and ActiveX

NAME='*entry-name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default name is GPLOT. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name-for example, GPLOT1.

See also: "About Filename Indexing" on page 99

NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias: NOAXES

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, and 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 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 JAVA, JAVAMETA, or JAVAIMG device drivers, the OVERLAY option cannot be used in the PLOTor PLOT2 statement under these conditions:

- if the PLOT or PLOT2 statement is combined with the global SYMBOL statement when the SYMBOL statement uses the INTERPOL= BOX, HILO, or STD.
- or for JAVA output using the PLOT2 statement, in a SYMBOL statement when the SYMBOL statement uses theINTERPOL= BOX, HILO, or STD, with or without the OVERLAY option.

Featured in: Example 6 on page 1377 and Example 7 on page 1380

Restriction: Partially supported by Java

REGEQN
displays the regression equation that is specified in the INTERPOL= option of the SYMBOL statement in the lower left hand 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 display if the chart area for the plot becomes so small that markers cannot be drawn because there are no coordinates from which to build the regression equation. In such cases, the regression equation is no longer meaningful.

Featured in: Example 4 on page 1372

Restriction: Not supported by ActiveX

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 also: "About the Input Data Set" on page 1331

VAXIS=*value-list* | AXIS<1...99>
specifies the major tick mark values for the vertical axis or assigns an axis definition. See the HAXIS= option for a description of the *value-list*. 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 "AXIS Statement" on page 198.

Featured in: Example 4 on page 1372 and Example 5 on page 1375

Restriction: Partially supported by Java and ActiveX

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=

Featured in: Example 2 on page 1368

VREF=*value-list*
draws one or more reference lines perpendicular to the vertical axis at points that are specified by the *value-list*. See the HAXIS= option for a description of the *value-list*. 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.

Featured in: Example 5 on page 1375.

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 with a zero. The VZERO request is ignored if the vertical variable either contains negative values or has been ordered with the VAXIS= option or the ORDER= option in an AXIS statement.

WAUTOHREF=*reference-line-type*
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 CAUTOREF= option.

Style reference: LineThickness attribute of the GraphGridLines element

WAUTOVREF=*value-list*
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 CAUTOREF= option.

Style reference: LineThickness attribute of the GraphGridLines element

WHREF=*value-list*
specifies line widths for reference lines as determined by the horizontal axis. Line widths are specified as whole numbers. To specify a color for these reference lines, use the CAUTOREF= option.

Style reference: LineThickness attribute of the GraphGridLines element

WVREF=*value-list*
specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. To specify a color for these reference lines, use the CAUTOREF= option.

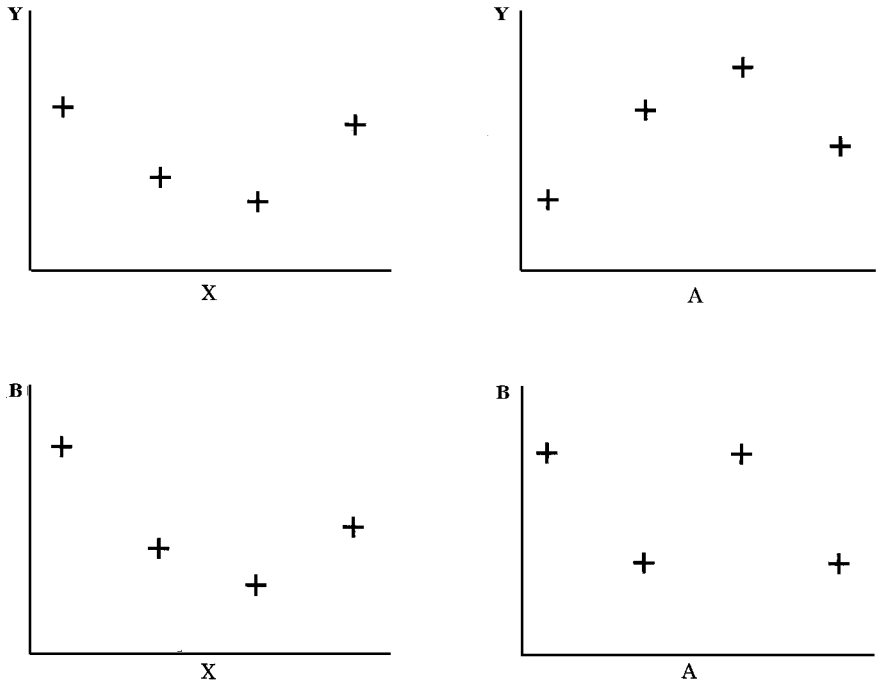
Style reference: LineThickness attribute of the GraphGridLines element

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 45.7 on page 1360.

```
plot (y b)*(x a);
```

Figure 45.7 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, or if the NOGSTYLE option is specified, 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 “SYMBOL Statement” on page 252.

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. See “SYMBOL Statement” on page 252 for more information on the SYMBOL statement.

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.

Requirements: 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.

Global statements: AXIS, FOOTNOTE , LEGEND , PATTERN, SYMBOL, TITLE

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 on the SYMBOL statement see “About SYMBOL Definitions” on page 1360.

Note: When using the PLOT2 statement to generate output with the Java or ACTIVEX device drivers, and when the global statement SYMBOL is used, 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.

Syntax

PLOT2 *plot-request(s)* </*option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- plot options:

AREAS=*n*
GRID
LEGEND | LEGEND=LEGEND<1...99>
NOLEGEND
OVERLAY
REGEQN
SKIPMISS

- appearance options:
  ANNOTATE=*Annotate-data-set*
  CAXIS=*axis-color*
  CFRAME=*background-color*
  COUTLINE=*outline-color*
  CTEXT=*text-color*
  FRAME | NOFRAME
  NOAXIS | NOAXES
- vertical axis options:
  AUTOVREF
  CAUTOVREF=*reference-line-color*
  CVREF=*reference-line-color* | (*reference-line-color*) | *reference-line-color-list*
  LAUTOVREF=*reference-line-type*
  LVREF=*reference-line-type* | (*reference-line-type*) | *reference-line-type-list*
  VAXIS=*value-list* | AXIS<1...99>
  VMINOR=*n*
  VREF=*value-list*
  VREVERSE
  VZERO
- ODS options:
  HTML=*variable*
  HTML_LEGEND=*variable*

Required Arguments

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 1347.

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.

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 1347 for descriptions of options that you can use with the PLOT2 statement.

Matching Plot Requests

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;
```

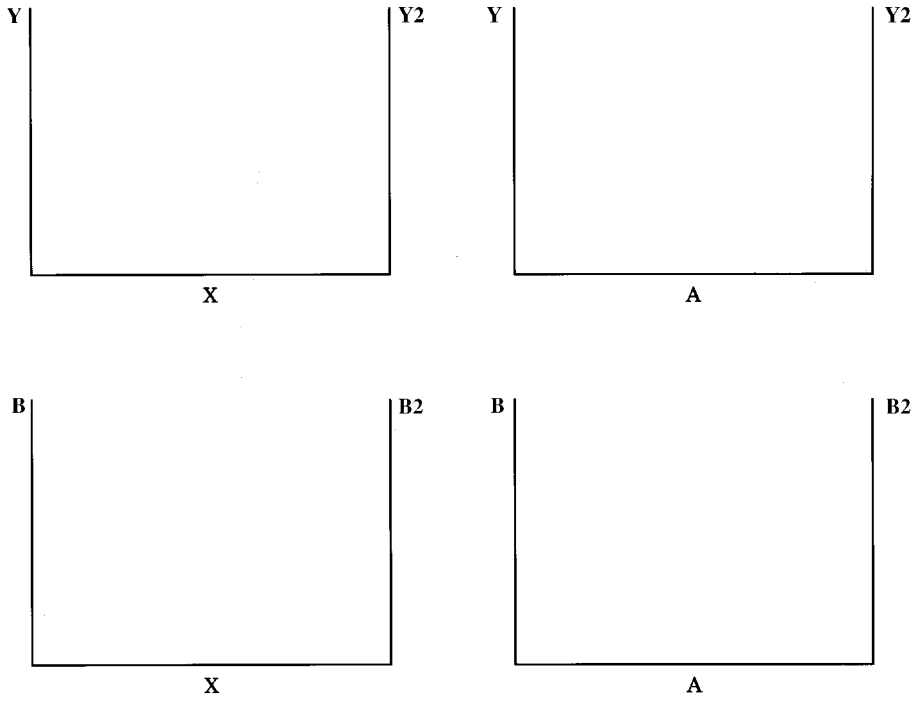
These statements produce two graphs, each with two vertical axes. The first pair of plot requests (Y*X and Y2*X) produce one graph in which X is plotted on the horizontal axis, Y is plotted on the left axis, and Y2 is plotted on the right axis. The second pair of plot requests (B*A and B2*A) produce another graph in which 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 45.8 on page 1364.

```
plot (y b)*(x a);
    plot2 (y2 b2)*(x a);
```

Figure 45.8 Graphs Produced by Multiple Plot Requests in PLOT and PLOT2 Statements

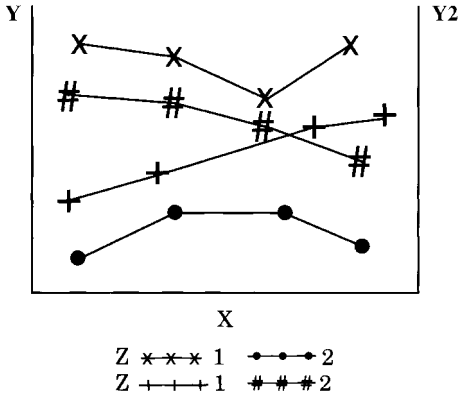


Requesting Plots of Three Variables with a Legend

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 45.9 on page 1365. 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 45.9 Multiple Plots on One Graph



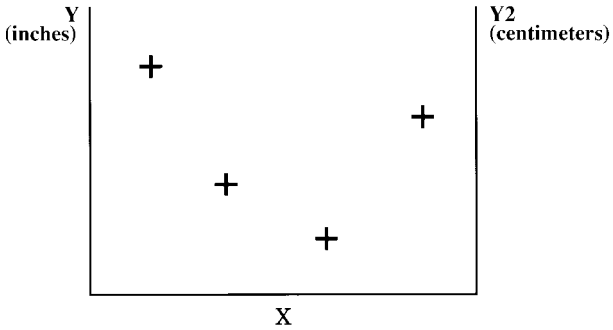
Using a Second Vertical Axis

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. If both vertical axes are calibrated so that they represent the same range of values, then for each observation of X the data points for Y and Y2 are the same.

For example, if Y is height in inches and Y2 is height in centimeters and 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 like Figure 45.10 on page 1365.

Figure 45.10 Right Axis with Different Scale of Values



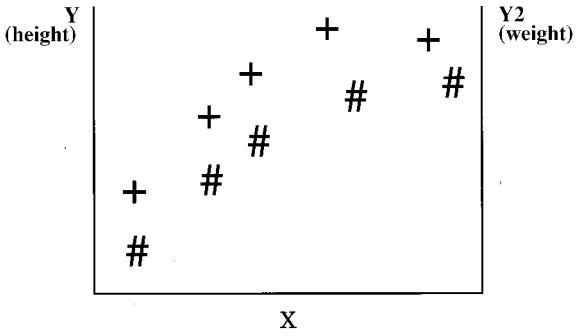
For these types of plots, the PLOT2 statement should use a SYMBOL statement that specifies INTERPOL=NONE and VALUE=NONE.

Displaying Different Values

If your data contain variables with different data values (such as height and weight), 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 45.11 on page 1366.

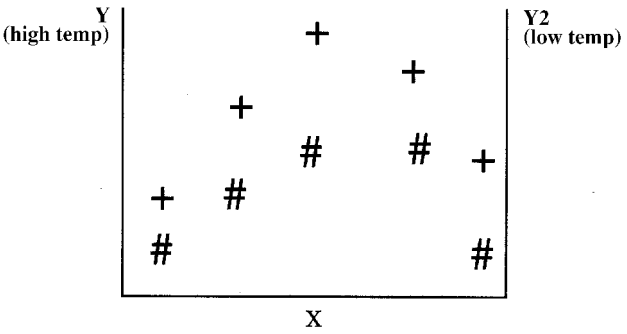
Figure 45.11 Right Axis with Different Values and Different Scale



Displaying the Same Scale on Both Axes

If your data contain two sets of values for the same type of data, you can use the PLOT2 statement to generate a right axis that is calibrated the same as the left axis so that the data points on the right of the graph are easier to read. For example, if Y is high temperatures and Y2 is low temperatures, you can create a graph like Figure 45.12 on page 1366.

Figure 45.12 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 1360.

Examples

Note: 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 HTML; procedure-code; ODS HTML CLOSE;`**). See "RUN-Group Processing" on page 56 for more information. △

Example 1: Generating a Simple Bubble Plot

Procedure features:

BUBBLE statement option:

HAXIS=

Other features:

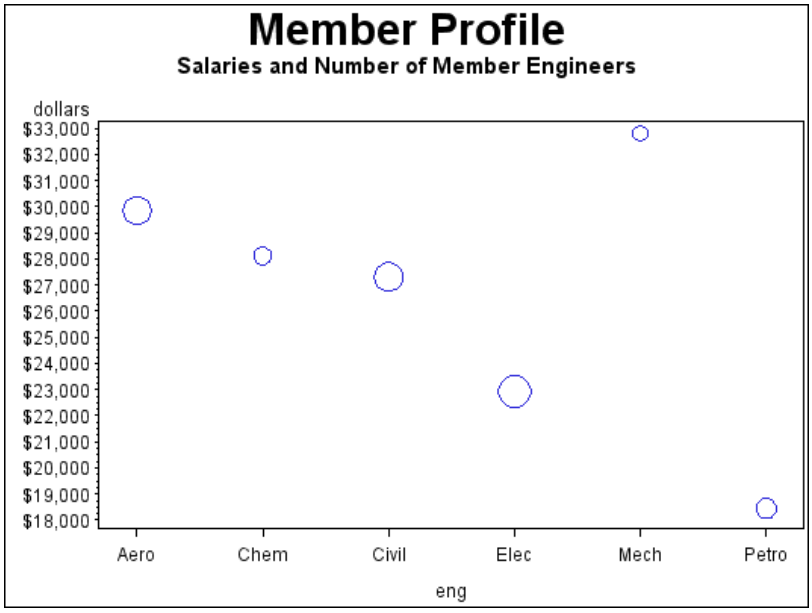
GOPTIONS statement option:

BORDER

AXIS statement

FORMAT statement

Sample library member: GPLBUBL1



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.

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 and footnote.

```
title1 "Member Profile"
title2 "Salaries and Number of Member Engineers";
footnote j=r "GPLBUBL1";
```

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

Procedure features:

BUBBLE statement options:

- BCOLOR
- BLABEL
- BSIZE
- HAXIS=
- VAXIS=
- VMINOR

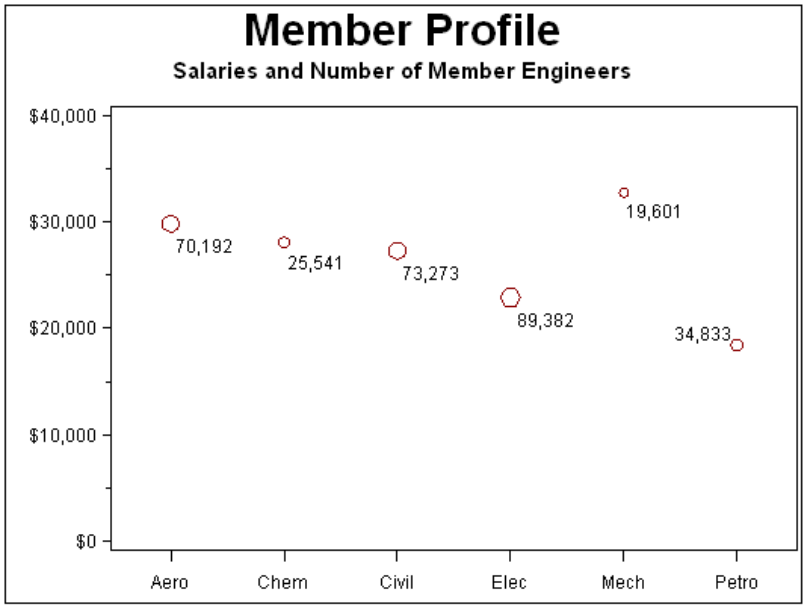
Other features:

GOPTIONS statement option:

- BORDER

AXIS statement

Sample library member: GPLBUBL2



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.

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 and footnote.

```
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 so bubbles are not clipped. AXIS2 uses the ORDER= option to set major tick mark intervals. This could be done with the VAXIS= option on 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. Thne 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

Procedure features:

BUBBLE statement options:

VAXIS=
HAXIS=
HMINOR=
VMINOR=
BLABEL

Other features:

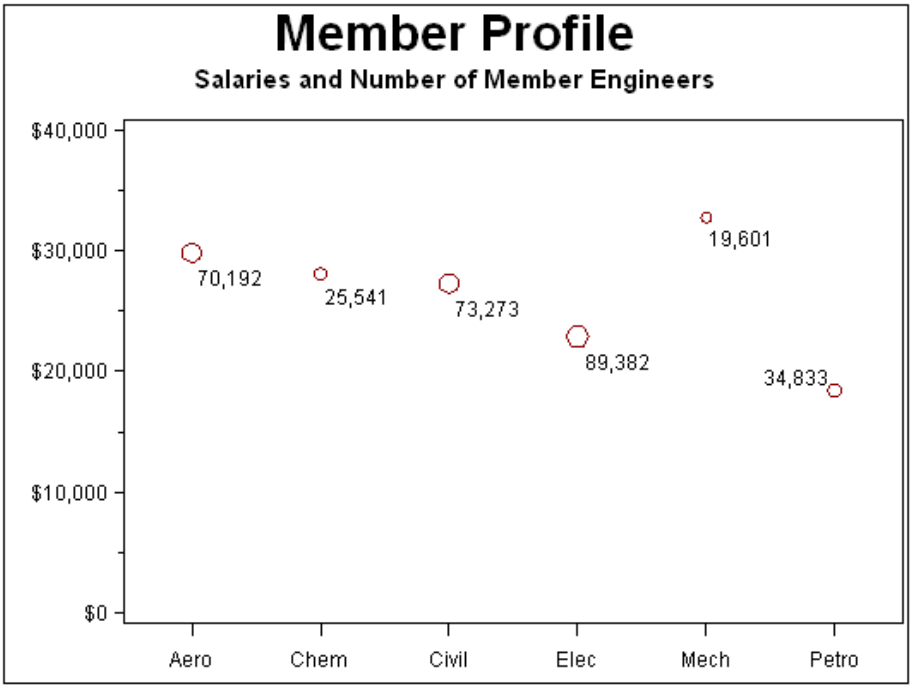
AXIS statement

FORMAT statement

GOPTIONS statement option:

BORDER

Sample library member: GPLAXIS1



This example modifies Example 2 on page 1368 to show how a BUBBLE2 statement generates a right vertical axis that 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. △

Set the graphics environment.

```
goptions reset=all border;
```

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 and footnote.

```
title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";
footnote j=r "GPLAXIS1 ";
```

Define horizontal-axis characteristics.

```
axis1 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

Procedure features:

PLOT statement options:

HAXIS=
HMINOR=
REGEQN
VAXIS=

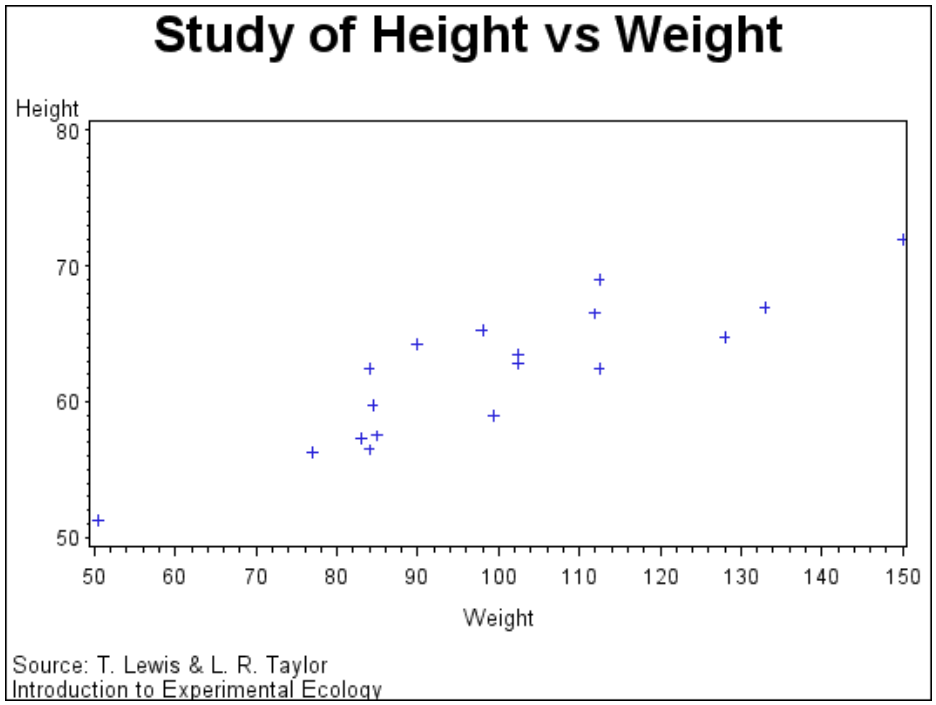
Other features:

GOPTIONS statement option:

BORDER

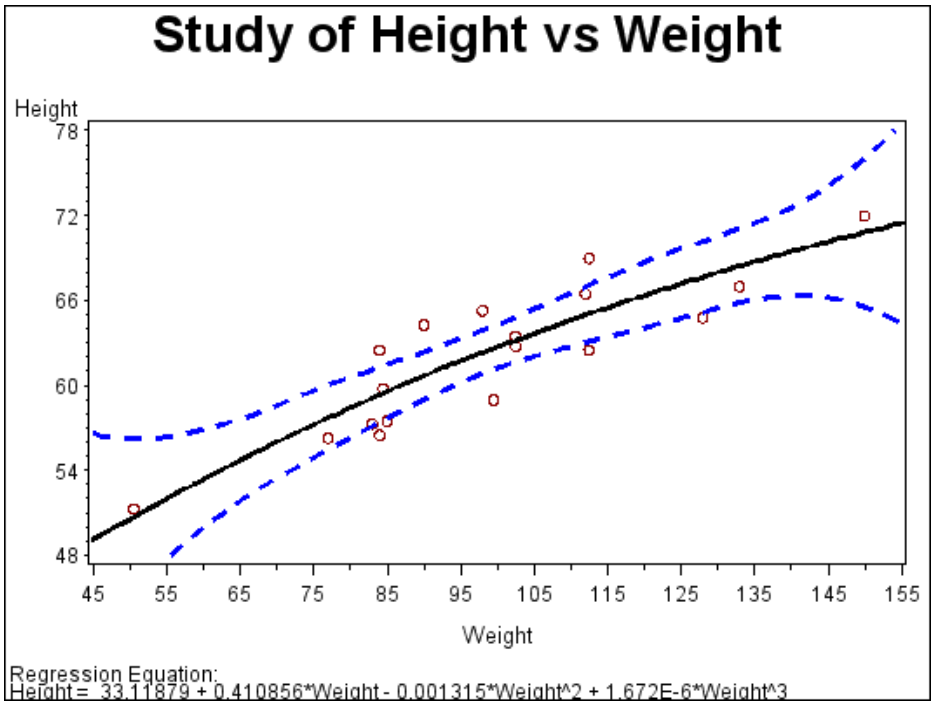
SYMBOL statement

Sample library member: GPLVRBL1



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.

This example then requests the same plot with some modifications. As shown by the following output, 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.



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"
          j=r "GPLVRBL1(a) ";
```

Generate a default scatter plot.

```
proc gplot data=sashelp.class;
   plot height*weight;
run;
```

Redefine footnotes to make room for the regression equation.

```
footnote1; /* this clears footnote1 */
```

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

Procedure features:

PLOT statement option:

HMINOR=
LVREF=
VAXIS=
VMINOR=
VREF=

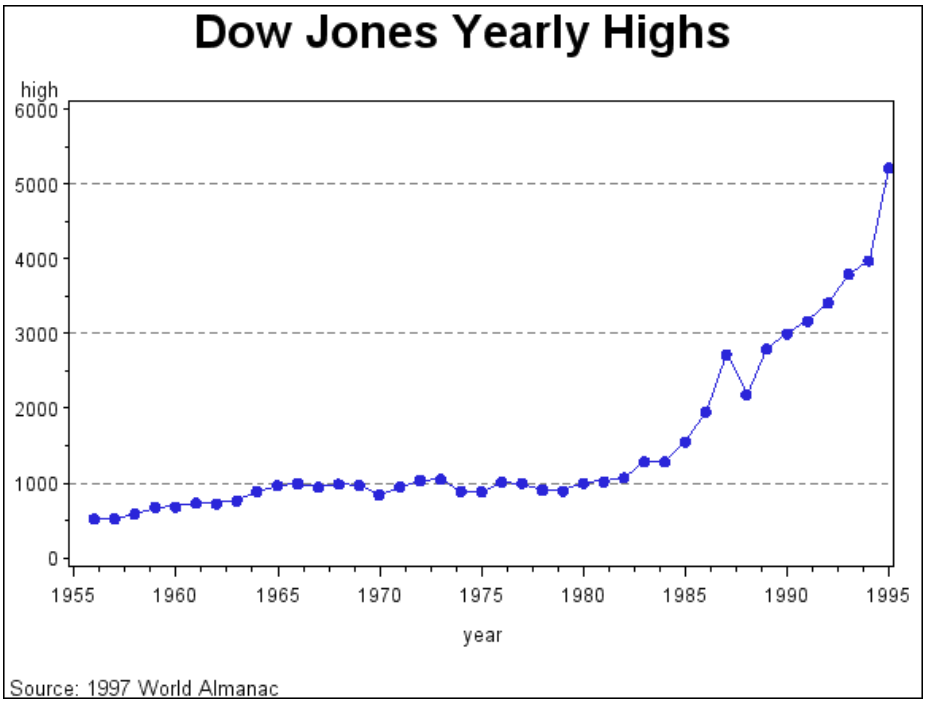
Other features:

GOPTIONS statement option:

BORDER

SYMBOL statement

Sample library member: GPLDTPT1



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).

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"
          j=r " GPLDTPT1 ";
```

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

Procedure features:

PLOT statement options:

COLOR=
HAXIS=

HMINOR=
LEGEND=
LVREF=
OVERLAY
VAXIS=
VMINOR=
VREF=

Other features:

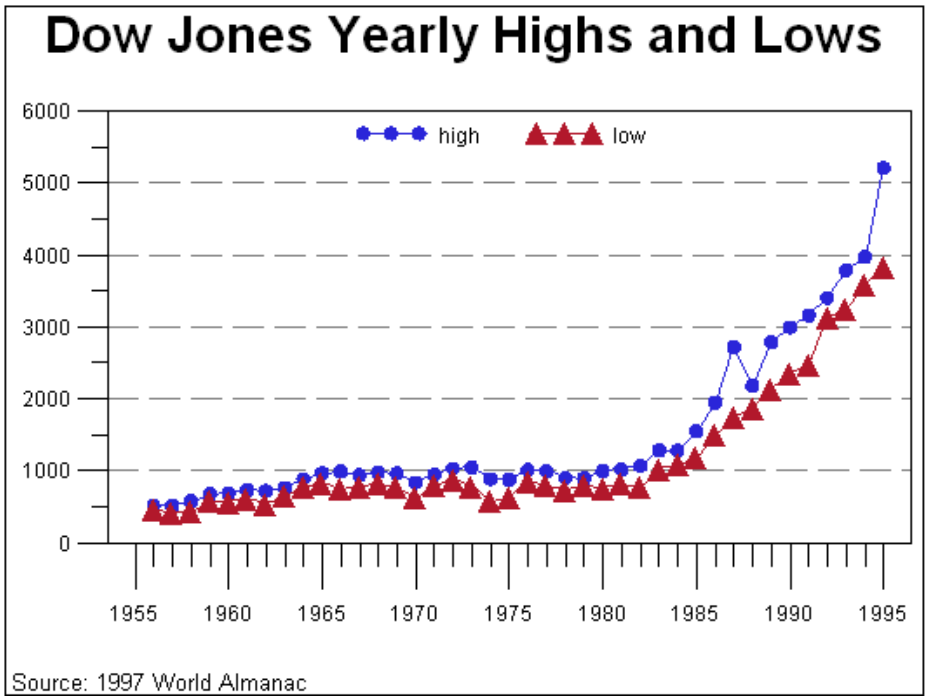
GOPTIONS statement options:

BORDER
RESET=

LEGEND statement

SYMBOL statement

Sample library member: GPLOVRL1



In this example, one PLOT statement plots both the HIGH and LOW variables against the variable YEAR using two plot requests. The OVERLAY option on 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. △

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 titles 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

Procedure features:

PLOT statement options:

AREAS=
HAXIS=
HMINOR=
VAXIS=
VMINOR=
CAXIS=
OVERLAY

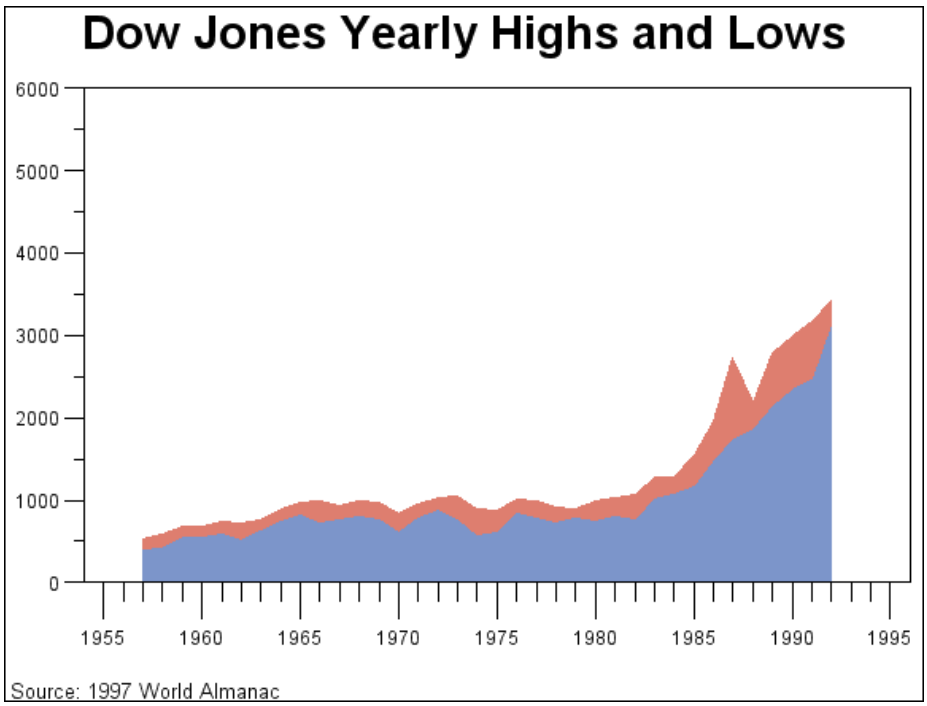
Other features:

GOPTIONS statement option:

BORDER

SYMBOL statement

Sample library member: GPLFILL1



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.

Set the graphics environment. BORDER draws a border around the graph.

```
goptions reset=all border;
```

Define title and footnote.

```
title1 "Dow Jones Yearly Highs and Lows";
footnote1 j=l " Source: 1997 World Almanac"
          j=r "GPLFILL1 ";
```

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 symbol characteristics. The INTERPOL= option specifies a line to connect data points. The line creates the fill boundary.

```
symbol1 interpol=join;
```

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);
```

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

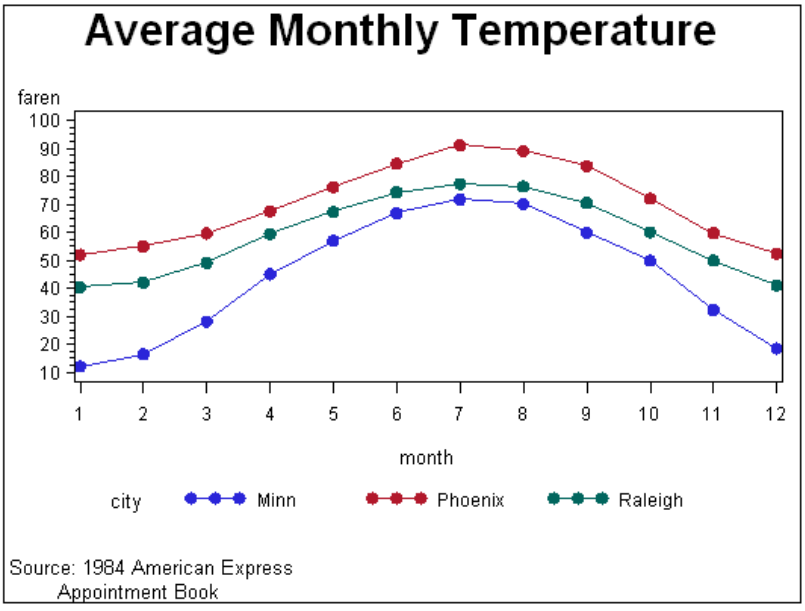
Procedure features:

PLOT classification variable
PLOT statement options:
HAXIS=
HMINOR=
LEGEND=
VAXIS=
VMINOR=

Other features:

GOPTIONS statement option:
BORDER
AXIS statement
SYMBOL statement
RUN-group processing

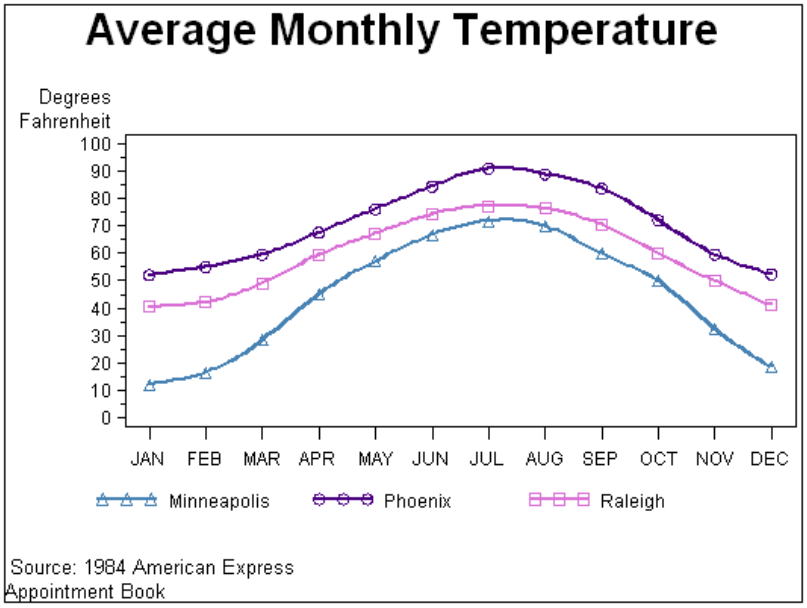
Sample library member: GPLVRBL2



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 which 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 following output, the plot is enhanced by using different symbol definitions and colors for each plot line, changing axes labels, and scaling the vertical axes differently.



Set the graphics environment.

```
goptions reset=all border;
```

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 footnote.

```
title1 "Average Monthly Temperature";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l "         Appointment Book"
        ;
```

Define symbol characteristics. 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;
```

Modify FOOTNOTE2 to reference new output.

```
footnote2 j=l "Appointment Book"
          ;
```

Define new symbol characteristics. SYMBOL statements are assigned to the values of CITY in alphabetical order. For example, the value **Minn** is assigned SYMBOL1.

```
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
      value=("JAN" "FEB" "MAR" "APR" "MAY" "JUN"
            "JUL" "AUG" "SEP" "OCT" "NOV" "DEC")
      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");
```

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

Procedure features:

PLOT statement options:

HAXIS=
HMINOR=

PLOT and PLOT2 statement options:

VAXIS=
VMINOR=

Other features:

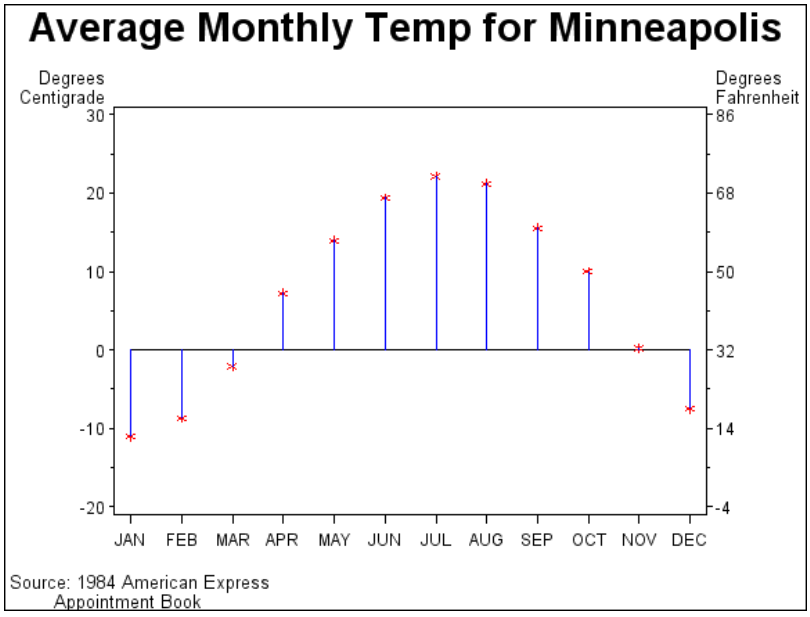
GOPTIONS statement option:

BORDER

AXIS statement

SYMBOL statement

Sample library member: GPLSCVL1



This example shows how a PLOT2 statement generates a right axis that displays the values of the vertical coordinates in a different scale from the scale that is used for the left axis.

In this plot of the average monthly temperature for Minneapolis, temperature variables that represent degrees centigrade (displayed on the left axis) and degrees Fahrenheit (displayed on the right axis) 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.

Set the graphics environment.

```
goptions reset=all border;
```

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 footnote.

```
title1 "Average Monthly Temp for Minneapolis";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l "         Appointment Book"
          j=r "GPLSCVL1 ";
```

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, and 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
      value=( "JAN" "FEB" "MAR" "APR" "MAY" "JUN"
            "JUL" "AUG" "SEP" "OCT" "NOV" "DEC")
      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 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;
   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

Procedure features:

PLOT statement options:

HTML=
HTML_LEGEND=

ODS features:

ODS HTML statement:

BODY=
NOGTITLE
PATH=

Other features:

GOPTIONS statement option:

BORDER

BY statement

GOPTIONS statement

Sample library member: GPLDRIL1

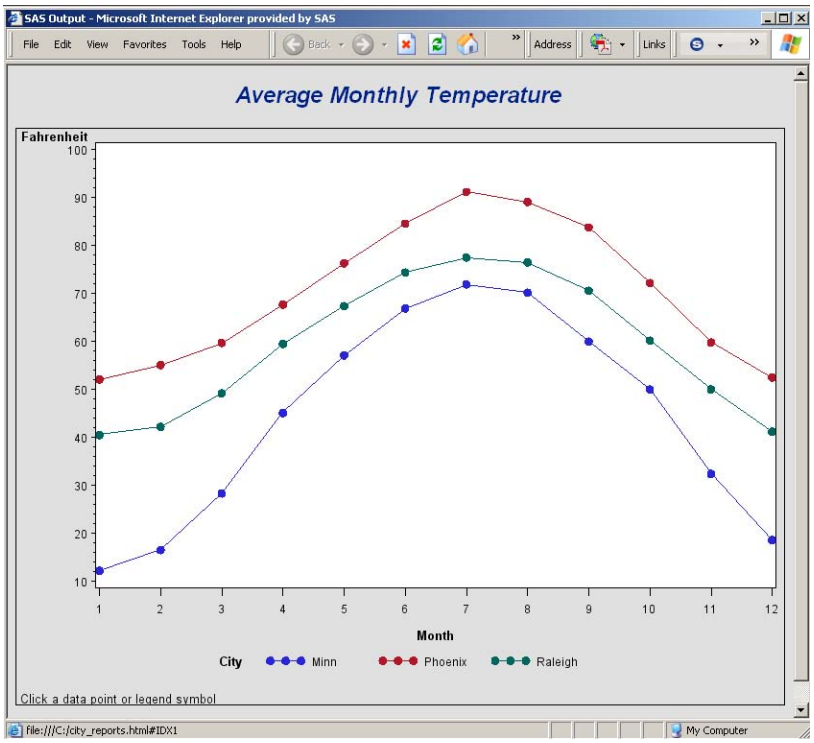
This example shows how to create a plot with simple drill-down functionality for the Web. If you display the plot in a Web browser, you can select any plot point or legend symbol to display a report on monthly temperatures for the selected city.

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:

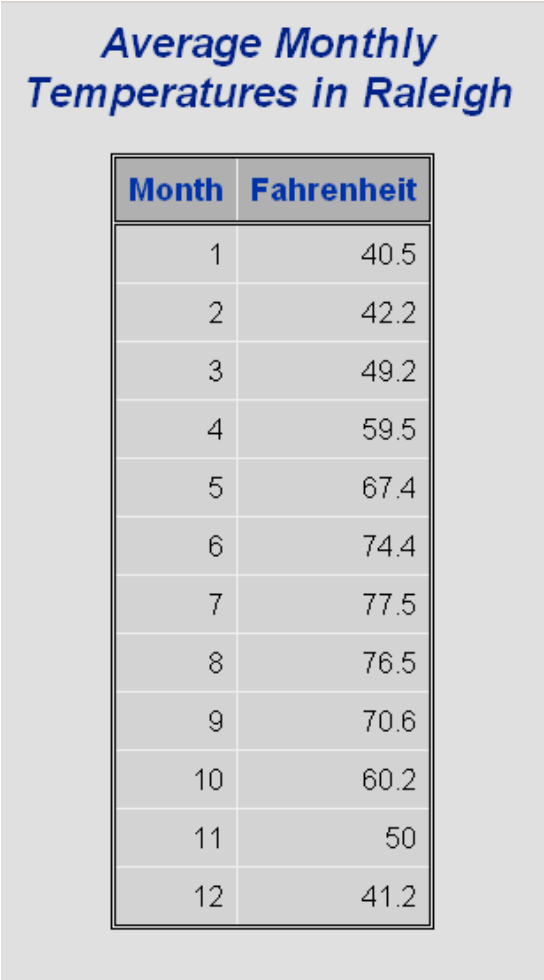
- explicitly name the HTML files and direct the different types of output to different files
- use BY-group processing with ODS, and determine the anchor names for the different pieces of output
- use the PATH= option to specify the destination for the HTML and GIF files created by the ODS statement
- 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 on drill-down graphs, see “Adding Links with the HTML= and HTML_LEGEND= Options” on page 601.

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, enabling you to select any plot point or legend symbol to drill down to a report on the yearly temperatures for the corresponding city. The following figure shows the drill-down plot as it is viewed in a browser.



The following figure shows the report that appears when you select any plot point or legend symbol that corresponds to the data for Raleigh.



Average 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

Close the ODS listing destination for output. To conserve system resources, use ODS LISTING to close the Listing destination for procedure output. Thus, the graphics output is not displayed in the GRAPH window, although it is written to the catalog.

```
ods listing close;
```

Define graphics output location.

```
filename odsout  "c:\";
```

Set the graphics environment.

```
goptions reset=all border device=gif;
```

Open an HTML output file in ODS.

```
ods html path=odsout gpath=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 will create. 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='citciy_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"
                  j=r "GPLDRIL1 ";

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 will be a hot zone that links to target output, and the HTML_LEGEND= option determines that the legend symbols will be 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 close;
ods html path=odsout
body="city_reports.html";
```

Sort data set NEWTEMP in order by city.

```
proc sort data=newtemp;
   by city month;
run;
quit;
```

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), and therefore 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      / display group;
  define Fahrenheit / display group;
run;
```

Close the HTML destination, and open the LISTING destination.

```
ods html close;
ods listing;
```

CHAPTER

46

The GPROJECT Procedure

Overview

The GPROJECT procedure processes traditional map data sets by converting spherical coordinates (longitude and latitude) into Cartesian coordinates for use by the GMAP procedure. The process of converting coordinates from spherical to Cartesian is called *projecting*. Many of the map data sets that are available with SAS/GRAPH contain unprojected longitude and latitude coordinates. When these coordinates are plotted by the GMAP procedure, which is designed to plot points on a two-dimensional plane, the resulting map is often reversed and distorted as a result of forcing the spherical map coordinates onto a flat plane.

The GPROJECT procedure enables you to use one of several map projection techniques to project the latitude and longitude coordinates onto a two-dimensional plane while attempting to minimize the distortion of area, distance, direction, and shape properties of the original sphere. The output map data set that is produced by the GPROJECT procedure contains Cartesian coordinates that can be displayed correctly using the GMAP procedure.

The GPROJECT procedure can also create a rectangular subset of the input map data set by excluding all points with longitude and latitude values that fall outside of a specified range. This provides a simple way to reduce the size of the map data set if you need only a portion of a larger map.

The GPROJECT procedure does not produce any graphics output. Instead, it produces an output map data set, which can be used as the input map data set for the GMAP procedure (see Chapter 43, "The GMAP Procedure," on page 1239).

Figure 46.1 on page 1396 and Figure 46.2 on page 1397 illustrate the effect of using GPROJECT defaults (Albers projection with standard parallels that are calculated by the procedure) to project a typical map data set with coordinates that are stored as longitude and latitude.

The program for the following maps can be seen in Example 1 on page 1409.

Figure 46.1 Map before Projection (GPJDEFLT(a))

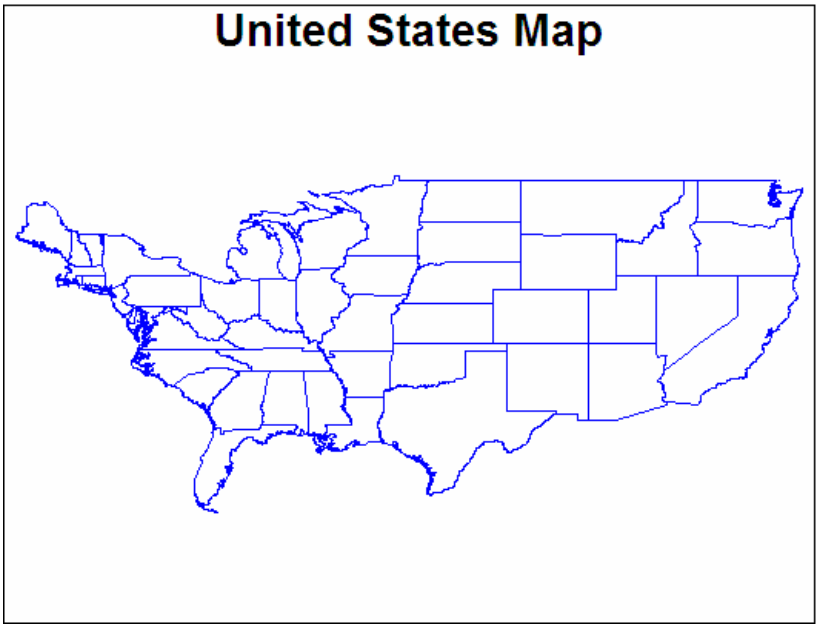


Figure 46.2 Map after Projection (GPJDEFLT(b))

Concepts

About the Input Map Data Set

The input map data set must be in traditional map data set format (see "About Traditional Data Sets" on page 1244), and it must contain these variables:

- a numeric variable named X that contains the longitude coordinates of the map boundary points.
- a numeric variable named Y that contains the latitude coordinates of the map boundary points.
- one or more *identification variables* that uniquely identify the unit areas in the map. These variables are listed in the ID statement.

The X and Y variables contain the values that are to be projected.
In addition, the input map data set can also contain these variables:

- a numeric variable named SEGMENT that distinguishes nonconterminous segments of the unit areas.
- a numeric variable named DENSITY that can be used to affect the output from PROC GPROJECT. See "Clipping Map Data Sets" on page 1408 for more information.

Other variables in the input map data set do not affect the GPROJECT procedure.

Input Map Data Sets that Contain Only Unprojected Values

The following is a list of all of the data sets supplied by SAS that contain X and Y variables whose values are unprojected:

CANADA3
CANADA4
COUNTIES
COUNTY
STATES

See Example 1 on page 1409 for an illustration of this type of input map data set and the variables it contains.

Note: Projection is appropriate for map data sets in which the X and Y variable values represent longitude and latitude. Some of the map data sets that are supplied with SAS/GRAPH have already been projected; such data set should not be projected again. △

Input Map Data Sets that Contain Both Projected and Unprojected Values

Most traditional map data sets contain both sets of variables (X, Y and LONG, LAT) for projected and unprojected maps. In these cases, the X and Y variables produce a projected map so you do not need to use the GPROJECT procedure. However, you might want to use the LONG and LAT variables to reproject the map using a different projection type. To do this you must first rename the LONG and LAT variables as X and Y. It is necessary to rename the LONG and LAT variables because the GPROJECT procedure looks for variables that are named X and Y by default. You can create a new map data set using the OUT= option, drop the current X and Y variables, and rename the LONG and LAT variables as X and Y. Your new data set then contains unprojected values in X and Y. The following statements illustrate how to do this:

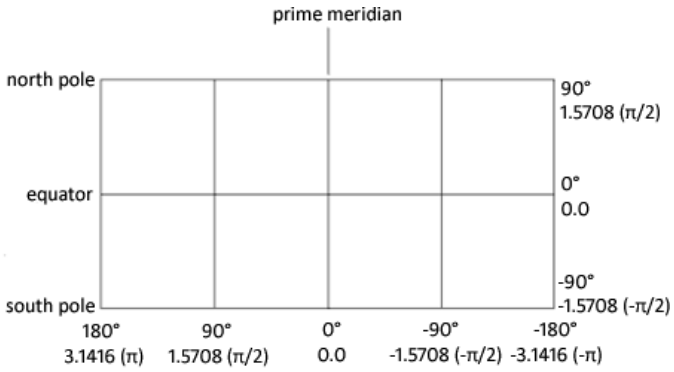
```
proc gproject data=maps.austral
              (drop=x y rename=(long=x lat=y))
              out=newaust;
   id id;
run;
```

For additional information on the supplied SAS/GRAPH map data sets, see “About Map Data Sets” on page 1244 and the METAMAPS data set in your maps data set directory.

About Coordinate Values

Figure 46.3 on page 1399 shows the standard coordinate system for map data sets with coordinates in longitude and latitude. For the longitude and latitude values (below and to the right of the figure, respectively) the upper value is expressed in degrees and the lower value is expressed in radians. A radian is approximately 57.3 degrees.

Figure 46.3 Longitude and Latitude Coordinates



By default, the GPROJECT procedure assumes that the units for the input coordinate values are radians and that values for the horizontal coordinate increase from east to west across the map. If your map coordinates are stored as degrees of arc, use the DEGREE option in the PROC GPROJECT statement. If the horizontal coordinate values in the map increase west-to-east rather than east-to-west, use the EASTLONG option in the PROC GPROJECT statement. See "Options" on page 1403 for details about the DEGREE and EASTLONG options.

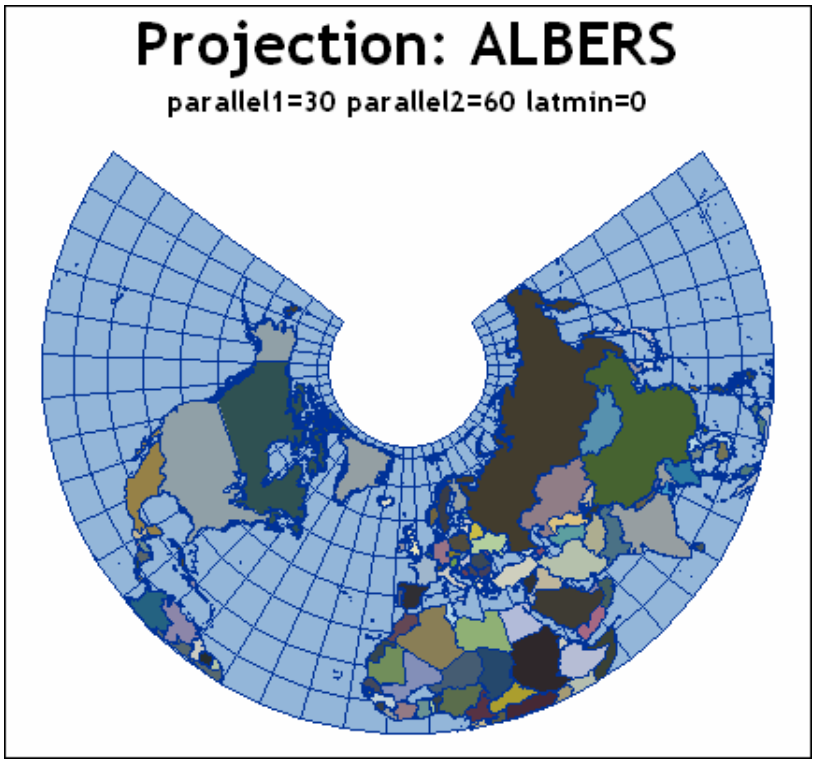
The unprojected map data sets that are provided with SAS/GRAPH can be projected if you use the default procedure characteristics: coordinate units in the data sets are radians, and horizontal values increase east-to-west.

About Types of Map Projections

The GPROJECT procedure performs three different types of projection: Albers' equal-area projection with two standard parallels (the default method), Lambert's conformal projection with two standard parallels, or the gnomonic projection (an azimuthal equidistant projection).

Albers' Equal-Area Projection

Figure 46.4 Albers' Projection



The *Albers' projection* is a conic projection from the surface of the sphere to a cone secant to the sphere, cutting it at two standard parallels of latitude. The axis of the cone coincides with an extension of the polar axis of the sphere. Each section of the resulting map bears a constant ratio to the area of the sphere. In general, distortion in shape tends to increase toward the poles in latitudes outside of the two standard parallels.

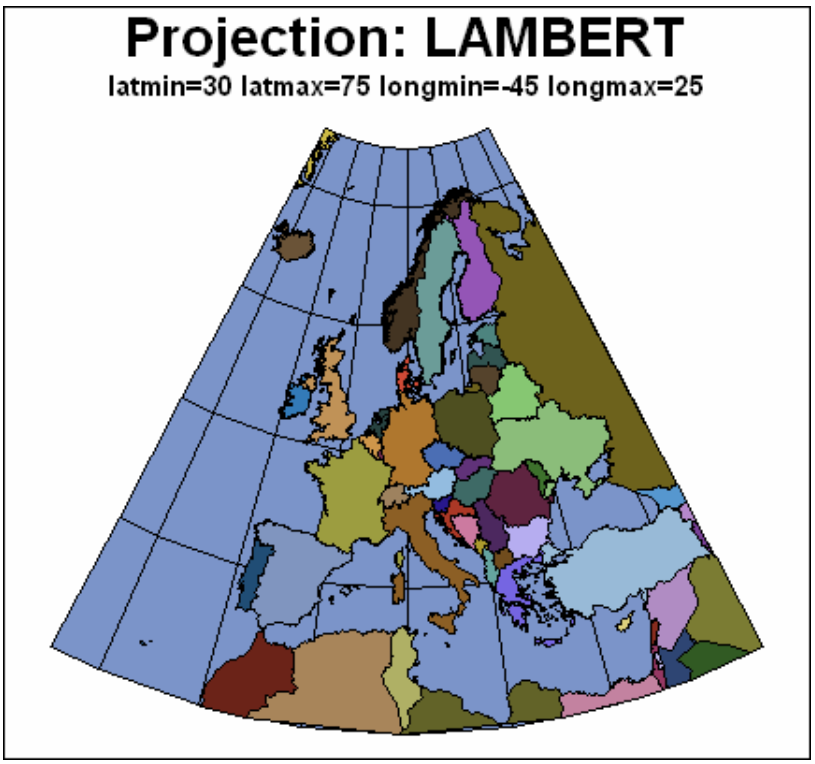
Figure 46.4 on page 1400 illustrates an Albers' equal-area projection of the northern hemisphere.*

The Albers' projection is suitable for portraying areas of large and small east-to-west extent and produces satisfactory results in most cases. However, both standard parallels must lie on the same side of the equator, so this method might not be suitable for map data sets of large north-to-south extent that span the equator. For those map data sets, use the gnomonic projection method.

* The projection examples in this topic include grid lines that were added with the Annotate facility. See the Samples area at **`support.sas.com`** for an example of adding latitude and longitude lines to a map.

Lambert's Conformal Projection

Figure 46.5 Lambert's Projection



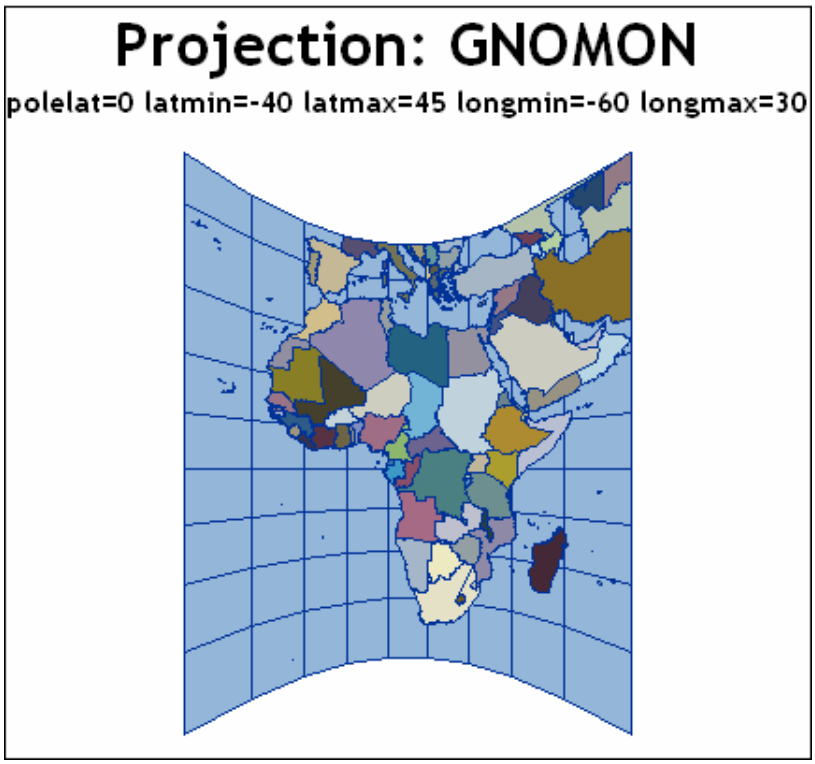
The *Lambert's projection* is obtained from a secant cone in the same manner as Albers' projection. In the Lambert's projection, meridians of longitude are straight lines that radiate from the apex of the cone, while parallels of latitude are concentric circles. The Lambert's projection is somewhat better than the Albers' projection at representing the original shape of projected unit areas, while the Albers' projection is somewhat better at representing relative sizes of projected unit areas.

Figure 46.5 on page 1401 illustrates a Lambert's conformal projection of Europe.

The Lambert's projection is ideal for navigational charts and maps of relatively small east-to-west extent. However, as in the Albers' projection, both standard parallels must lie on the same side of the equator, so this method might not be suitable for map data sets that span the equator. For those map data sets, use the gnomonic projection method.

Gnomonic Projection

Figure 46.6 Gnomonic Projection



The *gnomonic projection* is a planar projection from the surface of the sphere directly onto an imaginary plane tangent to the sphere at the map projection pole. By default, the projection pole is placed at the center of the map data set that is to be projected, but you can specify the projection pole to be anywhere on the surface of the sphere. (See the POLELAT= and POLELONG= option on page 1406.)

Figure 46.6 on page 1402 illustrates a gnomonic projection of Africa.

In the gnomonic projection, distortion increases as the distance from the map pole increases. Because of this distortion, the PROC GPROJECT procedure deletes all of the observations that lie more than 85 degrees from the map pole. The gnomonic projection is best suited for mapping areas of small east-to-west extent.

Procedure Syntax

Requirements: Exactly one ID statement is required.

PROC GPROJECT <*option(s)*>;

ID *id-variable(s)*;

PROC GPROJECT Statement

Identifies the input and output map data sets. Can specify the type of projection, and the criteria for clipping and projection.

Requirements: An input map data set is required.

Syntax

PROC GPROJECT <*option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- data set options:
  - DATA=*input-map-data-set*
  - OUT=*output-map-data-set*
- projection options:
  - PARADIV=*n*
  - PARALLEL1=*latitude*
  - PARALLEL2=*latitude*
  - POLELAT=*latitude*
  - POLELONG=*longitude*
  - PROJECT=ALBERS | GNOMON | LAMBERT | NONE
- coordinate options:
  - DEGREES
  - DUPOK
  - EASTLONG
  - NODATELINE
- clipping options:
  - LATMIN=*min-latitude*
  - LATMAX=*max-latitude*
  - LONGMIN=*min-longitude*
  - LONGMAX=*max-longitude*

Options

DATA=*input-map-data-set*
identifies the map data set to be processed. By default, the procedure uses the most recently created SAS data set.

See also: "About the Input Map Data Set" on page 1397 and "SAS Data Sets" on page 54

Featured in: Example 4 on page 1416

DEGREES
specifies that the units for the longitude (X variable) and latitude (Y variable) coordinates are degrees. By default, coordinate units are considered to be radians. The GPROJECT procedure stops processing the data set if coordinates are out of range.

Alias: DEG

DUPOK
specify that observations be retained when their projected X and Y values are identical to those in the previous observation. By default, successive identical observations are deleted.

Note: This option is useful when you want to add annotation to a map that contains duplicate coordinates. △

Alias: ASIS

EASTLONG
specifies that the longitude (X variable) values in the input map data set increase to the east (that is, positive longitude values are east of the prime meridian.) By default, longitude values increase to the west.

Alias: EAST

LATMAX=*max-latitude*
specify the maximum latitude that is included in the projection. Any unit areas that cross the selected latitude are clipped and closed along the specified parallels. The LATMAX= and LATMIN= options do not have to be paired; you can specify a maximum latitude without specifying a minimum.

When PROJECT=ALBERS, LAMBERT, or GNOMON, the GPROJECT procedure treats the value of *max-latitude* as degrees. When PROJECT=NONE, the procedure treats the value as a Cartesian coordinate.

Featured in: Example 3 on page 1414

LATMIN=*min-latitude*
specify the minimum latitude that is included in the projection. Any unit areas that cross the selected latitude are clipped and closed along the specified parallels. The LATMAX= and LATMIN= options do not have to be paired; you can specify a minimum latitude without specifying a maximum.

When PROJECT=ALBERS, LAMBERT, or GNOMON, the GPROJECT procedure treats the value of *min-latitude* as degrees. When PROJECT=NONE, the procedure treats the value as a Cartesian coordinate.

Featured in: Example 3 on page 1414

LONGMAX=*max-longitude*
specify the maximum longitude to be included in the projection. Any unit areas that cross the selected longitude are clipped and closed along the specified meridians. The LATMAX= and LATMIN= options do not have to be paired; you can specify a maximum longitude without specifying a minimum.

When PROJECT=ALBERS, LAMBERT, or GNOMON, the GPROJECT procedure treats the value of *max-longitude* as degrees. When PROJECT=NONE, the procedure treats the value as a Cartesian coordinate.

Featured in: Example 3 on page 1414

LONGMIN=*min-longitude*
specify the minimum longitude to be included in the projection. Any unit areas that cross the selected longitude are clipped and closed along the specified meridians. The LATMAX= and LATMIN= options do not have to be paired; you can specify a minimum longitude without specifying a maximum.

When PROJECT=ALBERS, LAMBERT, or GNOMON, the GPROJECT procedure treats the value of *min-longitude* as degrees. When PROJECT=NONE, the procedure treats the value as a Cartesian coordinate.

Featured in: Example 3 on page 1414

NODATELINE
enables contiguous projections for maps that cross the line between 180 degrees and -180 degrees longitude. For example, if you project a map of Asia, then the eastern tip of the continent might be projected on the left side of the map by default. The NODATELINE option enables the entire continent to be projected as a contiguous area.

OUT=*output-map-data-set*
names the new map data set, which contains the coordinates of the new unit areas that are created by the GPROJECT procedure.

By default, the GPROJECT procedure names the new data set that uses the DATA*n* naming convention. That is, the procedure uses the name WORK.DATA*n*, where *n* is the next unused number in sequence. Thus, the first automatically named data set is DATA1, the second is DATA2, and so on.

Featured in: Example 4 on page 1416

PARADIV=*n*
specifies the divisor that computes the values used for standard parallels for the Albers' or Lambert's projections when explicit values are not provided. By default PARADIV=4, which causes the standard parallels to be set at 1/4 and 3/4 of the range of latitude values in the input map data set.

See also: PARALLEL1= and PARALLEL2= option

PARALLEL1=*latitude*
PARALLEL2=*latitude*
specify values for the standard parallels that are used in the Albers' or Lambert's projection. *Latitude* must be in degrees. Positive values indicate north of the equator, and negative values indicate south of the equator. These options are ignored for the gnomonic projection.

By default, the GPROJECT procedure calculates values for the standard parallels. The defaults are chosen to minimize the distortion inherent in the projection process. The algorithm used is as follows:

PARALLEL1 = minlat + R / P_D
PARALLEL2 = maxlat - R / P_D
where:

R
is the range of latitude values in the input map data set.

P_D
is the PARADIV= value (see the discussion of the PARADIV= option).

minlat
is the minimum latitude value in the input map data set.

maxlat
is the maximum latitude value in the input map data set.

If you do not use PARALLEL1= or PARALLEL2=, or you omit either option, the GPROJECT procedure uses the calculated value for the missing parameter.

The standard parallels, whether explicitly specified or supplied by the procedure, must lie on the same side of the equator. If they do not, PROC GPROJECT prints an error message and stops (the procedure might calculate standard parallels that lie on opposite sides of the equator). When projecting a map data set that contains unit

areas that cross the equator, you might have to explicitly specify standard parallels that both lie on the same side of the equator. If this causes excessive distortion of the map, you might be able to use the gnomonic projection instead of the Albers' or Lambert's projection because the gnomonic technique has no such limitations at the equator.

Alias: PARALEL1, PARALEL2

POLELAT=*latitude*
POLELONG=*longitude*
specify a projection pole to use for the gnomonic projection. The projection pole is the point at which the surface of the sphere touches the surface of the imaginary plane onto which the map is projected. The POLELAT= option specifies the latitude of the projection point.

Units for latitude are degrees; positive values indicate north of the equator, and negative values indicate south of the equator. The POLELONG= option gives the longitude for the projection point. Units for longitude are degrees; positive values indicate west of the prime meridian, and negative values indicate east of the prime meridian (unless EASTLONG also has been used in the PROC GPROJECT statement).

If you do not use the POLELAT= or POLELONG= option, or you omit either option, PROC GPROJECT uses values for the position of the center of the unit areas that are defined by the DATA= data set for the missing parameter.

Note: The map that is defined by the input map data set should not contain points more than 85 degrees (1.48353 radians) from the projection pole; all points that exceed this value are deleted from the output map data set. △

Featured in: Example 2 on page 1412

PROJECT=ALBERS | LAMBERT | GNOMON | NONE
specifies the projection method to apply to the map data set. Values for the PROJECT= option are as follows:

ALBERS
specifies Albers' equal-area projection with two standard parallels.

LAMBERT
specifies Lambert's conformal projection with two standard parallels.

GNOMON
specifies the gnomonic projection, which is an azimuthal projection.

NONE
specifies that no projection should be performed. Use this option in conjunction with the LATMIN=, LATMAX=, LONGMIN=, and LONGMAX= options to perform clipping without projection.
By default, PROJECT=ALBERS.

Note: There are several additional projections available. They are: ADAMS, AITOFF, APIANUS, ARAGO, BEHRMANN, BRAUN, CYLINDRI, ECKERT1, ECKERT3, ECKERT5, EQUIRECT or MARINUS, GALL, HAMMER, KVRSKY7, MILLER1, MILLER2 , ORTHO, PARABOLI, PETERS, PUTNINS4, ROBINSON, STEREO, WINKEL2. △

See also: "About Types of Map Projections" on page 1399

Featured in: Example 2 on page 1412

ID Statement

Identifies the variable or variables that define the hierarchy of the current unit areas in the input map data set.

Requirements: At least one *id-variable* is required.

Featured in: Example 1 on page 1409.

Syntax

ID *id-variable(s)*;

Required Arguments

id-variable(s)
specifies one or more variables in the input map data set that identify unit areas. *Id-variable* can be either numeric or character.

Each group of observations with a different ID variable value is evaluated as a separate unit area.

Using the GPROJECT Procedure

You can use PROC GPROJECT statement options to do the following actions:

- select the map projection method
- specify the map projection criteria
- create a rectangular subset of the input map data set

The following sections describe how you can use PROC GPROJECT statement options to select your own projection method and projection criteria.

Selecting Projections

Except when projecting map data sets that cover large areas, all three types of projections (Albers', Lambert's, and gnomonic) produce relatively similar results when you use default projection criteria, so you usually do not need to be concerned about which projection method to use when you produce maps of small regions.

However, the default projection criteria might be unsuitable in some circumstances. In particular, the default specifications fail when the map that is being projected extends on both sides of the equator. On other occasions, you might want to select a projection method to achieve a particular effect.

For the Albers' and Lambert's projections, the two standard parallels must both lie on the same side of the equator. PROC GPROJECT stops and gives an error message if this condition is not met, regardless of whether you explicitly specify parallel values or let the procedure calculate default values. See the descriptions of the PARALEL1= and PARALEL2= options on page 1405 for more information on how to specify the two standard parallels.

Controlling Projection Criteria

For both the Albers' and Lambert's projections, PROC GPROJECT calculates appropriate standard parallels. You can override either or both of these selections if you explicitly specify values for the PARALEL1= or PARALEL2= option. You can influence the selection of default parallels if you use the PARADIV= option. See "Options" on page 1403 for more information on these options.

For the gnomonic projection, PROC GPROJECT determines the longitude and latitude of the approximate center of the input map data set area. You can override either or both of these selections if you explicitly specify values for the POLELAT= or POLELONG= option. See "Options" on page 1403 for more information.

The clipping options, discussed in "Clipping Map Data Sets" on page 1408, can also influence the calculations of the default standard parallels by changing the minimum and maximum coordinate values.

Clipping Map Data Sets

The GPROJECT procedure can create rectangular subsets of the input map data set. This capability provides a way to extract a portion of a larger map if you do not need all the original unit areas for your graph. The procedure enables you to clip unit area boundaries at specified latitude values, longitude values, or both. Unit areas that fall completely outside of the specified clipping limits are excluded from the output map data set. Unit areas bisected by the clipping limits are closed along the clipping parallels and meridians, and all points outside of the clipping limits are excluded.

If the input map data set contains the DENSITY variable, any new vertex points and corners that are created by PROC GPROJECT are assigned a DENSITY value of 0 in the output map data set. This enables you to use a subset of the clipped map without using PROC GREDUCE to assign new DENSITY values. (See Chapter 48, "The GREDUCE Procedure," on page 1447 for information on how to reduce the number of points that you need to draw a map.)

You can specify the minimum latitude to be retained in the output map data set with the LATMIN= option and the maximum latitude with LATMAX= option. Minimum and maximum longitude values are specified with the LONGMIN= and LONGMAX= options, respectively. See "Options" on page 1403 for more details on these options.

This is how the PROC GPROJECT interprets the clipping longitude and latitude values:

- If you specify PROJECT=NONE in the PROC GPROJECT statement, the procedure assumes that the input map data set is already projected and the clipping longitude and latitude values are Cartesian coordinates. In this case, the LATMAX= and LATMIN= options specify the top and bottom edges, respectively, of the area that you want to extract, and the LONGMAX= and LONGMIN= options specify right and left edges, respectively.

  You must be familiar with the range of values in the X and Y variables in order to select appropriate clipping limits. Use the MEANS or SUMMARY procedure in Base SAS to determine the range of values in X and Y. See the *Base SAS Procedures Guide* for more information.

- If PROJECT=ALBERS, LAMBERT, or GNOMON, the clipping values are treated as degrees.

Depending on the size and position of the clipped area and the type of projection that is performed, the resulting map might not be exactly rectangular. PROC GPROJECT performs clipping before projection, so the clipped area might be distorted by the projection process.

To produce a clipped area with a rectangular shape, use PROC GPROJECT in two steps:

1. Project the map using the appropriate projection method and projection criteria.
2. Project the map using PROJECT=NONE, and use the LATMIN=, LATMAX=, LONGMIN=, and LONGMAX= options to clip the map.

See Example 3 on page 1414, for an example of clipping an area from an unprojected map data set.

Examples

The following examples illustrate major features of the GPROJECT procedure.

Example 1: Using Default Projection Specifications

Procedure features:
ID statement

Sample library member: GPJDEFLT

This example demonstrates the effect of using PROC GPROJECT on an unprojected map data set without specifying any options. Because the PROJECT= option is not used in the PROC GPROJECT statement, the Albers' equal-area projection method is used by default. PROC GPROJECT supplies defaults for the standard parallels that minimize the distortion of the projected map areas.

Figure 46.7 Map before Projection (GPJDEFLT(a))

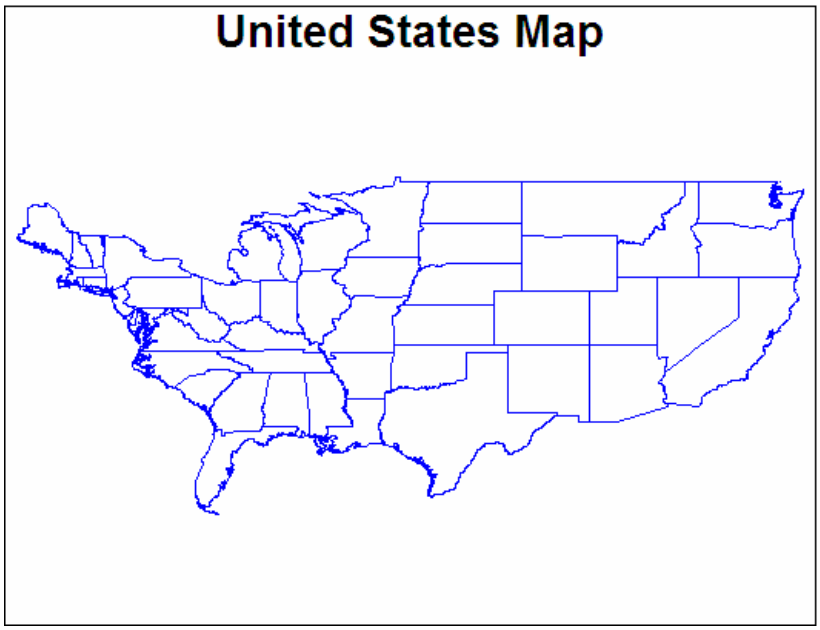


Figure 46.7 on page 1409 illustrates the output produced by the US48 map data set, which contains unprojected values in the X and Y variables. Output 46.1 shows the variables in the data set.

Output 46.1 The US48 Data Set

US48 Data Set

OBS	STATE	SEGMENT	DENSITY	X	Y
1	1	1	3	1.48221	0.56286
2	1	1	3	1.48226	0.56234
3	1	1	3	1.48304	0.56231
.					
.					
.					

The GPROJECT procedure is used with the US48 map data set as input to create the projected map data set, US48PROJ. The values for X and Y in this new data set are projected (Cartesian). Output 46.2 shows the variables in the data set.

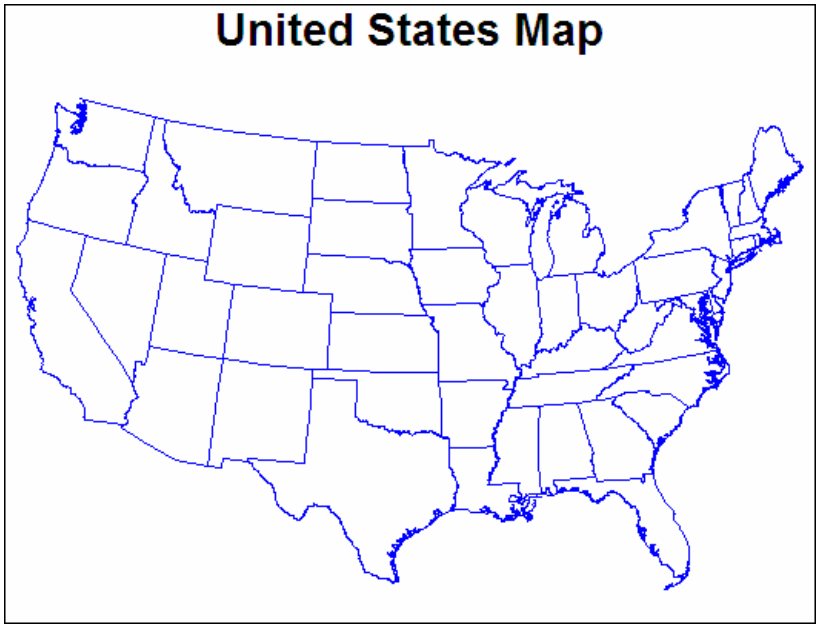
Output 46.2 The US48PROJ Data Set

US48PROJ Data Set

OBS	X	Y	DENSITY	STATE	SEGMENT
1	0.16068	-0.073470	3	1	1
2	0.16069	-0.073993	3	1	1
3	0.16004	-0.074097	3	1	1
.					
.					
.					

The new projected map data set, US48PROJ, is used to create the projected map, Figure 46.8 on page 1411.

Figure 46.8 Map after Projection (GPJDEFLT(b))



Set the graphics environment.

```
goptions reset=all border;
```

Create a reduced continental U.S. map data set and remove Alaska, Hawaii, and Puerto Rico.

```
data us48;
   set maps.states;
   if state ne 2 and state ne 15 and state ne 72;
run;
```

Define the title for the unprojected map.

```
title "United States Map";
```

Define the pattern characteristics.

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

Show the unprojected map.

```
proc gmap map=us48 data=us48 all density=4;
   id state;
   choro state / nolegend levels=1;
run;
```

Project the map data set using all default criteria. The ID statement identifies the variable in the input map data set that defines unit areas.

```
proc gproject data=us48
              out=us48proj;
   id state;
run;
```

Show the projected map.

```
proc gmap map=us48proj
          data=us48proj all density=4;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

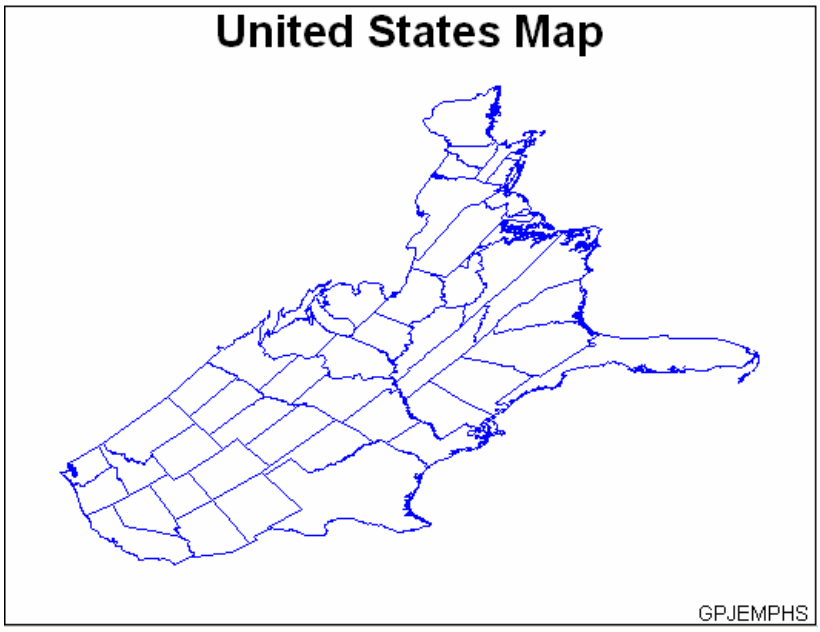
Example 2: Emphasizing Map Areas

Procedure features:

PROC GPROJECT options:

POLELAT=
POLELONG=
PROJECT=

Sample library member: GPJEMPHS



This example uses the gnomonic projection method to create a map in which the east coast of the United States appears disproportionately large compared to the west coast.

Set the graphics environment.

```
goptions reset=all border;
```

Create a reduced continental U.S. map data set and remove Alaska, Hawaii, and Puerto Rico.

```
data us48;
   set maps.states;
   if state ne 2 and state ne 15 and state ne 72;
   if density<4;
run;
```

Project the map onto a plane centered in the Pacific. The PROJECT= option specifies the projection method for the map data set. The POLELONG= and POLELAT= option specify a projection pole for the gnomonic projection. In this example, the pole is positioned in the Pacific Ocean.

```
proc gproject data=us48
              out=skew
              project=gnomon
              polelong=160
              polelat=45;
   id state;
run;
```

Define the title and footnote for the map.

```
title "United States Map";
footnote j=r "GPJEMPHS ";
```

Define the pattern characteristics.

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

Show the projected map.

```
proc gmap map=skew data=skew all;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

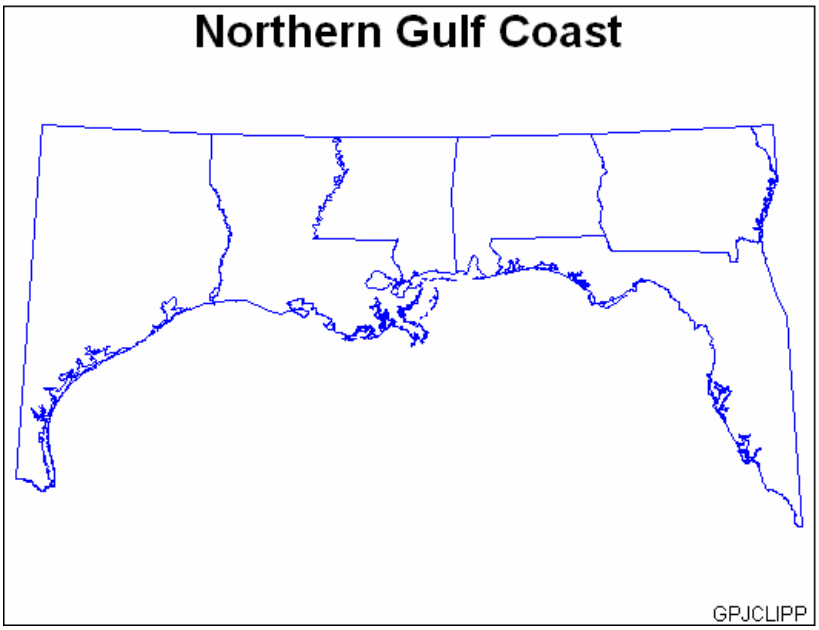
Example 3: Clipping an Area from the Map

Procedure features:

PROC GPROJECT options:

LONGMAX=
LONGMIN=
LATMAX=
LATMIN=

Sample library member: GPJCLIPP



This example uses the clipping capabilities of PROC GPROJECT to create a map of the states in the United States that border the Gulf of Mexico. Because the PROJECT= option is not used in the GPROJECT procedure, the Albers' equal-area projection method is used by default.

Set the graphics environment.

```
goptions reset=all border;
```

Clip and project a rectangular subset of the map. The LONGMIN= and LONGMAX= options specify the minimum and maximum longitudes to be included in the map projection.The LATMIN= and LATMAX= options specify the minimum and maximum latitudes to be included in the map projection.

```
proc gproject data=maps.states
              out=gulf
              longmin=81
              longmax=98
              latmin=25
              latmax=33;
   where density<5;
   id state;
run;
```

Define the title and footnote for the map.

```
title "Northern Gulf Coast";
footnote j=r "GPJCLIPP ";
```

Define the pattern characteristics.

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

Show the clipped map.

```
proc gmap map=gulf data=gulf all;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

Example 4: Projecting an Annotate Data Set

Procedure features:

PROC GPROJECT options:

DATA=
OUT=

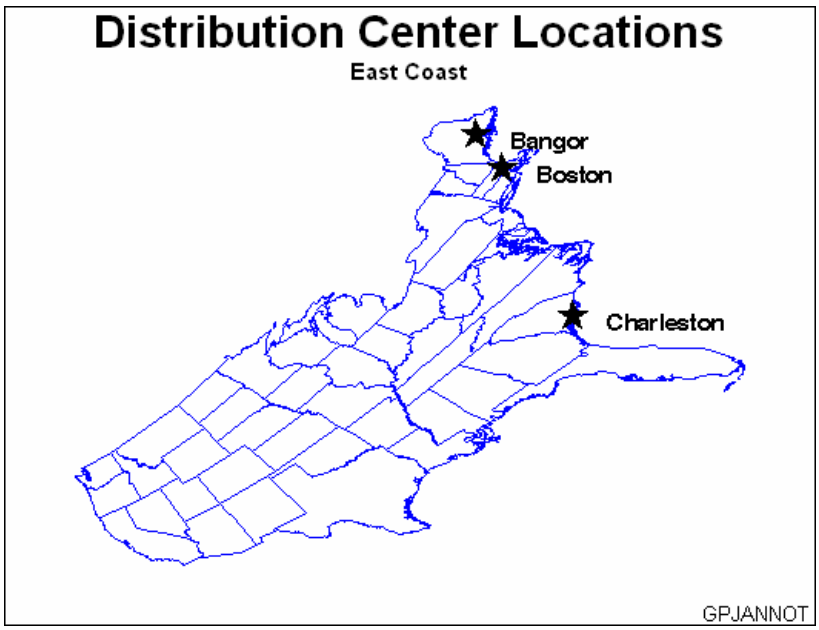
ID statement

Other features:

CHORO statement

Annotate data set

Sample library member: GPJANNOT



This example illustrates how to project an Annotate data set for use with a map data set. It labels the locations of Charleston, Boston, and Bangor on the map shown in the second example. Because the X and Y variables in the USCITY data set already have been projected to match the US data set, they cannot be used with the map that is produced by the second example. To properly label the projected map, the example uses the same projection method for the city coordinates that is used for the map coordinates. This example illustrates how to use the same projection method for both data sets.

Set the graphics environment.

```
goptions reset=all border;
```

Create a reduced continental U.S. map data set and remove Alaska, Hawaii, and Puerto Rico.

```
data us48;
   set maps.states;
```

```
   if state ne 2 and state ne 15 and state ne 72;
   if density<4;
run;
```

Create the Annotate data set CITIES from the MAPS.USCITY data set. The unprojected LONG and LAT variable values are converted to radians and substituted for the projected X and Y variable values. LONG and LAT are converted by multiplying them by the arccosine of -1 and dividing that amount by 180. The value of STATE is modified for each label to insure that it is unique.

```
data cities;
   set maps.uscity(keep=lat long city state);
   length function style color $ 8
          position $ 1 text $ 20;
   retain function "label" xsys ysys "2"
          hsys "1" when "a";
   if (state=45 and city="Charleston") or
      (state=25 and city="Boston") or
      (state=23 and city="Bangor");
    state+100; color="black"; size=8; text="V";
      position="5";
      style="marker"; x=long*arcos(-1)/180;
      y=lat*arcos(-1)/180; output;
    state+1; color="black"; size=5;
      text="   "||city;
      position="6"; style="swissb"; output;
run;
```

Create the data set ALL by combining the data set US48 and the data set CITIES.

```
data all;
   set us48 cities;
run;
```

Project the ALL data set. The DATA= option specifies the data set to be projected. The OUT= option specifies the name of the new projected data set that is created. The ID statement identifies the variable in the input map data set that defines map areas.

```
proc gproject data=all
              out=allp
              project=gnomon
              polelong=160
              polelat=45;
   id state;
run;
```

Separate the projected data set into the CITIESP Annotate data set and the US48P map data set. The annotate observations have STATE values that are greater than 100.

```
data citiesp us48p;
   set allp;
   if state > 100 then output citiesp;
   else output us48p;
run;
```

Define the title and footnote for the map.

```
title1 "Distribution Center Locations";
title2 "East Coast";
footnote j=r "GPJANNOT ";
```

Define the pattern characteristics.

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

Show the annotated map. The CHORO statement displays the projected map and annotates it using the projected Annotate data set.

```
proc gmap data=us48p map=us48p all;
   id state;
   choro state
         / nolegend levels=1
           annotate=citiesp;
run;
quit;
```

References

Pearson, F., II (1977), "Map Projection Equations," Report Number TR-3624, Naval Surface Weapons Center, Dahlgren Laboratory, March, 1977.

Richardus, P. and Adler, R.K. (1972), *Map Projections*, Amsterdam: North-Holland Publishing Company; New York: American Elsevier Publishing Company.

Robinson, A.H. (1978), *Elements of Cartography*, New York: John Wiley & Sons, Inc.

CHAPTER

47

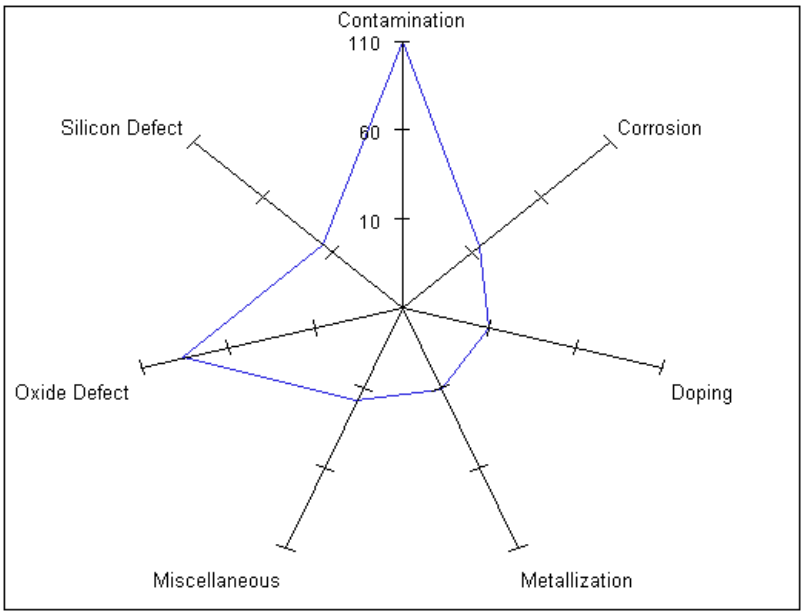
The GRADAR Procedure

Overview

The GRADAR procedure creates radar charts that show the relative frequency of data measures in quality control or market research problems. Radar charts are sometimes also called star charts.

On a radar chart, 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, thus giving 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.

Note: The GRADAR procedure is not supported by the Java device drivers. △



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	-

Procedure Syntax

Requirements: At least one CHART statement is required.

Global Statements: AXIS, FOOTNOTE, GOPTIONS, TITLE

Reminder: The procedure can include the BY, FORMAT, LABEL, and WHERE statements as well as SAS/GRAPH NOTE statement.

Supports: RUN-group processing

Restriction: Not supported by Java

PROC GRADAR <DATA=*input-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<ANNOTATE=*Annotate-data-set*>;

CHART *chart-variable* </ *option(s)*>;

PROC GRADAR Statement

Identifies the data set that contains the plot variables. Specifies an output catalog (optional).

Requirements: An input data set is required.

Syntax

PROC GRADAR <DATA=*input-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<ANNOTATE=*Annotate-data-set*>;

Options

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 650.

See also: Chapter 29, "Using Annotate Data Sets," on page 641

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.

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.

Requirements: At least one chart variable is required.

Global statements: AXIS, FOOTNOTE, and TITLE as well as the SAS/GRAPH NOTE statement

Syntax

CHART *chart-variable* </ *option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- chart options
  - ACROSSVAR=*variable*
  - CALENDAR
  - DOWNVAR=*variable*
  - FREQ=*variable*
  - MODE=SHARE | PROTECT | RESERVE
  - MISSING
  - NCOLS=*n*
  - NLEVELS=*n*
  - NROWS=*n*
  - NZEROREF
  - ORDERACROSS=FREQ | DATA | INTERNAL | FORMATTED | EXTERNAL
  - OTHER="*variable*"
  - OVERLAY=*overlay-variable*
  - STARTYPE=CORONA | POLYGON | RADIAL | SPOKE | WEDGE
  - SPEED=*speed-variable*
  - SUMVAR=*summary-variable*
  - WINDROSE
- axis options
  - STARAXIS= (AXIS<1...99><, . . . ,AXIS<1...99>>)
  - WAXIS=*n*
- appearance options
  - ANNOTATE=*Annotate-data-set*
  - CAXIS=*grid-color*
  - CFRAME=*background-color* | (*variable*)
  - CFRAMESIDE=*color*
  - CFRAMETOP=*color*
  - CSPOKES=*spoke-color*
  - CSTARCIRCLES=*color* | *(colors-list)*
  - CSTARFILL=*color* | *(colors-list)*

CSTARS=*color* | *(colors-list)*
CTEXT=*text-color*
CTILES=(*variable*) | *color*
FONT=*font*
FRAME | NOFRAME
HEIGHT=*height*
IFRAME=*fileref* | "*external-image-file*"
IMAGESTYLE=TILE | FIT
INBORDER
INHEIGHT=*value*
INTERTILE=*value*
LSPOKE=*linetype*
LSTARCIRCLES=(*linetypes*)
LSTARS=(*linetypes*)
MAXNVERT=*n*
MAXVERT=*n*
NOLEGEND
SPIDERWEB
SPKLABEL=CATEGORY | NONE
SPOKESCALE = CATEGORY | VERTEX
STARCIRCLES=(*values*)
STARFILL= lists of (SOLID | EMPTY) one for each star
STARINRADIUS=*value*
STARLEGEND=CLOCK | CLOCK0 | NUMBER | DEGREES | NONE
STARLEGENDLAB="*legend-label*"
STAROUTRADIUS=*value*
STARSTART=*value*
TILELEGEND=*variable*
TILELEGLABEL="*label*"
WFRAME=*n*
WINDROSECIRCLES=
WSPOKES=*n*
WSTARCIRCLES=(*line-widths*)
WSTARS=*line-widths* | (*line-widths*)

□ catalog entry description options

DESCRIPTION="*description*"
NAME="*name*"

□ ODS options

HTML=*variable*
HTML_LEGEND=*variable*

Required Arguments

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.

Options

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 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 not valid with CALENDAR charts—use the OVERLAY option instead.

See also: DOWNVAR=, NCOLS=, NROWS=, ORDERACROSS=

Featured in: Example 4 on page 1439 and Example 5 on page 1440

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 650.

See also: Chapter 29, “Using Annotate Data Sets,” on page 641

CALENDAR
produces a radar 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.

See also: NLEVELS=, OVERLAYVAR=

Featured in: Example 8 on page 1444

CAXIS=*grid-color*
specifies a color for the chart frame outline and the spokes or grid lines of the chart. The specified color must be a valid SAS/GRAPH color name. If you omit the CAXIS= option, 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.

Alias: CAXES=, CA=

Style reference: Color attribute of the GraphAxisLines element

See also: CFRAME=

Restriction: Not supported by ActiveX

CFRAME=*background-color* | (*variable*)
fills the frame area with the specified color. You can specify a valid SAS/GRAPH color name, or a character variable of length eight whose value is the background color.

Alias: CFR=

See also: CAXIS=

CFRAMESIDE=*color* | (*variable*)
specifies the color for filling the frame 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 of length eight 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 frame 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 frame 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 of length eight 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 frame 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 of length eight 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=

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, or a character variable of length eight whose value is the color. 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=

Featured in: Example 3 on page 1438

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 it finds from the following list:

1. the color(s) specified on the CSTARFILL= option
2. the color(s) specified on the CSTARS= option
3. the color(s) 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 also: CSTARS=

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 it finds from the following list:

1. the color(s) specified on the CSTARS= option
2. the color(s) specified on the CSTARFILL= option
3. the color(s) 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 also: CSTARFILL=

Featured in: Example 6 on page 1441

CTEXT=*color*

specifies a color for all text on the chart. The specified color must be a valid SAS/GRAPH color name. If you omit the CTEXT= option, the GRADAR procedure uses the first color it finds in the following list.

1. the CTEXT= option in a GOPTIONS statement
2. the color specified by the current style or, if the NOGSTYLE option is specified, then the default color is black for the ActiveX devices and the second color in the color list for all other devices

Style reference: Color attribute of the GraphValueText element

CTILES=(*variable*) | color

specifies either a character variable of length eight 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 596 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option on an ODS HTML statement, assuming 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. 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=

Restriction: This option is not valid with CALENDAR charts—use OVERLAY instead.

Featured in: Example 5 on page 1440

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 elements

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 for the frame, use the CFRAME= option for a filled frame and CAXIS= for only the frame outline color.

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 also: CAXIS=, CFRAME=

Featured in: Example 7 on page 1443

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 also: "Calculating Weighted Statistics" on page 1420

Featured in: Example 2 on page 1437, Example 3 on page 1438, andExample 4 on page 1439

HEIGHT=*height*
specifies the height in cells for labels and legends. The HEIGHT= option overrides the HTEXT= option in a GOPTIONS statement. This does not change the size of titles or footnotes

Alias: HLABEL=

HTML=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement.

See also: "Adding Custom Data Tips with the HTML= Option" on page 598 and "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with a legend value and point to the data or graph you want to display when the user drills down on the value. The maximum length for the value of this variable is 1024 characters.

See also: "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601

IFRAME=*fileref* | "*external-image-file*"
specifies an image file to use on 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 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 184.

Restriction: Not supported by ActiveX

IMAGESTYLE=TILE | FIT
specifies the way to display the image file that is specified on the IFRAME= option. TILE copies the image as many times as needed to fit 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 goes on the GOPTIONS statement. △

INBORDER
generates a inside border around the chart. This border is inside the border created by the BORDER option on 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

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

Featured in: Example 5 on page 1440

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. Be 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=

Featured in: Example 6 on page 1441

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. Missing values are always valid for the overlay variables.

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 name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default name is RADAR. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, RADAR1.

See also: "About Filename Indexing" on page 99

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 also: NROWS=

Featured in: Example 5 on page 1440

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.

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 also: NCOLS=

Featured in: Example 5 on page 1440

ORDERACROSS=FREQ | DATA | INTERNAL | FORMATTED | EXTERNAL
specifies the display order for the values of the ACROSS= option variable.

Restriction: Not supported by ActiveX

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.

Featured in: Example 3 on page 1438, Example 7 on page 1443, and Example 8 on page 1444

SPEED=*speed-variable*
specifies the wind speed in windrose 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

SPKLABEL=CATEGORY | NONE
labels the chart spokes with the category of the variable that is being charted. NONE suppresses the labels. The default is CATEGORY; however, if the STARLEGEND= option is specified, the default is NONE.

Default: CATEGORY

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 should also specify the OVERLAYVAR= 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=

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=

Featured in: Example 3 on page 1438

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 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 also: STAROUTRADIUS=

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, with 1 corresponding to 12 o'clock. Legend entries are assigned in clockwise order.
DEGREES	identifies the vertex variables by angles in degrees, with 0 degrees corresponding to 12 o'clock.
NONE	suppresses the legend. This is the default.

Featured in: Example 4 on page 1439

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 quotes. The default label is *Vertices:*.

Featured in: Example 4 on page 1439

STAROUTRADIUS=*value*
specifies 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 also: STARINRADIUS=

STARSTART=*value*
specifies the vertex angle for the first variable that is specified on the CHART statement. Vertex angles for the remaining variables are uniformly 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 also: "Calculating Weighted Statistics" on page 1420

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 quotes. The default label is *Tiles:*.

Restriction: Not supported by ActiveX

WFRAME=*n*
specifies the width in pixels for the frame lines.

Alias: WAXIS=

Default: 1

Restriction: Not supported by ActiveX

WINDROSE
specifies creation of a windrose chart. The windrose chart is named for charts of wind speed and direction. Windrose charts are a type of histogram which are useful when the extreme values of the histogram's midpoint variable are related. Typical applications include histograms involving direction, clock time, or other cyclical values.

Featured in: Example 7 on page 1443

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= option.

Alias: WSTARCIRCLE=

Default: 1

Restriction: This option is ignored unless the STARCIRCLES= 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

Featured in: Example 6 on page 1441

Examples

Note: 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 HTML;** ***procedure-code*****; ODS HTML CLOSE;**). See "RUN-Group Processing" on page 56 for more information. △

Example 1: Generating the Data Set for the GRADAR Examples

Procedure features: Data set generation
Sample library member: GRRDATA

Most of the GRADAR procedure examples use the FAILURE data set. You must submit this code before you can run any of the examples that use the FAILURE data set.

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, while 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.

```
data failure;
   label  Cause   = "Cause of Failure" ;
   input  Process $ 1-9 Day $ 13-19 Cause $ 23-36 Count 40-41;
   datalines;
Process A   March 1   Contamination    15
Process A   March 1   Corrosion         2
Process A   March 1   Doping            1
Process A   March 1   Metallization     2
```

```
Process A   March 1   Miscellaneous     3
Process A   March 1   Oxide Defect      8
Process A   March 1   Silicon Defect    1
Process A   March 2   Contamination    16
Process A   March 2   Corrosion         3
Process A   March 2   Doping            1
Process A   March 2   Metallization     3
Process A   March 2   Miscellaneous     1
Process A   March 2   Oxide Defect      9
Process A   March 2   Silicon Defect    2
Process A   March 3   Contamination    20
Process A   March 3   Corrosion         1
Process A   March 3   Doping            1
Process A   March 3   Metallization     0
Process A   March 3   Miscellaneous     3
Process A   March 3   Oxide Defect      7
Process A   March 3   Silicon Defect    2
Process A   March 4   Contamination    12
Process A   March 4   Corrosion         1
Process A   March 4   Doping            1
Process A   March 4   Metallization     0
Process A   March 4   Miscellaneous     0
Process A   March 4   Oxide Defect     10
Process A   March 4   Silicon Defect    1
Process A   March 5   Contamination    23
Process A   March 5   Corrosion         1
Process A   March 5   Doping            1
Process A   March 5   Metallization     0
Process A   March 5   Miscellaneous     1
Process A   March 5   Oxide Defect      8
Process A   March 5   Silicon Defect    2
Process B   March 1   Contamination     8
Process B   March 1   Corrosion         2
Process B   March 1   Doping            1
Process B   March 1   Metallization     4
Process B   March 1   Miscellaneous     2
Process B   March 1   Oxide Defect     10
Process B   March 1   Silicon Defect    3
Process B   March 2   Contamination     9
Process B   March 2   Corrosion         0
Process B   March 2   Doping            1
Process B   March 2   Metallization     2
Process B   March 2   Miscellaneous     4
Process B   March 2   Oxide Defect      9
Process B   March 2   Silicon Defect    2
Process B   March 3   Contamination     4
Process B   March 3   Corrosion         1
Process B   March 3   Doping            1
Process B   March 3   Metallization     0
Process B   March 3   Miscellaneous     0
Process B   March 3   Oxide Defect     10
Process B   March 3   Silicon Defect    1
Process B   March 4   Contamination     2
Process B   March 4   Corrosion         2
```

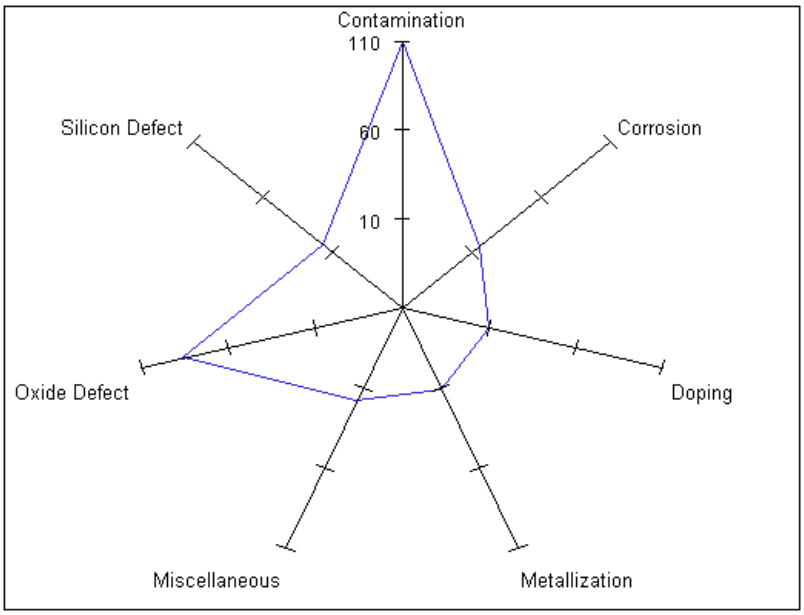
```
Process B   March 4   Doping           1
Process B   March 4   Metallization    0
Process B   March 4   Miscellaneous    3
Process B   March 4   Oxide Defect     7
Process B   March 4   Silicon Defect   1
Process B   March 5   Contamination    1
Process B   March 5   Corrosion        3
Process B   March 5   Doping           1
Process B   March 5   Metallization    0
Process B   March 5   Miscellaneous    1
Process B   March 5   Oxide Defect     8
Process B   March 5   Silicon Defect   2
run;
```

Example 2: Producing a Basic Radar Chart

Procedure features:
FREQ=

Data set: FAILURE (see Example 1 on page 1435)

Sample library member: GRRBASIC



In a radar chart, the vertices are determined by the levels of a single variable, which is specified on 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.

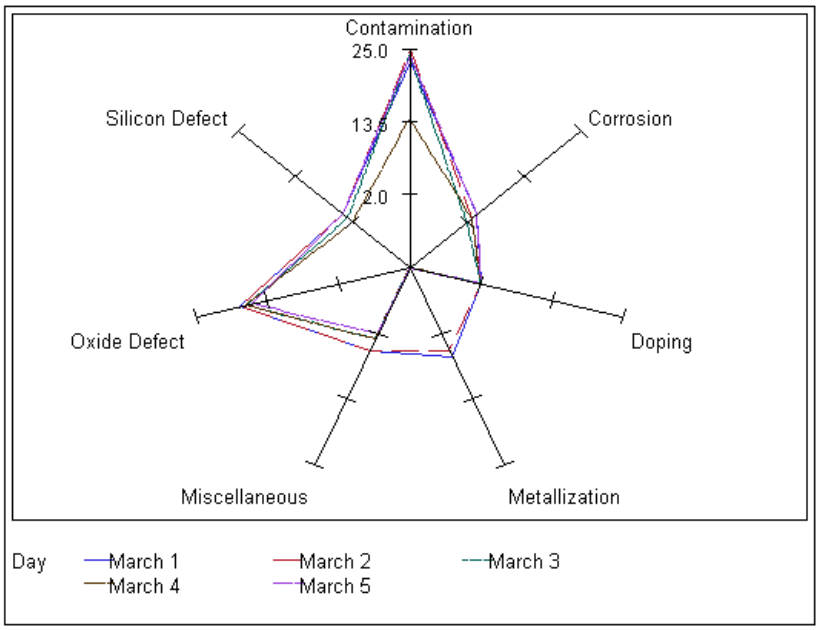
```
goptions reset=all;

proc gradar data=failure;
    chart cause / freq=count;
run;
quit;
```

Example 3: Overlaying Radar Charts

Procedure features:
FREQ=
OVERLAYVAR=

Data set: FAILURE (see Example 1 on page 1435)

Sample library member: GRROVER



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 on 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 the following example, the OVERLAY= option specifies variable DAY as the overlay variable.

```
goptions reset=all border;

proc gradar data=failure;
```

```
    chart cause / freq=count
        overlayvar=day;
run;
quit;
```

Example 4: Tiling Radar Charts

Procedure features:

ACROSSVAR=

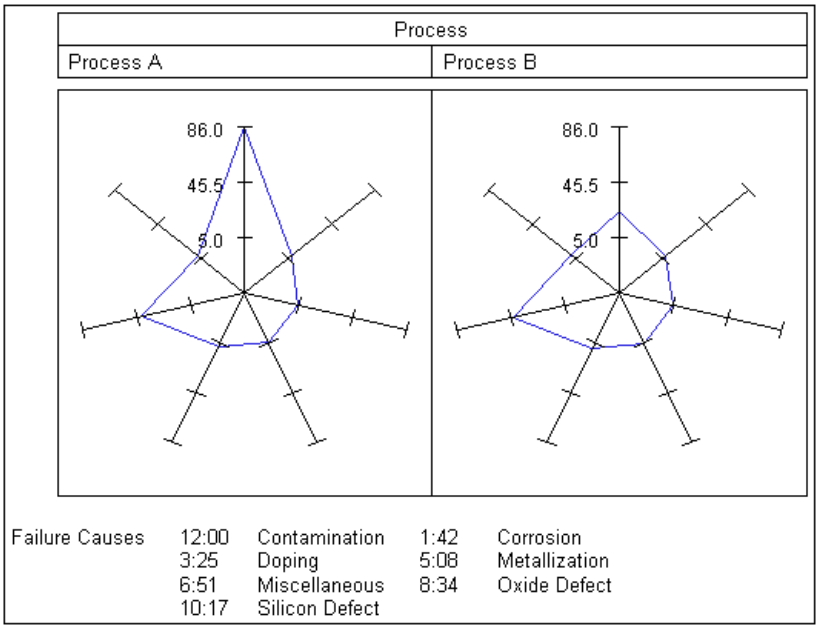
FREQ=

STARLEGEND=

STARLEGENDLAB=

Data set: Example 1 on page 1435

Sample library member: GRRTILE



As an alternative to overlaying multiple radar charts (see Example 3 on page 1438), you can *tile* charts horizontally, vertically, or in both directions (see Example 5 on page 1440) 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.

This example features the following options:

- 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;

proc gradar data=failure;
    chart cause / acrossvar=process
        freq=count
        starlegend=clock
        starlegendlab="Failure Causes";
run;
quit;
```

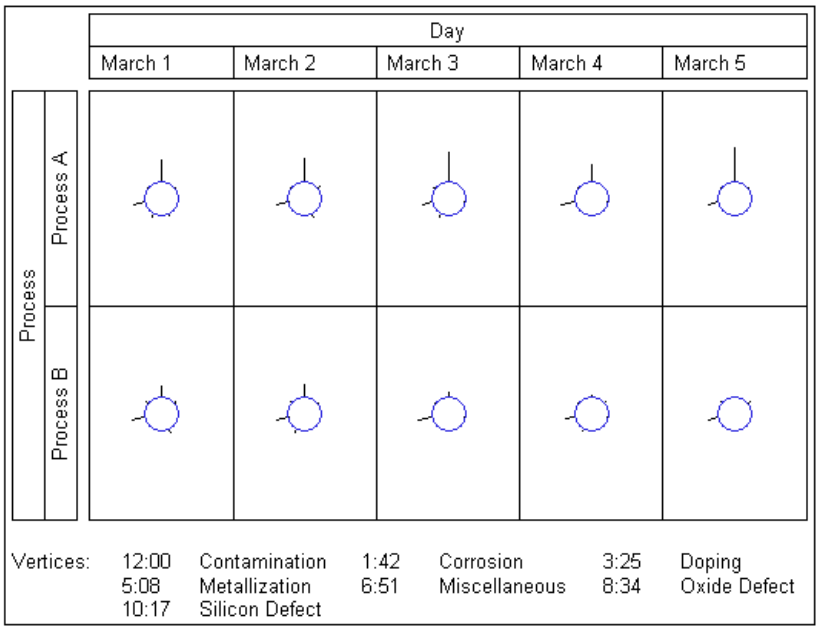
Example 5: Using Multiple Classification Variables in Radar Charts

Procedure features:

ACROSSVAR=
DOWNVAR=
FREQ=
STARTYPE=
NCOLS=
NROWS=
STARLEGEND=

Data set: FAILURE (see Example 1 on page 1435)

Sample library member: GRRTWOWY



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.

This example features the following options:

- The ACROSSVAR= option specifies variable DAY as the variable whose values determine the rows in the chart matrix.
- DOWNVAR= specifies variable PROCESS as the variable whose values determine the columns in the chart matrix.
- STARTYPE= determines that the stars are displayed with rays emanating from the inner circle.
- NROWS= and NCOLS= specify the number of rows and columns in the chart.
- STARLEGEND= CLOCK generates a legend that identifies spoke positions. Value CLOCK determines that the positions are identified using a clock metaphor.

```
goptions reset=all border;

proc gradar data=failure;
    chart cause / acrossvar=day
        downvar=process
        freq=count
        startype=spoke
        nrows=2 ncols=5
        starlegend=clock;
run;
quit;
```

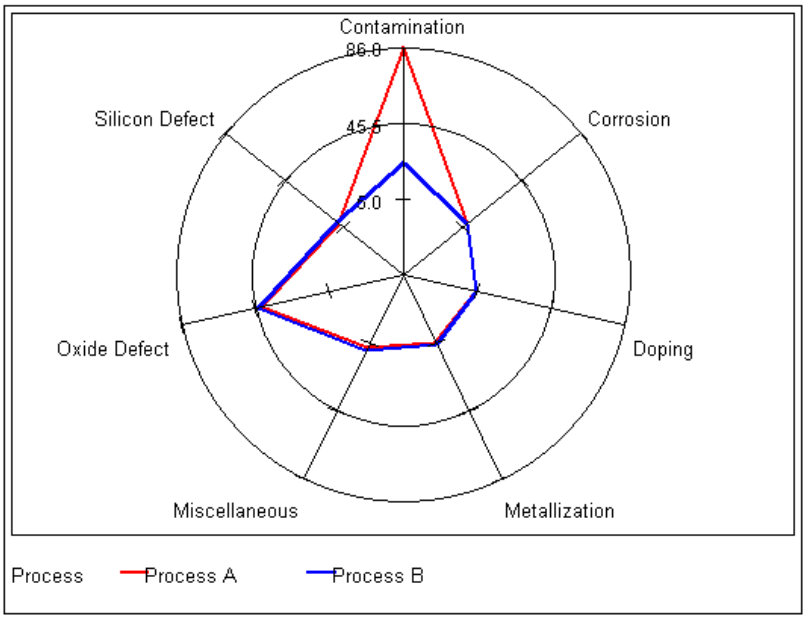
Example 6: Modifying the Appearance of Radar Charts

Procedure features:

CSTARS=
FREQ=
LSTARS=
OVERLAYVAR=
STARCIRCLES=
WSTARS=

Data set: FAILURE (see Example 1 on page 1435)

Sample library member: GRRAPEAR



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.

This example features the following options:

- CSTARS= specifies a different color for each of the star outlines in the chart.
- WSTARS= specifies the width of the line for each star outline.
- LSTARS= specifies a solid line as the line style for each star outline.
- STARCIRCLES= 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.

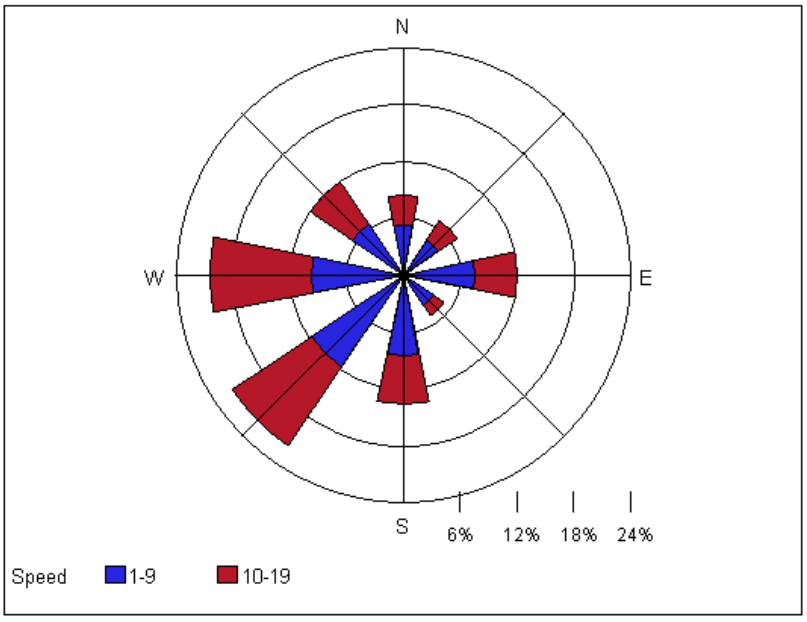
```
goptions reset=all border;

proc gradar data=failure;
    chart cause /  freq=count
       overlayvar=process
       cstars=(red, blue)
       wstars=2 2
       lstars=1 1
       starcircles=(0.5 1.0);
run;
quit;
```

Example 7: Creating a Windrose Chart

Procedure features:

NOFRAME
SPEED=
SUMVAR=
WINDROSE

Sample library member: GRRWNDRS



The windrose chart is named for charts of wind speed and direction. Windrose charts are a type of histogram which are useful when the extreme values of the histogram's midpoint variable are related. Typical applications include histograms involving direction, clock time, or other cyclical values.

```
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
```

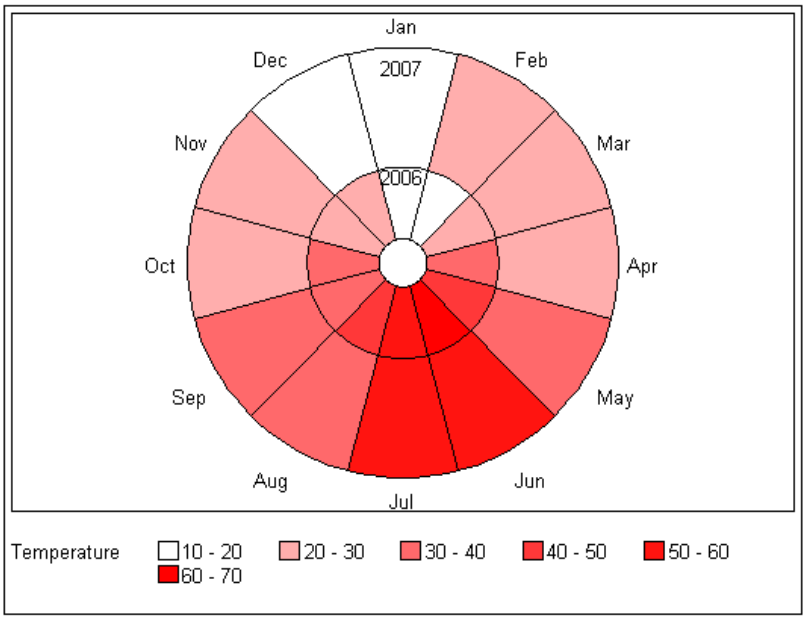
```
        windrose
        speed=speed
        noframe;
run;
quit;
```

Example 8: Creating a Calendar Chart

Procedure features:

FREQ=
CALENDAR
CSTARS=
OVERLAYVAR=

Sample library member: GRRCALEN



The CALENDAR option produces a radar chart displaying 12 equal-sized segments, one for each month of the year JAN through DEC. 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
       cstars=red;
    run;
quit;
```

CHAPTER

48

The GREDUCE Procedure

Overview

The GREDUCE procedure processes map data sets so that they can draw simpler maps with fewer boundary points. It creates an output map data set that contains all of the variables in the input map data set plus a new variable named DENSITY. For each observation in the input map data set, the procedure determines the significance of that point for maintaining a semblance of the original shape and gives the observation a corresponding DENSITY value.

You can then use the value of the DENSITY variable to create a subset of the original map data set. The observations in the subset can draw a map that retains the overall appearance of the original map but contains fewer points, requires considerably less storage space, and can be drawn much more quickly.

GREDUCE does not produce any graphics output. Instead, it produces an output map data set that can become either

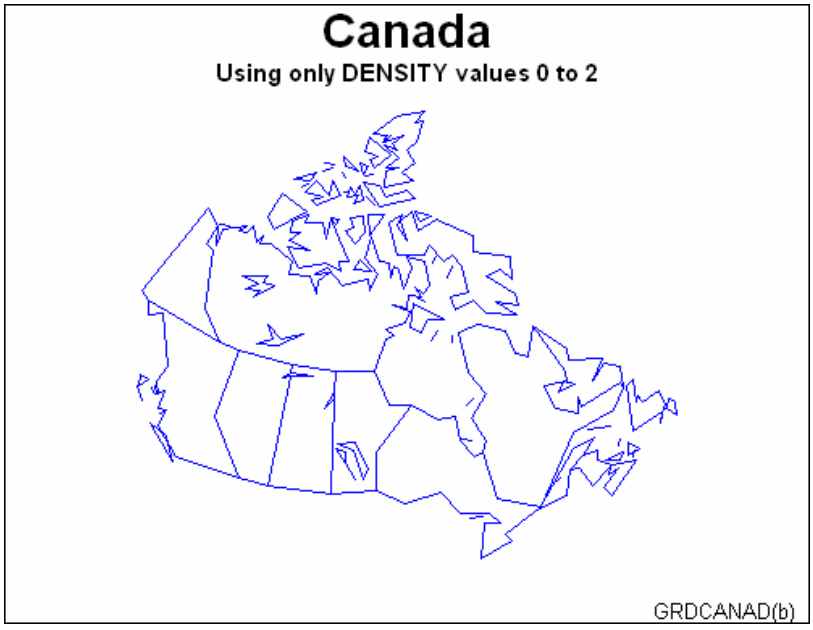
- the input map data set for the GMAP procedure
- the input map data set for a DATA step that removes points from the map.

Figure 48.1 on page 1448 and Figure 48.2 on page 1448 illustrate the effect of reduction on a typical map data set. Figure 48.1 on page 1448 uses observations with all DENSITY values as input to the GMAP procedure.

Figure 48.1 CANADA2 Map before Reduction (GRDCANAD(a))

Figure 48.2 on page 1448 uses only those observations with a DENSITY value of 0 to 2 as input to the GMAP procedure.

Figure 48.2 CANADA2 Map after Reduction (GRDCANAD(b))



The program for these maps is in Example 1 on page 1454.

The reduced map shown in Figure 48.2 on page 1448 retains the overall shape of the original but requires only 463 observations compared to the 4302 observations that are needed to produce the map in Figure 48.1 on page 1448.

Note: Many of the map data sets that are supplied by SAS Institute already have been processed by GREDUCE. If the map data set contains a DENSITY variable, you do not need to process the data set using GREDUCE. △

See also Chapter 49, "The GREMOVE Procedure," on page 1459 for more information on how to

- combine groups of unit areas into larger unit areas to create regional maps
- remove some of the boundaries in a map and create a subset of a map that combines the original areas.

Concepts

About the Input Map Data Set

The input map data set must be a traditional map data set and contain these variables:

- a numeric variable named X that contains the horizontal coordinates of the map boundary points.
- a numeric variable named Y that contains the vertical coordinates of the map boundary points.
- one or more *identification variables* that uniquely identify the unit areas in the map. These variables are listed in the ID statement.

It also can contain

- one or more variables that identify groups of unit areas (for BY-group processing)
- the variable SEGMENT, which distinguishes nonconterminous segments of the unit areas.

Any other variables in the input map data set do not affect the GREDUCE procedure.

About Unmatched Area Boundaries

If you are using map data sets in which area boundaries do not match precisely (for example, if the boundaries were digitized with a different set of points), PROC GREDUCE will not be able to identify common boundaries properly, and this results in abnormalities in your maps. These abnormalities include mismatched borders, missing vertex points, stray lines, gaps, and distorted polygons.

If the points in the area boundaries match up except for precision differences, round each X and Y value in your map data set accordingly, using the DATA step function ROUND before using PROC GREDUCE. (See *SAS Language Reference: Dictionary* for information on the ROUND function.)

For example, if the map data set APPROX has horizontal and vertical coordinate values for interior boundaries of unit areas that are exactly equal only to three decimal places, then this DATA step creates a new map data set, EXACT, that will be better suited for use with PROC GREDUCE:

```
data exact;
      set approx;
      if x ne . then x=round(x,.001);
      if y ne . then y=round(y,.001);
```

```
run;
```

See “About Map Data Sets” on page 1244 for additional information on map data sets.

Procedure Syntax

Requirements: Exactly one ID statement is required.

Reminder: The procedure can include the BY statement.

PROC GREDUCE <*option(s)*>;

ID *id-variable(s)*;

PROC GREDUCE Statement

Identifies the input and output map data sets. Optionally specifies the reduction criteria.

Requirements: An input map data set is required.

Syntax

PROC GREDUCE <*option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- data set options:

  DATA=*input-map-data-set*

  OUT=*output-map-data-set*

- level options:

  E1=*min-distance*

  E2=*min-distance*

  E3=*min-distance*

  E4=*min-distance*

  E5=*min-distance*

  N1=*max-points*

  N2=*max-points*

  N3=*max-points*

  N4=*max-points*

  N5=*max-points*

Options

DATA=*input-map-data-set*
identifies the map data set that you want to process. By default, the procedure uses the most recently created SAS data set.

See also: "About the Input Map Data Set" on page 1449 and "SAS Data Sets" on page 54.

E1=*min-distance*
E2=*min-distance*
E3=*min-distance*
E4=*min-distance*
E5=*min-distance*
specify the minimum distance that a point must lie from a straight line segment to be included at density level 1, 2, 3, 4, or 5, respectively. That is, in a reduced curve of three points, the middle point is at least a distance that is *min-distance* from a straight line between the two outside points.

Express *min-distance* values in the units for the coordinate system of the input map data set. For example, if the input map data set contains coordinates that are expressed in radians, express the *min-distance* values in radians.

Specify the E*n*= values in decreasing order. For example, the E2= value should be less than the E1= value and so on.

N1=*max-points*
N2=*max-points*
N3=*max-points*
N4=*max-points*
N5=*max-points*
specify that for density level 1, 2, 3, 4, or 5, the boundary of a unit area should contain no more than *max-points* points.

Specify the N*n*= values in increasing order. For example, the N2= value should be greater than or equal to the N1= value and so on.

By default, if you omit N*n*= and E*n* = , the GREDUCE procedure calculates values for the five N*n* = parameters using this formula:

$$\mathrm{N}n = n^2 \times \mathrm{N}_{\max}/36$$

Here $\mathrm{N}_{\max}$ is the maximum number of points in any unit area in the input map data set. However, the restriction that the number of points for any level cannot be less than the number of points in level 0 still applies.

OUT=*output-data-set*
names the new map data set, which contains all of the observations and variables in the original map data set plus the new DENSITY variable. If the input map data set contains a variable named DENSITY, the GREDUCE procedure replaces the values of the variable in the output map data set. The original values of the DENSITY variable from the input map data set are not included in the output map data set.

By default, the GREDUCE procedure names the new data set that uses the DATA*n* naming convention. That is, the procedure uses the name WORK.DATA*n*, where *n* is the next unused number in sequence. Thus, the first automatically named data set is DATA1, the second is DATA2, and so on.

ID Statement

Identifies the variable or variables that define the hierarchy of the current unit areas in the input map data set.

Requirements: At least one *id-variable* is required.

Featured in: Example 1 on page 1454.

Syntax

ID *id-variable(s)*;

Required Arguments

id-variable(s)
specifies one or more variables in the input map data set that identify unit areas. *Id-variable(s)* can be either numeric or character.

Each group of observations with a different ID variable value is evaluated as a separate unit area.

Using the GREDUCE Procedure

Specifying Density Levels

GREDUCE uses default criteria for determining the appropriate DENSITY variable value for each observation in the input map data set. If you do not want to use the default criteria, use PROC GREDUCE options to select

- the maximum number of observations for each DENSITY level
- the minimum distance that an intermediate point must lie from a line between two end points to be included in the level.

If you do not explicitly specify criteria, the procedure computes and uses default values.

GREDUCE creates seven density levels, numbered 0 through 6. Specify criteria for density levels 1 through 5. You cannot define criteria for level 0, which is reserved for map vertex points, such as common corners of unit areas. You also cannot define criteria for level 6, which is assigned to those points that do not meet the criteria for any lower level.

Specify the maximum number of observations per density level using N*n*= in the PROC GREDUCE statement, and specify the minimum point distance using E*n*= . You must have knowledge of the X and Y variable values in the particular input map data set to determine appropriate values for E*n*=. See the En= and Nn= option on page 1451 for details.

Figure 48.3 on page 1453 illustrates how to use the minimum distance parameter to determine which points belong in a particular density level. At density level *n*, only point C lies at a distance greater than the E*n*= value (70) from a line between points A

and B. Thus, after reduction only point C remains between points A and B at density level *n*, and the resulting reduced boundary is shown in Figure 48.4 on page 1453. See Douglas and Peucker (1973) for details of the algorithm used.

Figure 48.3 Points in Data Set before Reduction

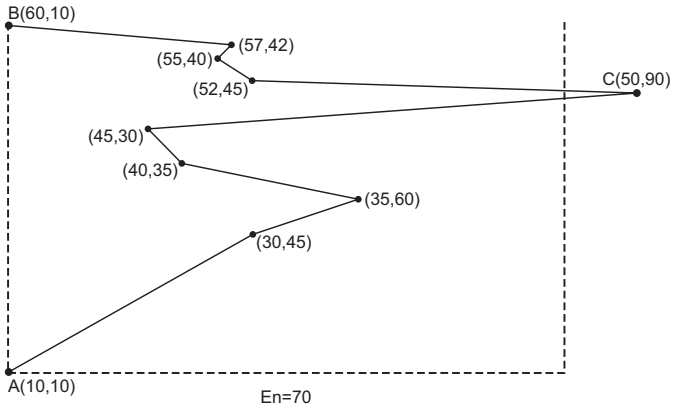
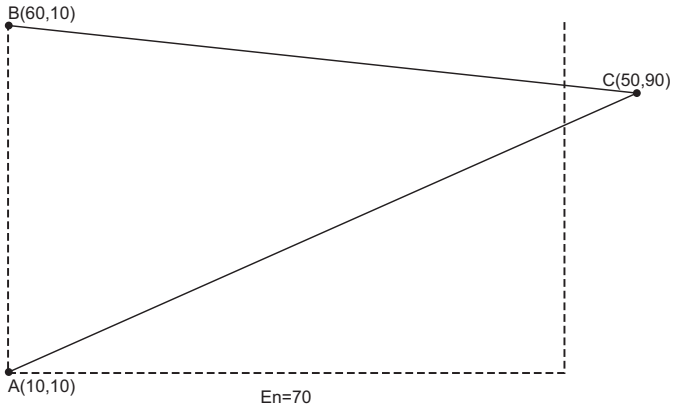


Figure 48.4 Points in Data Set at Density n after Reduction



GREDUCE uses the usual Euclidean distance formula to determine the distance between points. For example, the distance d between the points (x_0,y_0) and (x_1,y_1) is GREDUCE uses the usual Euclidean distance formula to determine the distance between points. For example, the distance d between the points (x_0,y_0) and (x_1,y_1) is

$$d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

If this distance function is not suitable for the coordinate system in your input map data set, transform the X and Y values to an appropriate coordinate system before using GREDUCE. An example of inappropriate coordinates is latitude and longitude values around one of the poles. In this case, the data values should be projected before they are reduced. See Chapter 46, "The GPROJECT Procedure," on page 1395 for more information on map projection.

If you specify both N*n*= and E*n*= values for a density level, GREDUCE attempts to satisfy both criteria. However, the number of points for any level is never reduced

below the number of points in density level 0. If you specify a combination of N*n*= or E*n*= values such that the resulting DENSITY values are not in order of increasing density, a note is printed in the SAS log, and the DENSITY values are calculated in increasing order of density.

Subsetting a Map Data Set

A map data set that is processed by GREDUCE does not automatically result in a map that uses fewer points. By default, the GMAP procedure produces a map that uses all of the points in the map data set, even if the data set has been processed by the GREDUCE procedure. To decrease the number of points that produce the map, you must create a subset of the original data set using a DATA step or the WHERE= data set option. For example, to create a subset of a map that uses only the DENSITY values 0, 1, and 2, use this DATA step:

```
data smallmap;
   set map;
   if density <= 2;
run;
```

Alternatively, you can use WHERE= in the PROC GMAP statement:

```
proc gmap map=map(where=(density<=2))
          data=response;
```

Note: GREDUCE does not reduce the size of the output map data set compared to the input map data set. By default, the output map data set from PROC GREDUCE will be larger than the input map data set because it contains all of the variables and observations from the original data set, with the addition of the DENSITY variable if it was not present in the original data set. If the input map data set already had a DENSITY variable, the output map data set will be the same size as the input map data set. △

Examples

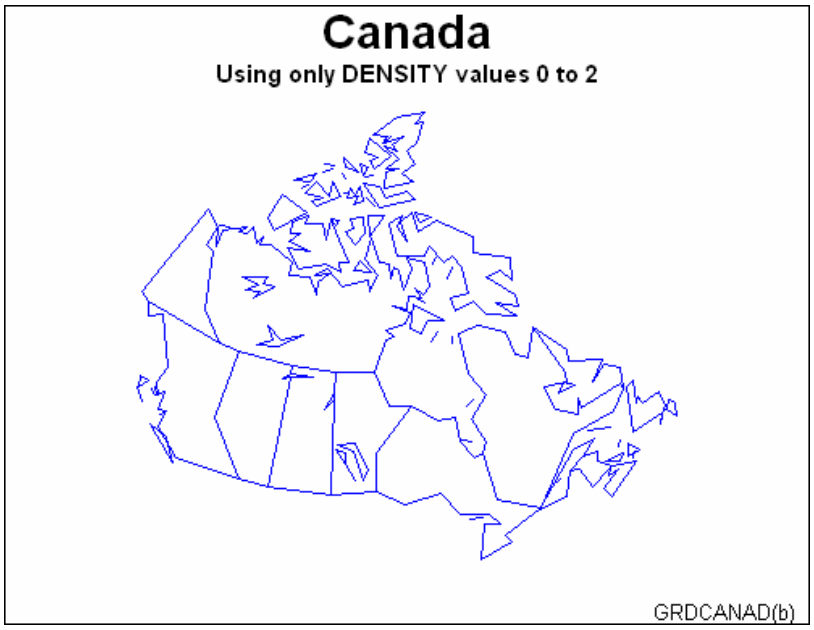
The following example illustrates major features of the GREDUCE procedure. Because the example uses one of the map data sets that are supplied with SAS/GRAPH, you may need to replace *SAS-data-library* in the LIBNAME statement with the actual location of the SAS data library that contains the Institute-supplied map data sets on your system. Contact your SAS Software Consultant for the location of the map data sets at your site. If your site automatically assigns the libref MAPS to the SAS data library that contains the Institute-supplied map data sets, delete the LIBNAME statement in this example.

Example 1: Reducing the Map of Canada

Procedure features:
ID statement

Sample library member: GRDCANAD

In this example, the GREDUCE procedure creates the DENSITY variable for the CANADA2 map data set that is provided with SAS/GRAPH. First, the map is displayed at its original density by using the GMAP procedure. Second, the map is displayed by using density values of 0 to 2.



Set the graphics environment.

```
goptions reset=all border;
```

Define titles and footnotes for the first map.

```
title1 "Canada";
title2 "Using all DENSITY values";
footnote j=r "GRDCANAD(a) ";
```

Define pattern characteristics.

```
pattern value=mempty repeat=12 color=blue;
```

Show the unreduced map. The ID statement specifies the variable in the map data set that defines unit areas.

```
proc gmap map=maps.canada2 data=maps.canada2 all;
  id province;
  choro province / nolegend;
run;
```

The GREDUCE procedure creates a new map data set, CAN2, containing a DENSITY variable. The ID statement specifies the variable in the map data set that defines unit areas.

```
proc greduce data=maps.canada2 out=can2;
   id province;
run;
```

Define title and footnote for the second map.

```
title2 h=4 "Using only DENSITY values 0 to 2";
footnote2 j=r "GRDCANAD(b) ";
```

Show reduced map with density levels 0-2. The DENSITY= option specifies the density levels that are used.

```
proc gmap map=can2
          data=can2 all density=2;
   id province;
   choro province / nolegend;
run;
quit;
```

References

Douglas, D.H. and Peucker, T.K. (1973), "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature," *The Canadian Cartographer*, 10, 112–122.

CHAPTER

49

The GREMOVE Procedure

Overview

The GREMOVE procedure processes a map data set that is used as input. It does not produce any graphics output. Instead, it produces an output data set that typically becomes the input map data set for the GMAP procedure (see Chapter 43, "The GMAP Procedure," on page 1239). The GREMOVE procedure combines unit areas defined in a map data set into larger unit areas by removing shared borders between the original unit areas. For example, Figure 49.1 on page 1460 and Figure 49.2 on page 1460 show combined unit areas in a typical map data set by removing state boundaries to create regional census divisions.

Figure 49.1 Map before Removing Borders (GRMUSMAP(a))

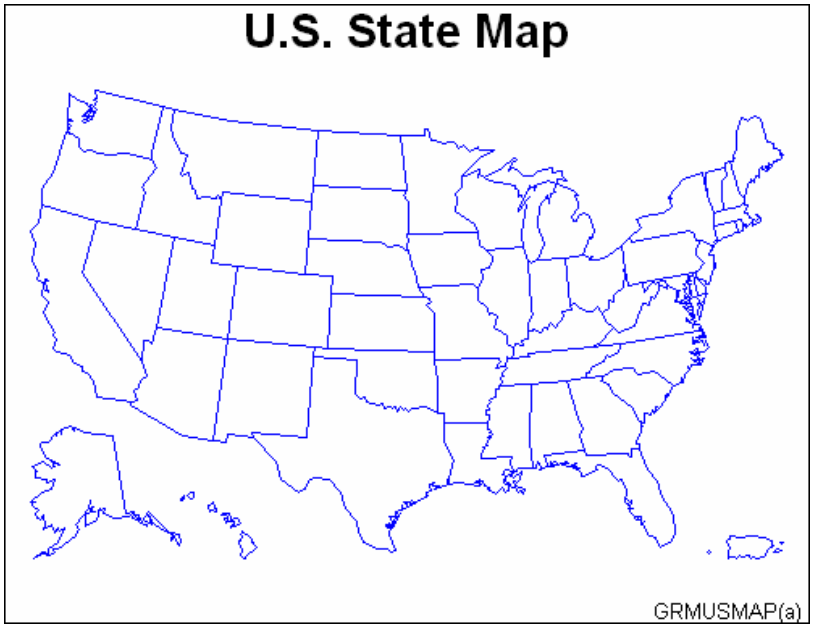
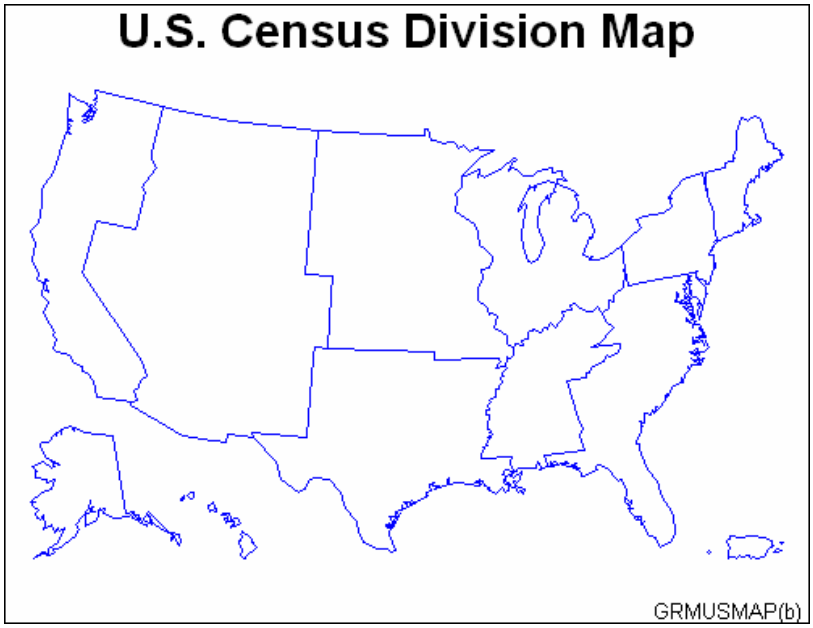


Figure 49.2 Map after Removing Borders (GRMUSMAP(b))



The program for these maps is shown in Example 1 on page 1465.

Concepts

The GREMOVE procedure processes the input map data set to remove internal boundaries and creates a new map data set. The PROC GREMOVE statement

identifies the input and output map data sets. The ID statement identifies the variable or variables in the input map data set that define the current unit areas. The BY statement identifies the variable or variables in the input map data set that define the new unit areas.

About the Input Map Data Set

The input map data set must be in traditional map data set format (see "About Map Data Sets" on page 1244) and it must contain these variables:

- a numeric variable named X that contains the horizontal coordinates of the map boundary points.
- a numeric variable named Y that contains the vertical coordinates of the map boundary points.
- one or more variables that uniquely identify the current unit areas in the map. These variables are listed in the ID statement. Each group of observations with a different ID variable value is evaluated as a separate unit area.
- one or more variables that identify the new unit areas to be created in the output map data set. These variables are listed in the BY statement.

It might also contain the variable SEGMENT, which is used to distinguish non-conterminous segments of the same unit areas. Other variables might exist in the input map data set, but they do not affect the GREMOVE procedure and they will not be carried into the output map data set.

About the Output Map Data Set

The output map data set contains the newly defined unit areas. These new unit areas are created by removing all interior line segments from the original unit areas. All variables in the input map data set except X, Y, SEGMENT, and the variables listed in the BY statement are omitted from the output map data set.

The output map data set might contain missing X, Y coordinates to construct any polygons that have enclosed boundaries (like lakes or combined regions that have one or more hollow interior regions).

The SEGMENT variable in the output map data set is ordered according to the size of the bounding box around the polygon that it describes. A SEGMENT value of 1 describes the polygon whose bounding box is the largest in the unit area, and each additional SEGMENT value describes a smaller polygon. This information is useful for removing small polygons that clutter up maps.

All current unit areas with common BY-variable value(s) are combined into a single unit area in the output map data set. The new unit area contains

- all boundaries that are not shared, such as islands and lakes
- all boundaries that are shared by two different BY groups.

About Unmatched Area Boundaries

If you are using map data sets in which area boundaries do not match precisely (for example, if the boundaries were digitized with a different set of points), PROC GREMOVE will not be able to identify common boundaries properly, resulting in abnormalities in your output data set.

If the points in the area boundaries match up except for precision differences, before using PROC GREMOVE round each X and Y value in your map data set accordingly,

using the DATA step function ROUND. See *SAS Language Reference: Dictionary* for information on the ROUND function.

For example, if you have a map data set named APPROX in which the horizontal and vertical coordinate values for interior boundaries of unit areas are exactly equal only to three decimal places, this DATA step creates a new map data set, EXACT, that is better suited for use with the GREMOVE procedure:

```
data exact;
   set approx;
   if x ne . then x=round(x,.001);
   if y ne . then y=round(y,.001);
run;
```

You can also use the FUZZ option to specify a level of tolerance so that the boundaries do not need to match precisely.

Procedure Syntax

Requirements: The BY and ID statements are required.

PROC GREMOVE <DATA=*input-map-data-set*>
<FUZZ=*fuzz-factor*>
<OUT=*output-map-data-set*>
<NODECYCLE>;

BY <DESCENDING>*variable-l*
<...<DESCENDING>*variable-n*>
<NOTSORTED>;

ID *variable(s)*;

PROC GREMOVE Statement

Identifies the input and output map data sets.

Requirements: An input map data set is required.

Syntax

PROC GREMOVE <DATA=*input-map-data-set*>
<FUZZ=*fuzz-factor*>
<OUT=*output-map-data-set*>
<NODECYCLE>;

Options

DATA=*input-map-data-set*
specifies the map data set that is to be processed. By default, the procedure uses the most recently created SAS data set. The GREMOVE procedure expects the observations in the input map data set to be sorted in ascending order of the BY-variable values.

See also: "About the Input Map Data Set" on page 1461 and "SAS Data Sets" on page 54.

Featured in: Example 2 on page 1469.

FUZZ=*fuzz-factor*
specifies a tolerance for possible error in the data. This allows for points that are very close but not quite equal to be considered as the same point. The fuzz-factor can be any non-negative number. A fuzz-factor of 0.0 would indicate that the points have to be exactly the same. The unit represented by the fuzz-factor (degrees, radians, feet, meters, kilometers, miles) is the same as that represented by the X and Y values of the points.

The error is computed the same in both X and Y directions using the following formula:

```
Point_is_equal = (ABS(x1 - x2) <= fuzz-factor) && (ABS(y1 - y2) <= fuzz-factor)
```

NODECYCLE | NC
tells PROC GREMOVE to use a topological algorithm for closing the resulting polygons. By default, PROC GREMOVE simply removes internal boundaries without using any polygon information. This might cause errors in closing the resulting polygons in certain cases—specifically when two resulting polygons intersect at a single point. Using a topological algorithm allows PROC GREMOVE to traverse the resulting polygons for proper closure of the polygons. When the single point intersection is encountered, the algorithm uses the topology to correctly interpret which existing segment to choose in closing the polygon. The use of NODECYCLE, thus, requires that the data be topologically correct (that is, polygons do not overlap themselves or each other and there are no anomalies in the boundaries such as a repeated series of points).

Certain SAS/GRAPH procedures, such as PROC GREDUCE, which have no knowledge of topology and do not maintain topology, can produce topologically incorrect polygons. Therefore, it is recommended that you not use PROC GREDUCE if you are going to use PROC GREMOVE with NODECYCLE specified.

OUT=*output-data-set*
names the new map data set, which contains the coordinates of the new unit areas created by the GREMOVE procedure. By default, the GREMOVE procedure names the new data set using the DATA*n* naming convention. That is, the procedure uses the name WORK.DATA*n*, where *n* is the next unused number in sequence. Thus, the first automatically named data set is DATA1, the second is DATA2, and so on.

See also: "About the Output Map Data Set" on page 1461.

Featured in: Example 2 on page 1469.

BY Statement

Lists the variable or variables that identify the new unit areas.

Requirements: At least one variable is required.

See also: "BY Statement" on page 216.

Featured in: Example 1 on page 1465.

Syntax

BY <DESCENDING>*variable-l*
<...<DESCENDING>*variable-n*>
<NOTSORTED>;

Required Arguments

variable(s)
identifies one or more variables in the input map data set that define the new unit areas. *Variable(s)* can be either numeric or character.

The BY variables in the input map data set become the ID variables for the output map data set.

Options

DESCENDING
indicates that the input map data set is sorted in descending order. By default, the GREMOVE procedure expects all BY-variable values to appear in ascending order.

This option affects only the variable that immediately follows the option.

NOTSORTED
indicates that observations with the same BY-variable values are to be grouped as they are encountered without regard for whether the values are in alphabetical or numerical order. NOTSORTED can appear anywhere in the BY statement. It affects all of the variables that are specified in the statement. NOTSORTED overrides DESCENDING if both appear in the same BY statement.

Ordering Observations

To sort the input map data set, use the SORT procedure in Base SAS, for example

```
/* arrange the observations in desired order */
proc sort data=mapdata out=mapsort;
   by state;
run;
   /* remove the county boundaries              */
proc gremove data=mapsort out=newmap;
   by state;
```

```
   id county;
run;
```

Notice that the GREMOVE procedure uses the same BY statement as the SORT procedure.

See the *Base SAS Procedures Guide* for further information on the SORT procedure.

Note: If an observation is encountered for which the BY-variable value is out of the proper order, the GREMOVE procedure stops and issues an error message. △

ID Statement

Identifies the variable or variables that define the hierarchy of the current unit areas in the input map data set.

Requirements: At least one *id-variable* is required.

Featured in: Example 1 on page 1465.

Syntax

ID *id-variable(s)*;

Required Arguments

id-variable(s)
specifies one or more variables in the input map data set that identify the unit areas to be combined. These variables are not included in the output map data set. *Id-variable(s)* can be either numeric or character.

See also: "About the Input Map Data Set" on page 1461.

Examples

The following examples illustrate major features of the GREMOVE procedure.

Example 1: Removing State Boundaries from U.S. Map

Procedure features:
BY statement
ID statement

Other features:
SORT procedure
MERGE statement

LIBNAME statement

Sample library member: GRMUSMAP

This example processes the MAPS.US map data set, supplied with SAS/GRAPH, to produce a new map data set containing boundaries for the U.S. Bureau of the Census divisions. Because the MAPS.US map data set does not contain a variable to identify any unit area other than states, this example creates a map data set that contains the census divisions and that can be processed with the GREMOVE procedure.

The STATE variable in the MAPS.US data set, containing numeric FIPS codes for each state, is used as the BY-variable to merge the CBSTATES and MAPS.US data sets. Output 49.1 shows some of the variables that are present in the data set before using the GREMOVE procedure:

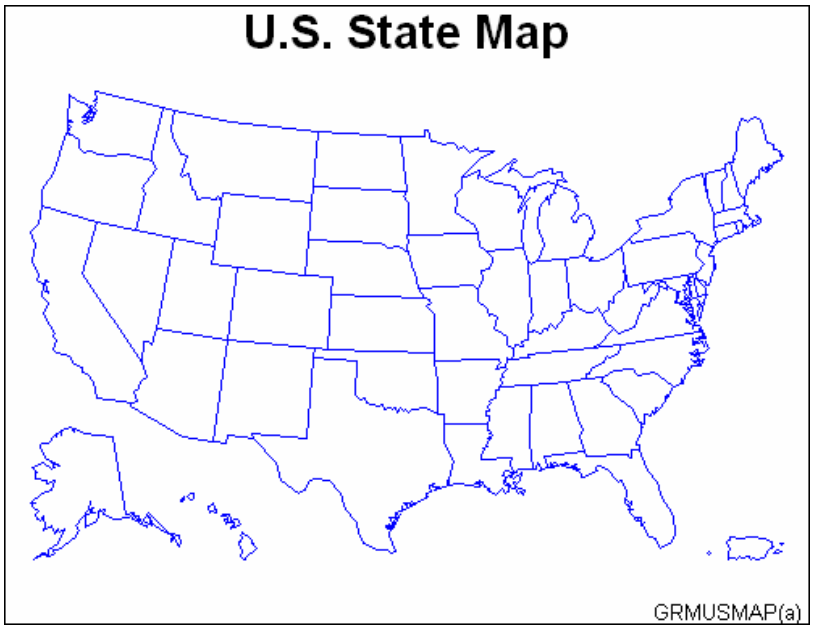
Output 49.1 The MAPS.US Data Set

MAPS.US Data Set

OBS	STATE	SEGMENT	X	Y
1	1	1	0.16175	-0.10044
2	1	1	0.12305	-0.10415
3	1	1	0.12296	-0.10678
.				
.				
.				
1524	56	1	-0.18757	0.15035
1525	56	1	-0.10158	0.13997
1526	56	1	-0.10398	0.11343

Figure 49.3 on page 1466 shows the map before processing:

Figure 49.3 Map before Removing Borders (GRMUSMAP(a))



Output 49.2 shows the variables that are present in the data set after you use the GREMOVE procedure. Notice that the new map data set contains a new variable called DIVISION:

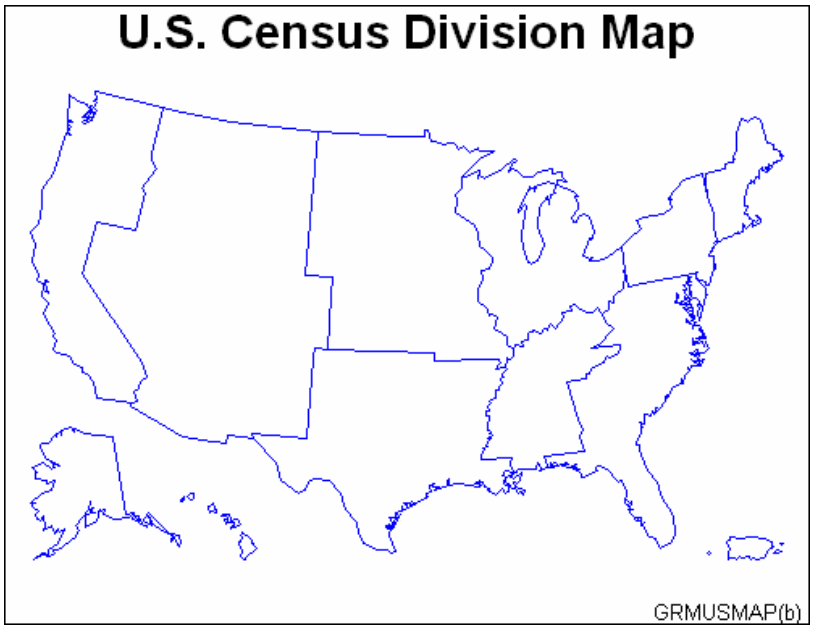
Output 49.2 The REMSTATE Data Set

REMSTATE Data Set

OBS	X	Y	SEGMENT	DIVISION
1	0.29825	0.17418	1	1
2	0.29814	0.17820	1	1
3	0.30206	0.18045	1	1
.				
.				
.				
1082	-0.18715	-0.16010	8	9
1083	-0.18747	-0.15971	8	9
1084	-0.18747	-0.15951	8	9

Figure 49.4 on page 1467 shows the new map after PROC GREMOVE has removed interior state boundaries.

Figure 49.4 Map after Removing Borders (GRMUSMAP(b))



Set the graphics environment.

```
goptions reset=all border;
```

Create data set CBSTATES. This data set includes a variable, DIVISION, that contains the number of the U.S. Bureau of the Census division for the state. This data step converts letter codes to numeric FIPS codes that match those in the STATE variable of MAPS.US.

```
data cbstates;
   length state 8 stcode $ 2 division 4;
   input stcode division @@;
   state=stfips(stcode);
   drop stcode;
   datalines;
CT 1 MA 1 ME 1 NH 1 RI 1 VT 1 PA 2 NJ 2 NY 2 IL 3 IN 3 MI 3 OH 3 WI 3 IA 4 KS 4
MN 4 MO 4 ND 4 NE 4 SD 4 DC 5 DE 5 FL 5 GA 5 MD 5 NC 5 PR 5 SC 5 VA 5 WV 5
AL 6 KY 6 MS 6 TN 6 AR 7 LA 7 OK 7 TX 7 AZ 8 CO 8 ID 8 MT 8 NM 8 NV 8 UT 8
WY 8 AK 9 CA 9 HI 9 OR 9 WA 9
;
```

Sort data set in FIPS-code order. Create a sorted data set, CBSORT. It can be properly match-merged with the MAPS.US map data set, which is already sorted in FIPS-code order.

```
proc sort data=cbstates out=cbsort;
   by state;
run;
```

Add DIVISION variable to map data set by merging the CBSORT data set with MAPS.US. Create a new map data set, USCB, that contains all of the state boundary coordinates from the MAPS.US data set plus the added variable DIVISION.

```
data uscb;
   merge cbsort maps.us;
   by state;
run;
```

Sort data set in DIVISION order. Sort USCB by the DIVISION variable to create the DIVSTATE data set.

```
proc sort data=uscb out=divstate;
   by division;
run;
```

Remove interior boundaries within divisions. BY specifies the variable, DIVISION, in the input map data set that identifies the new unit areas. ID specifies the variable, STATE, in the input map data set that identifies the current unit areas.

```
proc gremove data=divstate out=remstate;
   by division;
   id state;
run;
```

Define title and footnote for map.

```
title "U.S. State Map";
footnote j=r "GRMUSMAP(a) ";
```

Define pattern characteristics.

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

Show the original map.

```
proc gmap map=maps.us data=maps.us all;
   id state;
   choro state / nolegend levels=1;
run;
```

Define new title and footnote for map.

```
title "U.S. Census Division Map";
footnote j=r "GRMUSMAP(b) ";
```

Show the regional map. ID specifies the variable, DIVISION, that identifies the unit areas in the processed data set. CHORO specifies DIVISION as the response variable.

```
proc gmap map=remstate data=remstate all;
   id division;
   choro division / nolegend levels=1;
run;
quit;
```

Example 2: Creating an Outline Map of Africa

Procedure features:

PROC GREMOVE options:

DATA=
OUT=

Other features:

GMAP procedure

Sample library member: GRMAFRIC

This example processes the MAPS.AFRICA map data set, supplied with SAS/GRAPH, to produce a new map data set that contains no internal boundaries. This is done by adding a new variable, REGION, to the map data set and setting it equal to 1. Unit areas from the input map data set that have the same BY-variable value are

combined into one unit area in the output map data set. Output 49.3 shows some of the variables that are present in the original map data set:

Output 49.3 The MAPS.AFRICA Data Set

MAPS.AFRICA Data Set

OBS	ID	SEGMENT	X	Y
1	125	1	0.57679	1.43730
2	125	1	0.57668	1.43467
3	125	1	0.58515	1.42363
.				
.				
.				
3462	990	1	1.04249	0.50398
3463	990	1	1.04184	0.50713
3464	990	1	1.04286	0.50841

Figure 49.5 on page 1470 shows the map before processing:

Figure 49.5 Map before Removing Borders (GRMAFRIC(a))

The new AFRICA map data set is created with a new variable, REGION. Output 49.4 shows the variables that are present in the new map data set created by the GREMOVE procedure:

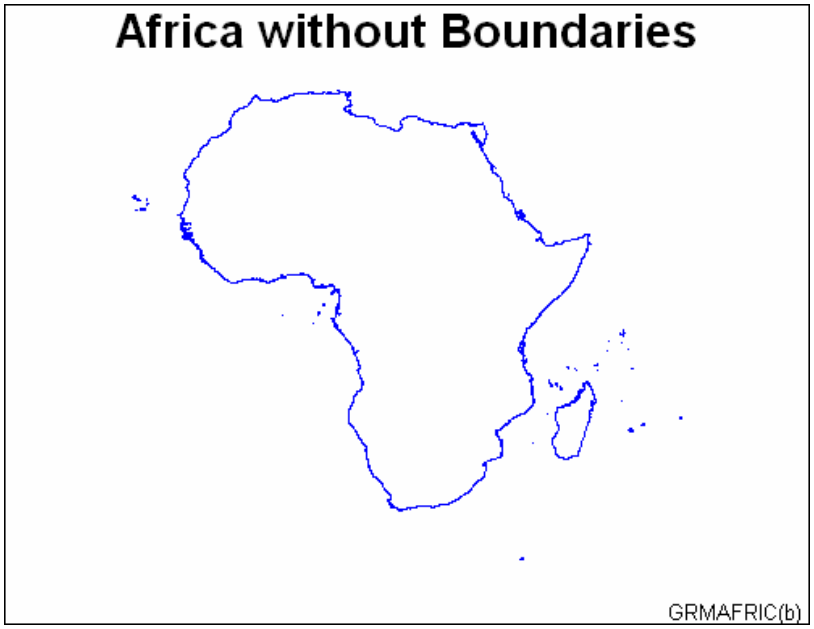
Output 49.4 The AFRICA Data Set

AFRICA Data Set

OBS	X	Y	SEGMENT	REGION
1	0.24826	1.02167	1	1
2	0.25707	1.02714	1	1
3	0.26553	1.03752	1	1
.				
.				
.				
982	1.19071	1.30043	3	1
983	1.18675	1.30842	3	1
984	1.18518	1.32822	3	1

Figure 49.6 on page 1471 shows the new map after PROC GREMOVE has removed all of the interior boundaries:

Figure 49.6 Map after Removing Borders (GRMAFRIC(b))



Set the graphics environment.

```
goptions reset=all border;
```

Create the NEWAF data set. This new map data set contains all the variables in the SAS/GRAPH supplied MAPS.AFRICA map data set plus the added variable REGION.

```
data newaf;
   set maps.africa;
   region=1;
run;
```

Remove the unit areas from the AFRICA data set. DATA= specifies the input map data set and OUT= specifies the output map data set. The input map data set has a variable called REGION that is used as the BY-variable to identify the new unit areas. The ID statement specifies the current unit areas from the input map data set.

```
proc gremove data=newaf out=africa;
   by region;
   id id;
run;
```

Define the title and footnote.

```
title "Africa with Boundaries";
footnote j=r "GRMAFRIC(a) ";
```

Define pattern characteristics.

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

Display the original map.

```
proc gmap map=maps.africa data=maps.africa all;
   id id;
   choro id / nolegend levels=1;
run;
```

Define a new title and footnote for the map.

```
title "Africa without Boundaries";
footnote j=r "GRMAFRIC(b) ";
```

Display the map with no boundaries. ID specifies the variable, REGION, that identifies the unit areas in the processed data set.

```
proc gmap data=africa map=africa;
   id region;
   choro region / nolegend levels=1;
run;
quit;
```

CHAPTER

50

The GREPLAY Procedure

Overview

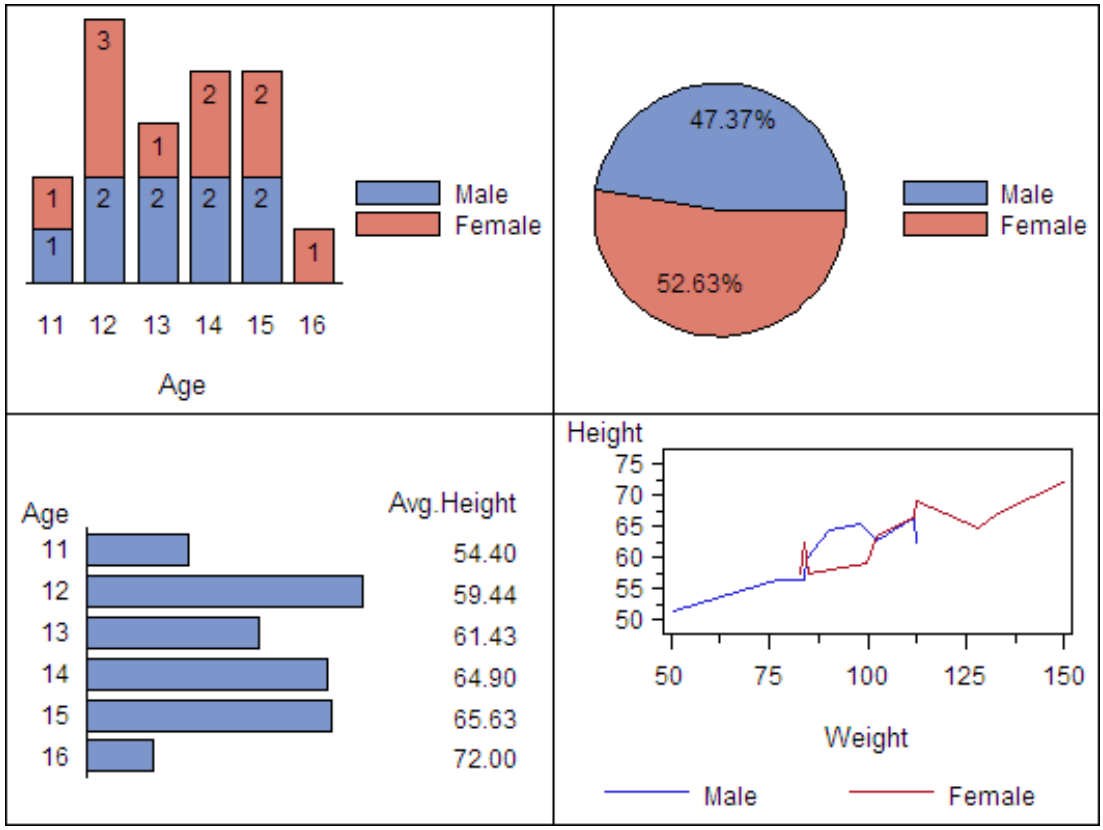
The GREPLAY procedure displays and manages graphics output that is stored in SAS catalogs. The GREPLAY procedure also creates templates and color maps that you can use when you replay your graphics output. The GREPLAY procedure operates in line mode, batch mode, and in the SAS windowing environments.

With the GREPLAY procedure, you can perform any of the following actions:

- Layout multiple graphs on one page; this output can be used to create dashboards.
- 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 50.1 on page 1475 shows four catalog entries that were replayed into a template and displayed as a single graph.

Figure 50.1 Graphics Output in a Template



Concepts

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 may 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 50.1 on page 1476.

Table 50.1 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 with fewer than eight characters, the procedure adds a numeric suffix to the entry's name. The total number of characters is limited to eight.

- For entry names greater than or equal to eight characters, the procedure drops the number of characters needed to add a numeric suffix. The total number of characters is limited to eight. 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 99. △

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 "Running the GREPLAY Procedure Using Code-based Statements" on page 1504.
- 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 1500.

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 the "FS Statement" on page 1487 and the NOFS option.

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 1476 and Example 3 on page 1512.

Procedure Syntax

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.

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.

Restriction: Not supported by Java or ActiveX

Supports: RUN-group processing

PROC GREPLAY <BYLINE>
<CC=*color-map-catalog*>
<CMAP=*color-map-entry*>
<FS>
<GOUT=<*libref.*>*output-catalog*>
<IGOUT=<*libref.*>*input-catalog*>
<IMAGEMAP=*output-data-set*>
<NOBYLINE>
<NOFS>
<PRESENTATION>
<TC=*template-catalog*>
<TEMPLATE=*template-entry*>;

? *required-argument*;

BYLINE;

CC *color-map-catalog*;

CCOPY <*color-map-catalog.*>*color-map-entry*<.CMAP>;

CDEF *color-map-entry*
<*color-definition(s)*>
<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 *required-argument*;

MODIFY *modify-pair(s)*;

MOVE *entry-id-1* AFTER | BEFORE *entry-id-2*;

NOBYLINE;

PREVIEW *template-entry(s)* | _ALL_ ;

QUIT | END | STOP;

REPLAY *entry-id(s)* | _FIRST_ | _LAST_ | _ALL_ ;

TC *template-catalog*;

TCOPY <*template-catalog.*>*template-entry*<.TEMPLATE>;

TDEF *template-entry*
< *panel definition(s)*>
<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*>;

Options

BYLINE
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.

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.

Featured in: Example 4 on page 1514

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.

Featured in: Example 4 on page 1514

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 also: "Catalog Entries" on page 1475

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.

Note: The input catalog can be the same catalog specified in the GOUT= option.

Featured in: Example 2 on page 1510

IMAGEMAP=*output-data-set*
used with the REPLAY statement (see "REPLAY Statement" on page 1493). The IMAGEMAP= option creates a temporary SAS data set that contains information about the graph. The graph is replayed from the graphics catalog. The image map data set contains the following information about the graph:

- the shapes of the elements
- the coordinates of the elements
- the values that are associated with those element; in variables that are identified in the HTML= option
- the values that are associated with those element; in variables that are identified in the HTML_LEGEND= option

The image map data set can be used to generate an HTML image map in an HTML output file using the IMAGEMAP macro. The IMAGEMAP macro takes two arguments: the image map data set name and the name or fileref of an HTML output file, as shown in this example:

```
%imagemap(imgmapds, myimgmap.html);
```

See also: "Adding Links with the HTML= and HTML_LEGEND= Options" on page 601

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

Featured in: Example 1 on page 1508

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.

Featured in: Example 1 on page 1508

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.

Featured in: Example 2 on page 1510

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 50.2 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

? *required–argument(s)*;

CC
CMAP
DEVICE
GOUT
IGOUT
TC
TEMPLATE

Required Arguments

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 also: "NOBYLINE Statement" on page 1491

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 Arguments

color-map catalog
identifies the SAS catalog where color map entries are stored.
Style element:

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.

Requirements: Assign a color map catalog before using the CCOPY statement.

Syntax

CCOPY <*libref.*><*color–map-catalog.*>*color-map-entry.*<.CMAP>;

Required Arguments

***<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.

See also: "CC Statement" on page 1482

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 1476.

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 a color map in the color map catalog.

Requirements: Assign a color map catalog before using the CDEF statement.

Syntax

CDEF *color map entry*
<*color-definition(s)*>
<DES="*description*">;

color-definition has the following form:

color-number / *from-color:to-color*

color-definition has the following form: *color-number* / *from-color:to-color*

Required Arguments

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 also: CC statement

Featured in: Example 4 on page 1514

Options

color-number / from-color:to-color
specifies a color pair and how it is defined.

color-number
specifies the number of a color pair.

from-color:to-color
defines the colors that are being mapped.

from-color
is the color to be mapped.

to-color
is 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: NEW COLOR MAP

CDELETE Statement

Deletes one or more color map entries from the current color map catalog.

Caution: 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 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.

Requirements: Assign a color map catalog before using the CMAP statement.

See also: CC

Featured in: Example 4 on page 1514.

Syntax

CMAP *color-map-entry*;

Required Arguments

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.

Requirements: 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 also: "GOUT Statement" on page 1488 and"IGOUT Statement" on page 1489

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.

Caution: 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.

Requirements: You must specify a device driver that your graphics device can support, and is available to your SAS session.

Syntax

DEVICE *device-name*;

Required Arguments

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.

Requirements: Device must support windows

See also: NOFS on page 1480

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 Arguments

<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 Arguments

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.

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 1476.

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 Arguments

<libref.>input-catalog
identifies the SAS catalog with entries that contain graphics output that you want to replay.

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.

Featured in: Example 4 on page 1514

Syntax

LIST *required-argument*;

required-argument must be one of the following:

CC
CMAP
IGOUT
TC
TEMPLATE

Required Arguments

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 *modify-pair(s)*;

modify-pair or pairs has the following form:

entry-id / *description(s)*

Required Arguments

entry-id* / *description(s)
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. Multiple entry ids can contain both numbers and names.

description(s)

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="*entry-name*"
specifies the entry name. Maximum length for an entry name is eight characters. Duplicate entry names are resolved as described "Duplicate Entry Names" on page 1476.

See also: "About Filename Indexing" on page 99

DES="*description*"
specifies the graph's description. Maximum length for entry description is 256 characters. The description does not appear on the graph.

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 before or after *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 also: "BYLINE Statement" on page 1482

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.

Tip: When previewing templates, press END or ENTER, to move to the next template in the list.

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.

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.

Note: 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.

Alias: PLAY

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.

Tip: Use the TC statement to change the template catalog without exiting the procedure.

Note: SAS supplies several templates in the SASHELP.TEMPLT catalog.

Syntax

TC *template-catalog*;

Required Arguments

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.

Requirements: Assign a template catalog before using the TCOPY statement.
See also: "TC Statement" on page 1493.

Syntax

TCOPY <*template-catalog.*>*template-entry*<.TEMPLATE>;

Required Arguments

<template-catalog.>template-entry<.TEMPLATE>
identifies the template entry to be copied.

template-catalog
is the SAS catalog that contains the template to be copied.

template-entry
is the template entry name.

TEMPLATE
is the catalog entry type. Duplicate entry names are resolved as described in "Duplicate Entry Names" on page 1476.

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.

Requirements: Assign a template catalog before using the TDEF statement.

See also: TC statement

Featured in: Example 1 on page 1508

Syntax

TDEF *template-entry*
<*panel-definition(s)*>
<DES="*description*">;

panel-definition has the following form:

panel-number / <*panel-option(s)*>

panel-option(s) can be one or more of the following:

CLIP
COLOR=*border-color*
COPY=*panel-number*
DEF
DELETE
LLX=*x*
LLY=*y*
LRX=*x*
LRY=*y*
PANEL NUMBER=
ROTATE=*degrees*
SCALEX=*factor*
SCALEY=*factor*
ULX=*x*
ULY=*y*
URX=*x*
URY=*y*
XLATEX=*distance*
XLATEY=*distance*

Required Arguments

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.

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. If you do not specify a border color, then no border is displayed around the panel when you replay graphics output in the panel. A template that contains a panel without a border color is assigned a color when previewed. The GREPLAY procedure creates output with borders.

COPY=*panel-number*
specifies the panel number definition to be copied to this panel.

DEF
specifies a default panel with these coordinates:

Panel Corner	Coordinates
lower left	(0,0)
upper left	(0,100)
upper right	(100,100)
lower right	(100,0)

DELETE
deletes the panel.

Alias: DEL

DES="*description*"
specifies the template entry description. The maximum length for the template entry description is 256 characters. By default, the procedure uses *** new template *** for the description.

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-right 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 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-rignt corner of the panel. Units for *x* are a percentage of the graphics output area.

URY=*y*
specifies the Y coordinate upper-rignt 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 percent.

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

Caution: 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.

Requirements: 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 Arguments

template-entry
identifies an existing template to use when replaying graphics output. Use the TREPLAY statement to replay graphics output in the template.

Featured in: Example 1 on page 1508

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.

Requirements: Before issuing the TREPLAY statement, specify a graphics input catalog with the "IGOUT Statement" on page 1489, assign a template entry catalog with the "TC Statement" on page 1493, and choose a template with the "TEMPLATE Statement" on page 1498.

Alias: TPLAY

Featured in: Example 2 on page 1510

Syntax

TREPLAY *select-pairs*<DES="*description*" NAME="*entry-name*">;

select-pairs follow this form:

template-panel-number1:*entry-id1* <...*template-panel-numberN*:*entry-idN*>

Required Arguments

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.

Options

DES="*entry–description*"
adds a description to the entry in the graphics output catalog. Descriptions are truncated after 256 characters.

NAME="*entry-name*"
names the entry in the graphics output catalog. The name must begin with a letter, and continue with up to seven more letters, numbers, or underscores. Duplicate entry names are resolved as described in "Duplicate Entry Names" on page 1476.

Default: TEMPLATE

See also: "About Filename Indexing" on page 99

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.

Using the GREPLAY Procedure Windows

You can use the GREPLAY windows instead of code-based statements to replay and manage catalog entries. 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:

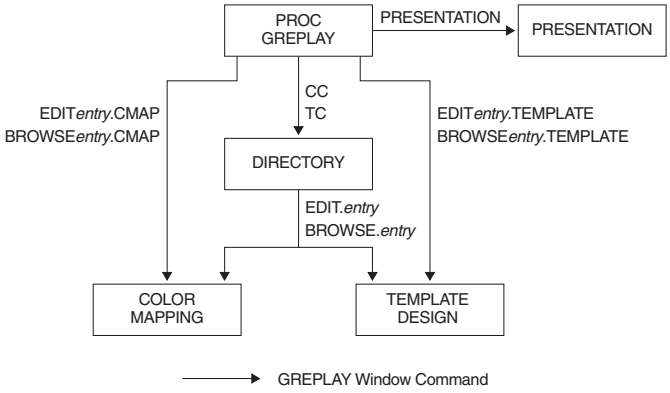
```
proc greplay;
```

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 50.2 on page 1500 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 50.2 GREPLAY Procedure Windows



GREPLAY Window Commands

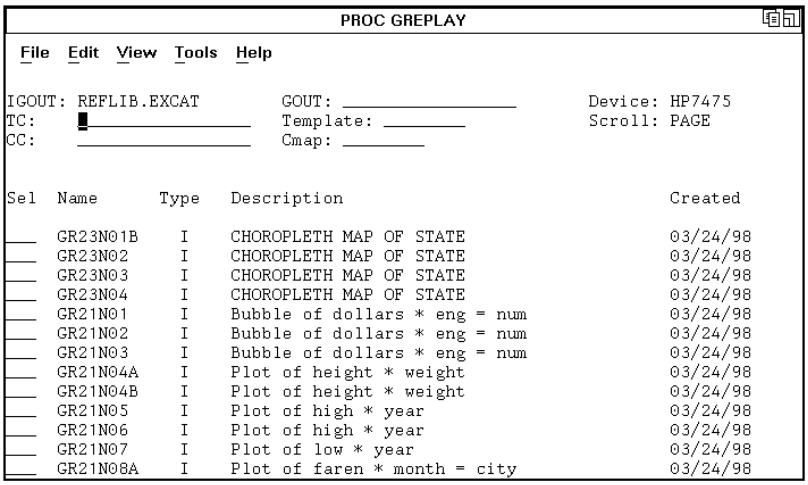
When using GREPLAY windows, tasks are performed by entering values in the fields. Each window can be scrolled forward, or backward to display additional fields

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 on navigating among these windows, see "Commands For Using The GREPLAY Procedure Windows" on page 1503.

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 50.3 PROC GREPLAY Window



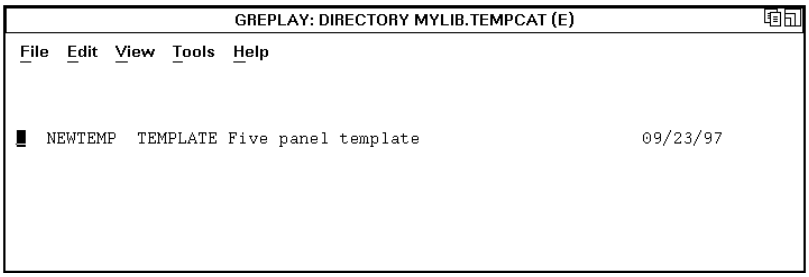
PRESENTATION Window

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 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.

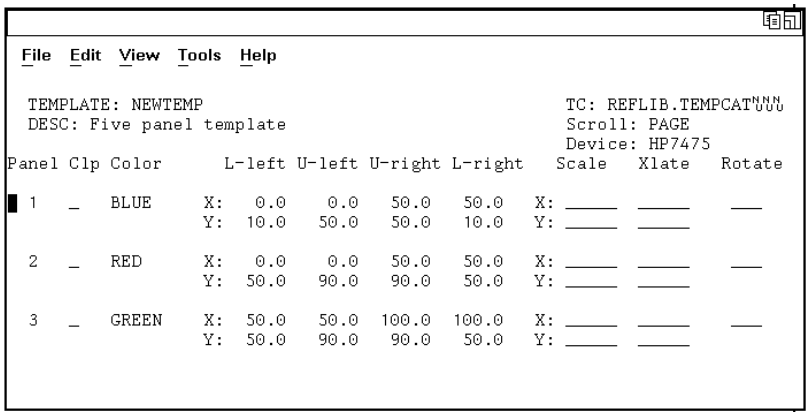
Figure 50.4 Directory Window



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've 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.

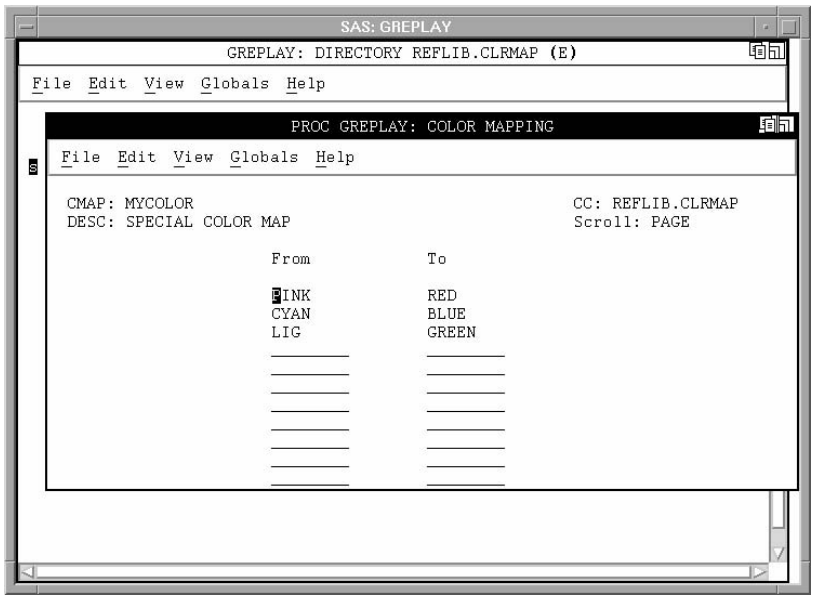
Figure 50.5 Template Design Window



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 50.6 Color Mapping Window



Commands For Using The GREPLAY Procedure Windows

Table 50.3 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 a catalog and issue the PRES command.
	Open DIRECTORY window.	Specify a template catalog and issue the TC command. OR Specify a color map catalog and issue the CC command.
	Open TEMPLATE DESIGN window.	Specify a template catalog and issue the following command: edit *template-name*.template
	Open COLOR MAPPING window.	Specify a color map catalog and issue the following command: edit *color-map-name*.cmap

Location	Task	Command
DIRECTORY window	Open TEMPLATE DESIGN window.	Place an S beside the name of an existing template. OR Issue the following command: edit *template-name*.template
	Open COLOR MAPPING window.	Place an S beside the name of an existing color map. OR Issue the following command: edit *color-map-name*.cmap

Running the GREPLAY Procedure Using Code-based Statements

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 50.4 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

Task	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 50.5 GREPLAY Procedure Statements For Managing GRSEG Catalog Entries

Task	Statement
copy GRSEG entries from an input catalog to an output catalog*	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

* You must assign an output catalog before copying graphics output.

Replaying 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 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. △

Creating 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 use 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;
```

Replaying 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 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”.

Creating 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

The following examples illustrate major GREPLAY procedure features.

Note: 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 HTML;** ***procedure-code;*** **ODS HTML CLOSE;**). See "RUN-Group Processing" on page 56 for more information. △

Example 1: Creating a Template

Procedure features:

GREPLAY statement options:

NOFS

TC= option

TDEF statement

TEMPLATE statement

Sample library member: GRECRTM1

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 50.1 shows the template definition written to the SAS log file. The template defined here is also used in Example 2 on page 1510.

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"
Panel 1: Lower Left Quadrant
```

```
     1/llx=0   lly=10
       ulx=0   uly=50
       urx=50  ury=50
       lrx=50  lry=10
       color=navy
Panel 2: Upper Left Quadrant
     2/llx=0   lly=50
       ulx=0   uly=90
       urx=50  ury=90
       lrx=50  lry=50
       color=lime
Panel 3: Upper Right Quadrant
     3/llx=50 lly=50
       ulx=50 uly=90
       urx=100 ury=90
       lrx=100 lry=50
       color=yellow
Panel 4: Lower Right Quadrant
     4/llx=50 lly=10
       ulx=50 uly=50
       urx=100 ury=50
       lrx=100 lry=10
       color=cyan
Panel 5: Container for Title and Panels 1--4
     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;
```

Output 50.1 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;
.
.
.
```

Example 2: Replaying GSLIDE Procedure Output in a Template

Procedure features:

GREPLAY statement options:

- GOUT= option
- IGOUT= option
- TEMPLATE= option

TREPLAY statement

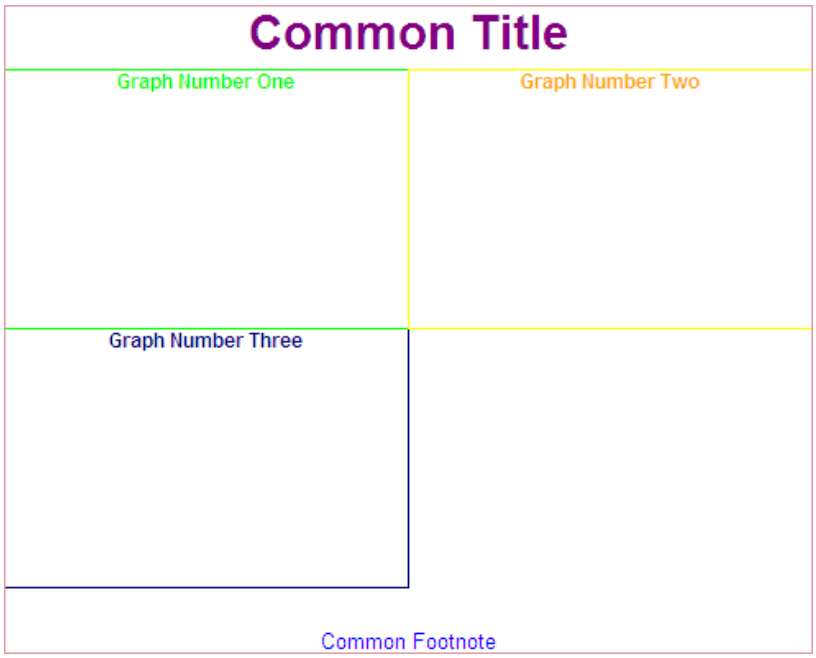
Other features:

PROC GSLIDE

Sample library member: GRERGOT1

The TREPLAY statement replays into the template GRFTMPLT, the four catalog entries that contain graphics output. It 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.

Figure 50.7 Replayed Output in a Graphics Template (grergot1)



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=4.13in;
```

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

Procedure Features:

GREPLAY statement options:

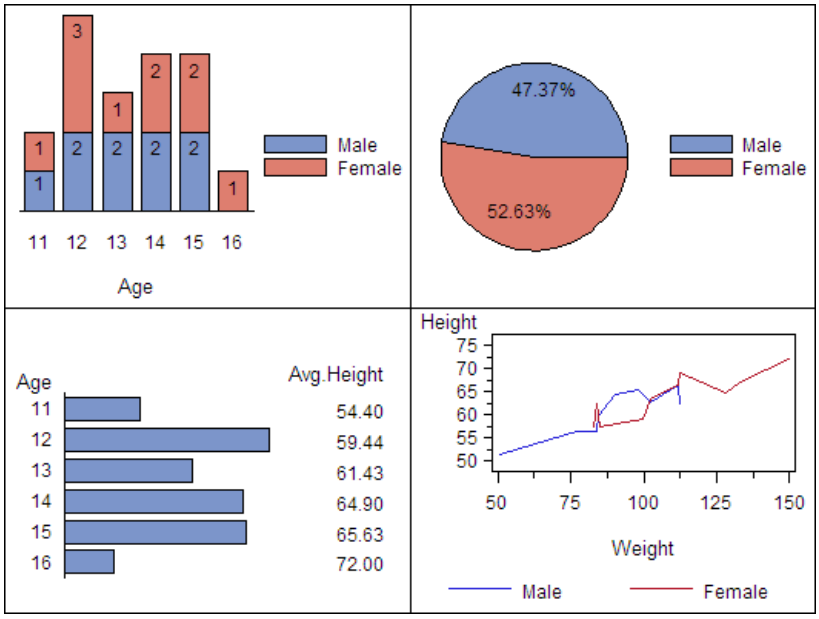
GOUT=
IGOUT=
NOFS
TC=
TEMPLATE=

DEVICE statement

TREPLAY statement

Sample library member: GREGRSEG

Display 50.1 SAS/GRAPH Graphs Replayed into a Template



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 sasuser.class (drop=name );
   length Gender $ 6;
   set sashelp.class;
   if sex="F" then Gender="Female";
      else Gender="Male";
run;
proc sort data=sasuser.class out=sasuser.class;
   by weight height;
run;
```

Define the size of each individual graph. Each graph is replayed into a separate panel in the template.

```
  goptions reset=all hsize=2.75in vsize=2.06in;
```

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=(right middle outside) across=1;
legend2 label=none value=("Male" "Female");
```

Create a symbol definition for the plot.

```
symbol i=join;
```

Generate the graphs, and store them in the SASUSER.EXCAT catalog. Generate a vertical bar chart, a horizontal bar chart, a pie chart, and a subgrouped plot.

```
proc gchart data=sasuser.class gout=sasuser.excat;
   vbar age/discrete hminor=0 subgroup=gender
                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=sasuser.class gout=sasuser.excat;
    plot height*weight=gender/ vminor=1 vaxis=axis3
    haxis=axis4 legend=legend2;
run;
quit;
```

Define the size of each 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=5.5in vsize=4.12in;
```

Replay the graphs with a template to create one graph. The graphs stored in SASUSER.EXCAT are replayed to create one graph. The graph is also stored in the SASUSER.EXCAT catalog.

```
proc greplay gout=sasuser.excat igout=sasuser.excat nofs
   tc=sashelp.templt template=l2r2;
   device win;
   treplay 1:gchart 2:gchart1 3:gchart2 4:gplot;
quit;
```

Example 4: Creating a Color Map

Procedure features:

GREPLAY statement options:

CC= option
GOUT= option

CDEF statement

CMAP statement

LIST statement

Sample library member: GRECRCM1

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 50.2 shows a partial SAS log listing.

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. Both options must be assigned for the color map to be created.

```
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;
```

Output 50.2 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;
.
```

CHAPTER

51

The GSLIDE Procedure

Overview

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 51.1 on page 1517 shows a slide containing text that was produced with TITLE, FOOTNOTE, and NOTE statements.

Figure 51.1 Text Slide Produced by the GSLIDE Procedure (GSLTEXTS)

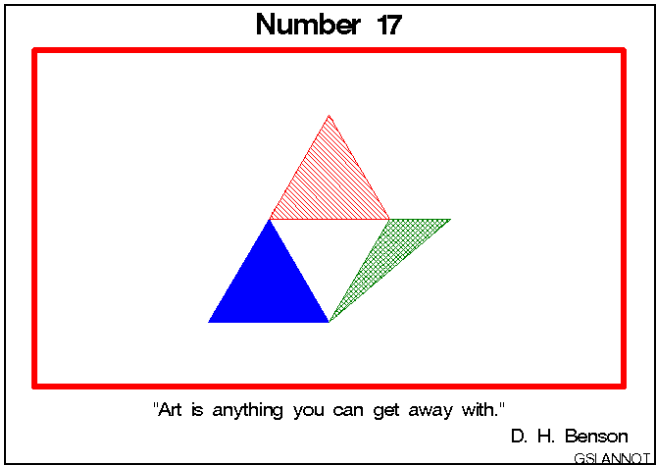
The program for this slide is in Example 1 on page 1522.

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 29, "Using Annotate Data Sets," on page 641 for more information on creating and displaying Annotate data sets.

Figure 51.2 on page 1518 shows output from an Annotate data set that is displayed with titles and footnotes that were generated by TITLE and FOOTNOTE statements.

Figure 51.2 Output from an Annotate Data Set Displayed with the GSLIDE Procedure (GSLANNOT)



The program for this slide is in Example 2 on page 1524.

Procedure Syntax

Requirements: At least one of these is required: a TITLE, FOOTNOTE, or NOTE statement; an appearance option; the BORDER graphics option.

Global statements: FOOTNOTE, TITLE

Reminder: The procedure can include the SAS/GRAPH NOTE statement.

PROC GSLIDE <*option(s)*>;

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 <*option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options:
  - ANNOTATE=*Annotate-data-set*
  - BORDER
  - CFRAME=*frame-color*
  - FRAME
  - IFRAME= *fileref* | *'external-file'*
  - IMAGESTYLE = TILE | FIT
  - LFRAME=*line-type*
  - WFRAME=*n*
- description options:
  - DESCRIPTION='*entry-description*'
  - GOUT=<*libref*.>*output-catalog*
  - NAME='*entry-name*'
- HTML option:
  - <IMAGEMAP=*output-data-set*>

Options

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.

See also: Chapter 29, "Using Annotate Data Sets," on page 641

Alias: ANNO=*Annotate-data-set*

Featured in: Example 2 on page 1524

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 also: “Adding Frames, Borders, and Images” on page 1521

Featured in:

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 also: “Adding Frames, Borders, and Images” on page 1521.

Featured in: Example 1 on page 1522.

DESCRIPTION=’*entry-description*’
specifies the description of the catalog entry for the slide . The maximum length for *entry-description* is 256 characters. By default, the GSLIDE procedure assigns the description GRAPHICS TEXT SLIDE.

The descriptive text is shown in each of the following:

- the “description” portion of the Results window
- the catalog-entry properties that you can view from the Explorer window
- the Description field of the PROC GREPLAY window

Alias: DES=’*entry-description*’

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. The FRAME option is overridden by the IFRAME= option, which fills the backplane frame with an image.

See also: “Adding Frames, Borders, and Images” on page 1521

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 also: “Specifying the Catalog Name and Entry Name for Your GRSEGs” on page 100

IFRAME=*fileref* | ’*external-file*’
identifies the image file you want to apply to the backplane of the plot. See also the IMAGESTYLE= option. The IFRAME= option is overridden by the NOIMAGEPRINT goption.

IMAGEMAP=*output-data-set*
creates a temporary SAS data set that is used to generate an image map in an HTML output file. The information in the image map data set includes the shape and coordinates of the elements in the graph and drill-down URLs that have been associated with those elements. The drill-down URLs are provided by one or two variables in the input data set. These variables are identified to the GSLIDE procedure with the HTML= or HTML_LEGEND= options or both.

The %IMAGEMAP macro generates the image map in the HTML output file. The macro takes two arguments, the name of the image map data set and the name or fileref of the HTML output file, as shown in the following example:

```
%imagemap(imgmapds, myimgmap.html);
```

See also: “Adding Links with the HTML= and HTML_LEGEND= Options” on page 601 and “HTML Variable” on page 709

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 14.22 on page 277. By default, the line type is specified by the current style. LFRAME=1, which produces a solid line, is the default.

NAME='*entry-name*'
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created. The name can be up to 256 characters long, but the GRSEG name is truncated to eight characters. Uppercase characters are converted to lowercase, and periods are converted to underscores. The default GRSEG name is GSLIDE. If the name duplicates an existing name, then SAS/GRAPH adds a number to the name to create a unique name—for example, GSLIDE1.

See also: "About Filename Indexing" on page 99

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 also: "Adding Frames, Borders, and Images" on page 1521

Featured in: Example 1 on page 1522

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.

While 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 59 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, "The GANNO Procedure," on page 913 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 "RUN-Group Processing" on page 56 for details.

Examples

Note: 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 HTML; procedure-code; ODS HTML CLOSE;`**). See "RUN-Group Processing" on page 56 for more information. △

Example 1: Producing Text Slides

Procedure features:

PROC GSLIDE options:

BORDER
CFRAME=
WFRAME=

Other features: NOTE Statement

Sample library member: GSLTEXTS

This example uses FOOTNOTE, NOTE, and TITLE statements to produce a text slide. PROC GSLIDE statement options add both a border and a frame.

Set the graphics environment.

```
goptions reset=all cback=blue
           border;
```

Define titles and footnotes.

```
title color=red "New Directions";
footnote1 j=l " ABC Engineering, Inc";
footnote2 j=l " January 1998" ;
```

Generate the slide and define additional text. The BORDER option draws a box around the entire graphics output area. The CFRAME= option draws a 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=red
            wframe=4;
   note height=5;
   note height=3
        justify=center
                       color="white"
                      "Goals and strategies"
        justify=center
                      "for the coming year";
run;
quit;
```

Example 2: Displaying Annotate Graphics

Procedure features:

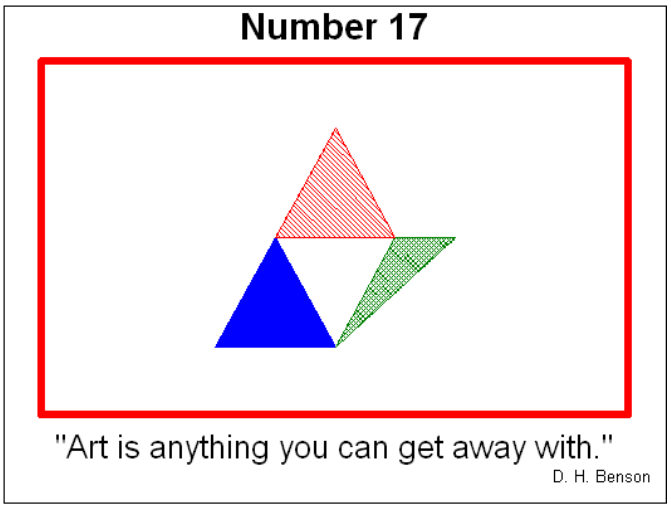
PROC GSLIDE option:

ANNOTATE=

Other features:

Annotate data set

Sample library member: GSLANNOT



In this example, the GSLIDE procedure displays Annotate graphics along with current TITLE and FOOTNOTE definitions. See Chapter 29, "Using Annotate Data Sets," on page 641 for information on creating Annotate data sets.

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

52

The GTILE Procedure

Overview

The GTILE procedure creates charts that consist of a rectangle or a square divided into tile-shaped segments. These charts represent the relative sizes of tiles to one another and to the whole. The GTILE procedure provides three statements you can use to define the layout in order to visualize your data. The statements require one numeric variable. This variable defines the top level of the chart.

The TILEBY= statement is followed by any number of numeric or character variables that are delineated by a comma or a blank space. By providing multiple TILEBY variables, and specifying either a JAVA or 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

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 sizevariable must be numeric. All the values are treated as discrete. The sum of the chart's size variable variable 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. △

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 non-missing, 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, with its value 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. △

Assigning Colors

The COLORRAMP= option enables you to customize the chart's colors. You can specify colors for the tiles using any of the color-naming schemes supported by SAS/GRAPH.

Table 52.1 Examples of Specifying Colors

Color-Naming Scheme	Example
RGB	`COLORS=(cx98FB98 cxDDA0DD cxFFDAB9 cxDB7093 cxB0E0E6)`
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")`

Color Naming Schemes

For information about color naming schemes, see Chapter 12, "SAS/GRAPH Colors and Images," on page 167.

Procedure Syntax

Requirements:

- a GOPTIONS statement with a JAVA, JAVAIMG, ACTIVEX or ACTXIMG device specified
- an ODS statement to close the listing destination
- an ODS statement to open the output destination
- at least one FLOW, TILE or TOGGLE statement
- at least one TILEBY= variable
- an ODS statement to close the output destination
- an ODS statement to open the listing destination

Global Graph Statements: FOOTNOTE, GOPTIONS , TITLE

Global Procedure Statements: FORMAT, LABEL, WHERE, ODS

Supports: Activex devices and Java devices, RUN-Group Processing

PROC GTILE<DATA=*input-data-set*>;

FLOW | TILE | TOGGLE *size-variable* **TILEBY=***(variable) | (variable-list)* </ *option(s)*>;

PROC GTILE Statement

Identifies the data set containing the chart variables.

Requirements: An input data set is required.

Syntax

PROC GTILE<DATA=*input-data-set*>;

Options

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: "SAS Data Sets" on page 54

FLOW, TILE, and TOGGLE Statements

Create a tile chart in one of the three display layouts.

Requirements: At least one numeric chart sizevariable and one TILEBY= variable is required.

Global Graph Statements: FOOTNOTE, GOPTIONS, TITLE

Supports: Drill-down functionality

Description

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 left-bottom 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.

Syntax

FLOW | TILE | TOGGLE *size-variable* **TILEBY=***(levels-variable) | (levels-list)*
variable list delimiter can be either a blank space or a comma <*/option(s)*>;

option(s) can be one or more of the options from any or all of the following categories:

- appearance options:
  - CMISSING=*missing-value-color*
  - COLORRAMP=*color-ramp-color-list*
  - COLORVAR=*data-set-variable*
  - DETAILLEVEL=*number-of-levels-to-show-in-detail*
  - LABELLEVEL=*number-of-levels-to-label*
- midpoint option:
  - BASELINE=*midpoint-value*
- description options:
  - DESCRIPTION="*description*"
  - NAME="*name*"

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. If the COLORVAR= option is not specified, then the sizevariable is used to assign a color to each tile.

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. Variable must be enclosed in parenthesis and you can use either commas or blank spaces as delimiters.

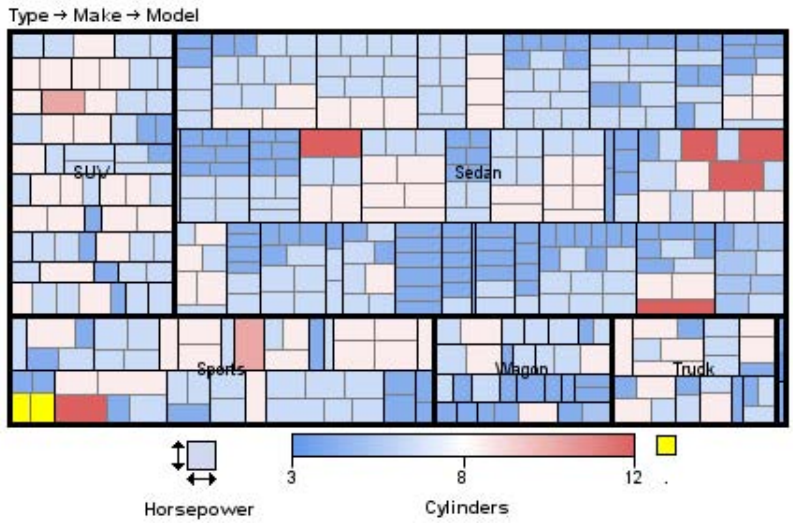
Options

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.

CMISSING=*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 52.1 CMISSING=yellow



Alias: CMISS=

Restriction: Partially supported by Activex.

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. 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 52.2 COLORRAMP=(cxbcd3aa cxae9aeb)

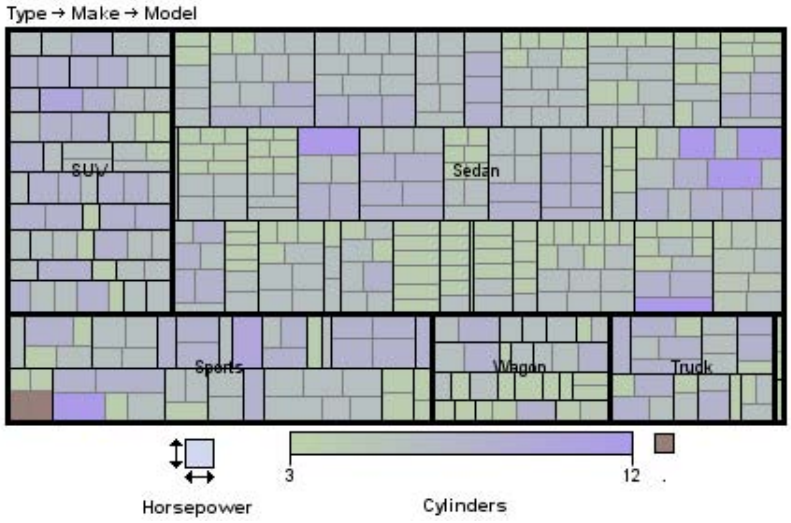
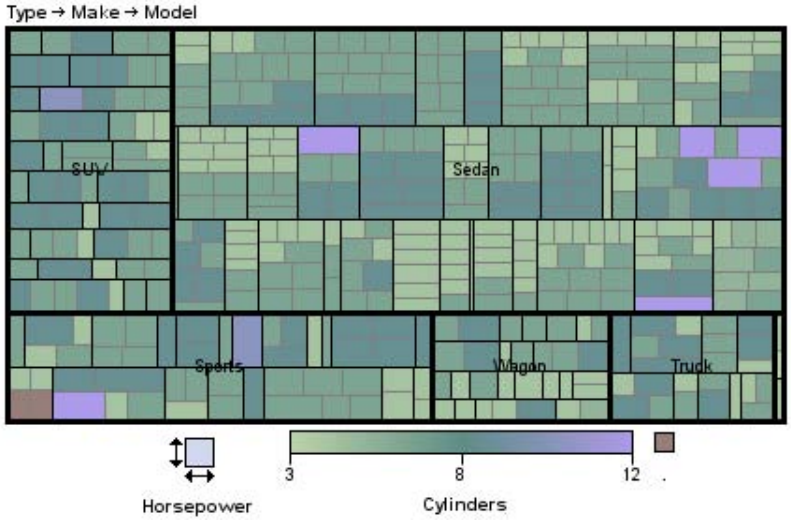


Figure 52.3 COLORRAMP=(cxbcd3aa cx5f8e97 cxae9aeb)

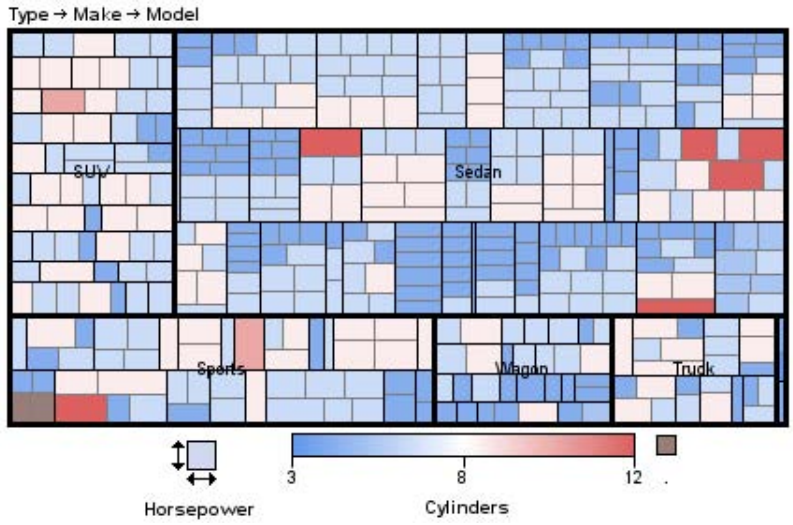


Alias: RAMP=

Style reference: Color attribute of the ThreeColorRamp element

COLORVAR=*color-variable*
specifies a numeric variable whose values determine the color of the tiles. The smallest value of this variable is assigned to the first color in the color ramp. The largest value of this variable is assigned the last color in the color ramp. Each value is assigned a color from the gradient list of colors between the first, and the last colors in the color ramp.

Figure 52.4 COLORVAR=cylinders



DESCRIPTION="*description*"
specifies the description of the chart. The maximum length for description is 256 characters.

The descriptive text is displayed as follows:

- the description in the Results window
- the description in the Table of Contents that is generated when you use the CONTENTS= ODS option
- the chart description in the HTML file when the output destination is ODS HTML and DEVICE=ACTXIMG or DEVICE=JAVAIMG

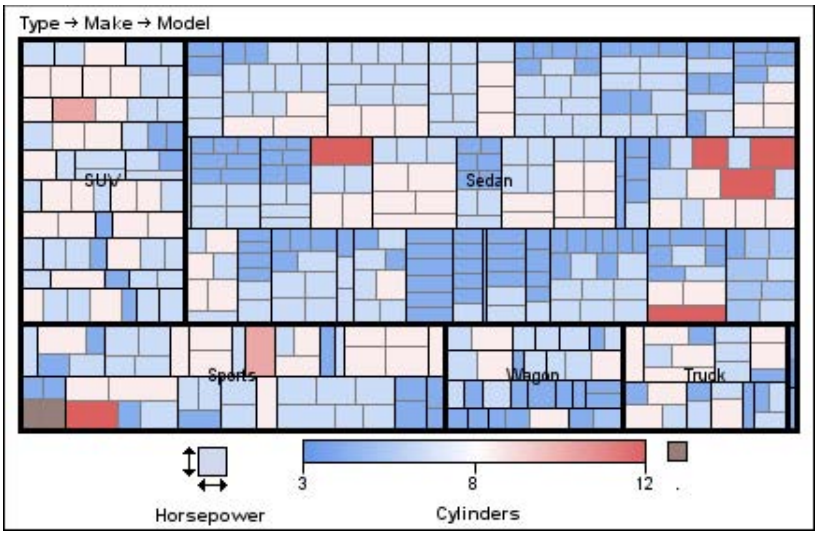
Alias: DES=

Default: Tile chart of *tileby*

Restriction: Partially supported by Activex and Java.

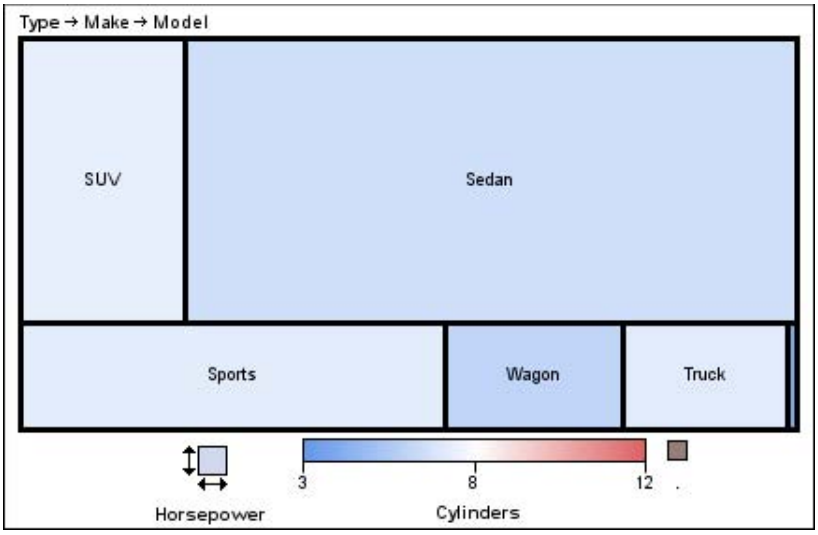
DETAILLEVEL=*1-to-the-number-of-variables-specified-by-TILEBY*=
specifies the number of levels to display. The valid values for the DETAILLEVEL= option are from one to the number of variables listed in the TILEBY= levels list. Each level has a unique outline. As you drill down through the levels, the 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 52.5 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 52.6 DETAILLEVEL=1



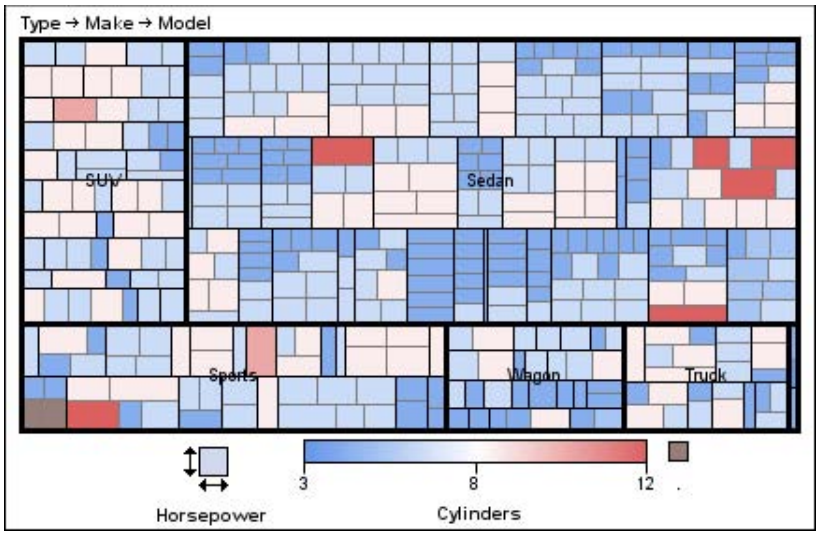
Alias: DLEVEL=

Default: 3

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.

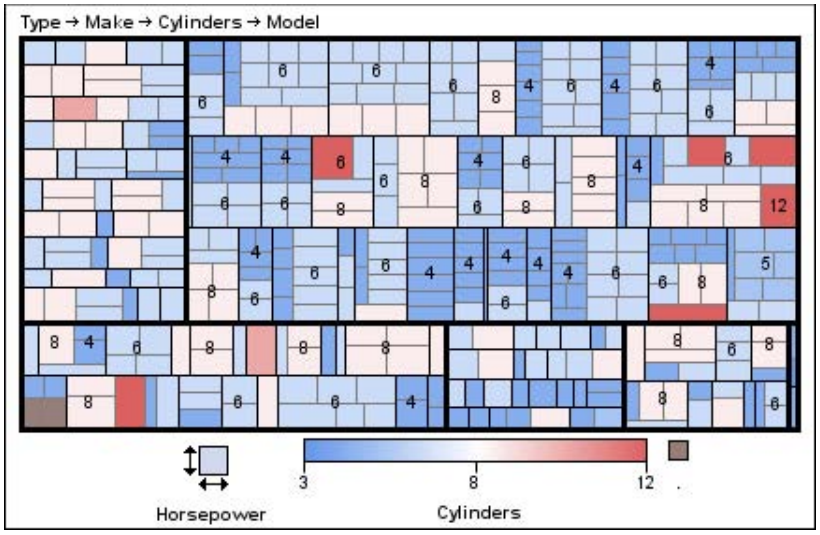
If the LABELLEVEL= option is omitted, level 1 labels are initially displayed.

Figure 52.7 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 52.8 LABELLEVEL=3



Alias: LLEVEL=

Default: 1

NAME=*"entry"*

specifies the name of any graphics output file created. The name can be up to 256 characters long. If the name duplicates an existing filename, SAS/GRAPH adds a number, or increments the last number used to create a unique graph name—for example, graph1.png.

Default: graph.png

See also: "About Filename Indexing" on page 99

Examples

Example 1: Simple GTILE with the COLORVAR= Option

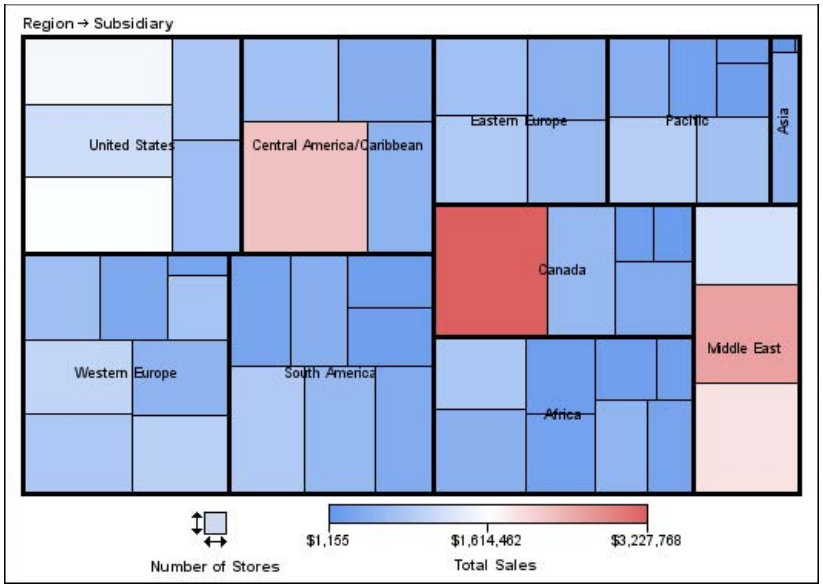
Procedure features:
PROC GTILE statement
TILE statement
TILEBY=*(levels-list)*
COLORVAR= option

Data Set: SASHELP.SHOES

Sample Library Member: GTLSIMPL

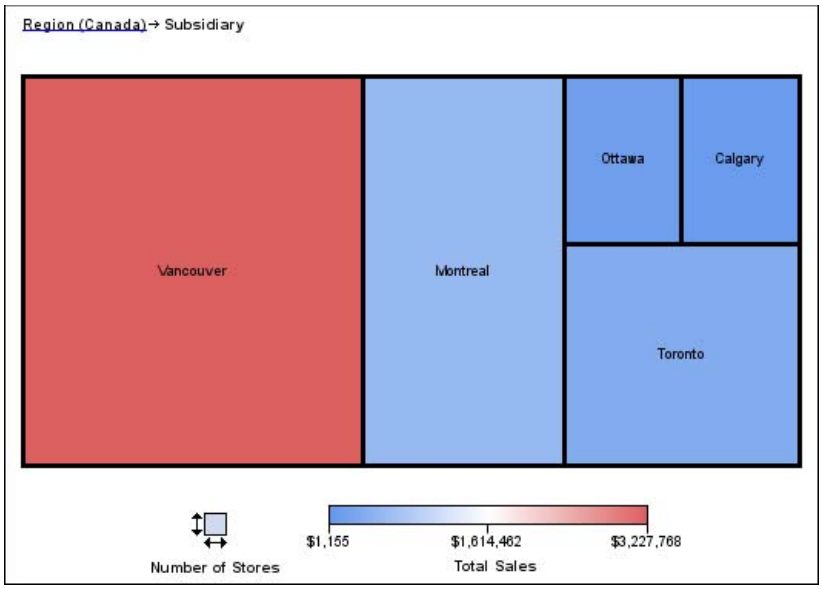
PROC GTILE generates a chart for the SASHELP.SHOES data set. The size of each tile represents the number of stores. The COLORVAR=SALES option specifies that the color of each tile represents the sales revenue for that tile. The visualization of the data with the GTILE procedure makes it easy to see the data extremes for sales revenue relative to the number of stores.

Display 52.1 GTILE Chart of SASHELP.SHOES (gtlsimpl)



The following chart shows the result of drilling down on the region labeled “Canada”. The region was selected to further explore the data. This region displayed the largest area of red, indicating a greater amount of shoe sales.

Display 52.2 Subset of SASHELP.SHOES where Region="Canada" (gtlsimpl)



Close the LISTING destination.

```
ods listing close;
```

Open the HTML destination.

```
ods html file="shoe_sales.html";
```

Set the graphics options.

```
goptions reset=all device=java;
```

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 the HTML output destination.

```
ods html close;
```

Open the listing destination.

```
ods listing;
```

Example 2: Specifying the COLORRAMP= Option, and Setting the DETAILLEVEL= Option

Procedure features:

PROC GTILE statement

FLOW statement

COLORVAR=

COLORRAMP=

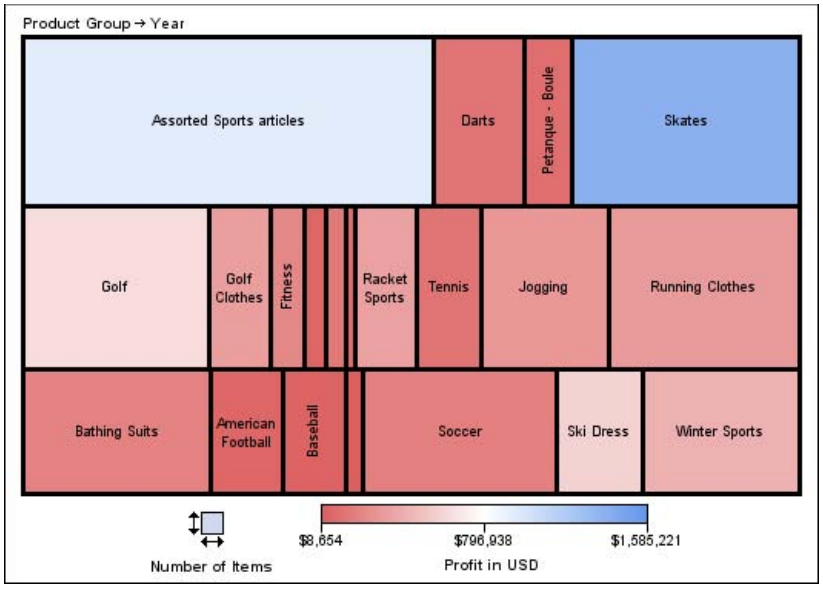
DETAILLEVEL=

Data Set: SASHELP.ORSALES Subset

Sample Library Member: GTLCOLOR

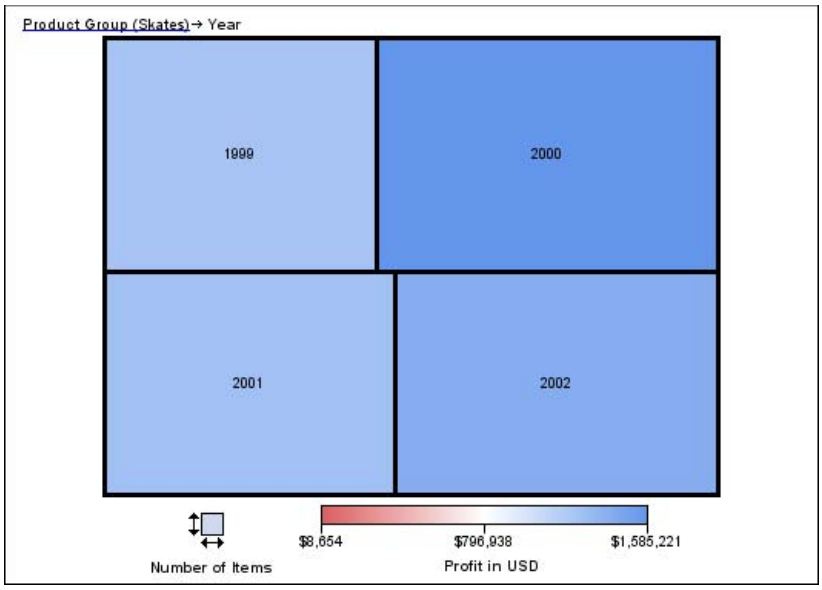
PROC GTILE generates this chart displaying a subset of the SASHELP.ORSALES data set. The FLOW statement defines the layout of the data. The size of each tile indicates the number of items sold. The color of each tile indicates the profit. The visualization of the data with the GTILE procedure makes it easy to see the data extremes for profit relative to the number of items sold.

Display 52.3 Tile Chart of a Subset of SASHELP.ORSALES (gtlcolor)



Clicking on the "Skates" product_group subsets the "Skates" observations by year. The updated display provides information on the "Skates" profits by year.

Display 52.4 Subset of SASHELP.ORSALES where Product_Group="Skates" (gtlcolor)



Close the listing destination.

```
ods listing close;
```

Open the HTML output destination.

```
ods html file="sport_sales.html";
```

Set the graphics options.

```
goptions reset=all device=java;
```

Subset the data, format the quantity variable, and the profit variable.

```
data sports_only;
  set sashelp.orsales;
  if product_line="Sports";
  format profit dollar12.;
  format quantity comma12.;
run;
```

The chart variable QUANTITY specifies the size of the tiles.

The FLOW layout arranges the tiles.

The TILEBY=*(levels-list)* variable list defines the tile segments and the chart levels.

The COLORVAR=PROFIT option specifies the variable to use to color the tiles.

The DETAILLEVEL=1 option defines the level of display detail.

The COLORRAMP= option reverses the colors. Blue represents the highest value. Red represents the lowest value.

```
proc gtile data=sports_only;
    flow quantity tileby=(product_group year) /
        colorvar=profit
        /* display less details */
        detaillevel=1
        /* reverse the colors so that blue is highest */
        colorramp=(CXDD6060 CXFFFFFF CX6497EB);
run;
quit;
```

Close the HTML output destination.

```
ods html close;
```

Open the listing destination.

```
ods listing;
```

CHAPTER

53

The G3D Procedure

Overview

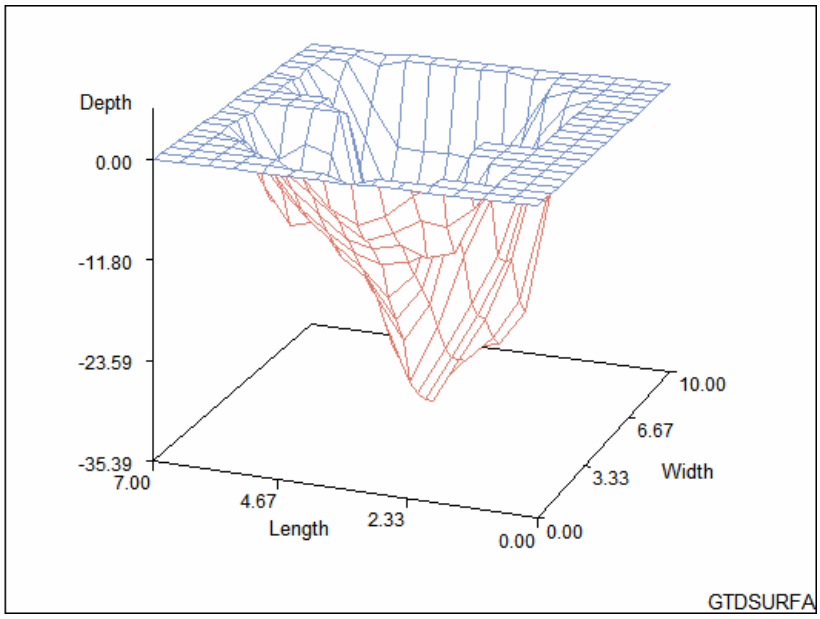
The G3D procedure enables you to produce three-dimensional plots. The surface plot in the following section displays various depths of a lake. The dimensions of the lake are plotted on the X-Y axes. The Z variable is plotted as the third dimension. The coordinates of each point correspond to the values of the three numeric variable values in an observation from the selected input data set.

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.

Figure 53.1 G3D Surface Plot



With the PLOT statement, you can do the following actions:

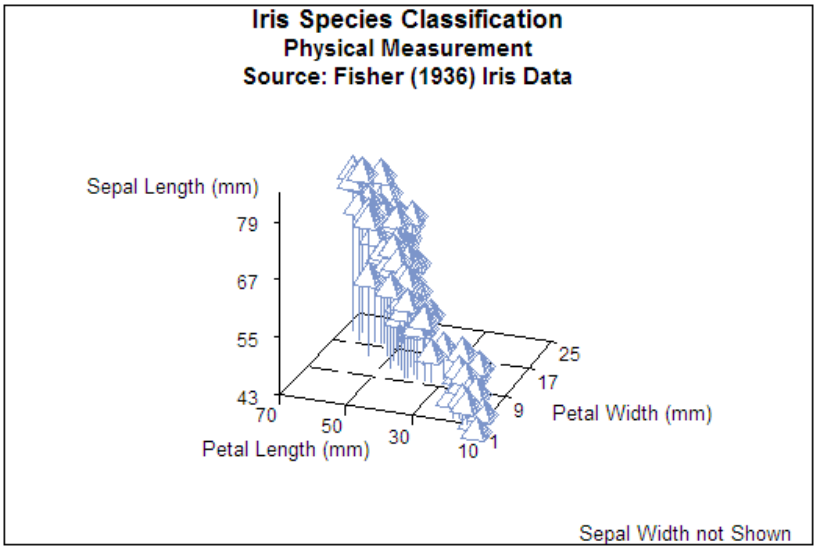
- show the three-dimensional shape of your data (useful for examining data trends).
- change the data ranges that are displayed.
- rotate and tilt the plot to enhance viewing angles.
- customize the axes.

Featured in Example 1 on page 1560. For more information on producing surface plots, see the "PLOT Statement" on page 1547.

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 53.2 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

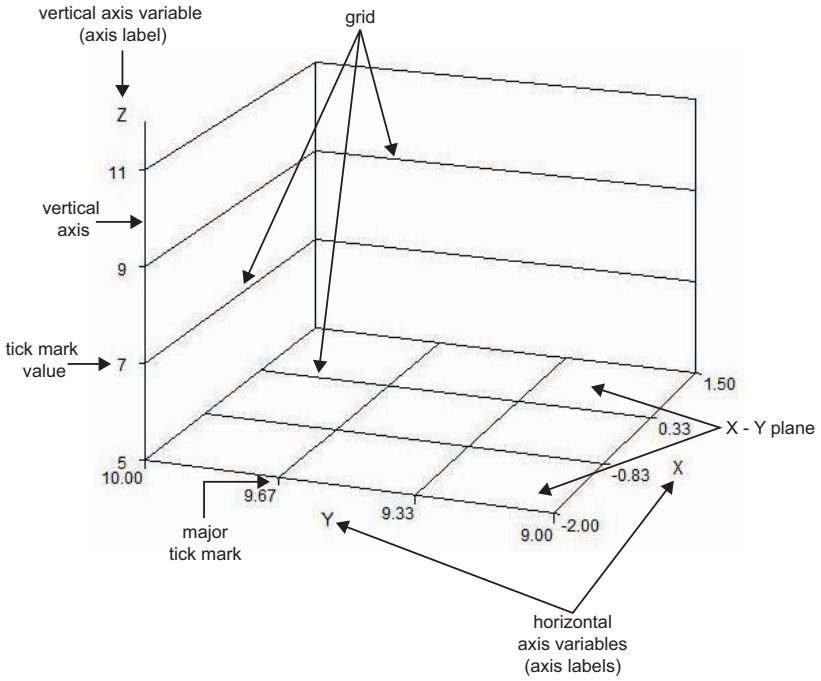
For more information on producing scatter plots, see the "SCATTER Statement" on page 1554.

Concepts

G3D Procedure Terms

The following illustration provides the terminology used to describe the elements of the three-dimensional plots generated with G3D.

Figure 53.3 Elements of a Three-Dimensional Plot



The Input 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 non-missing Z values for at least 50 percent 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 non-missing 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 54, "The G3GRID Procedure," on page 1571 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.

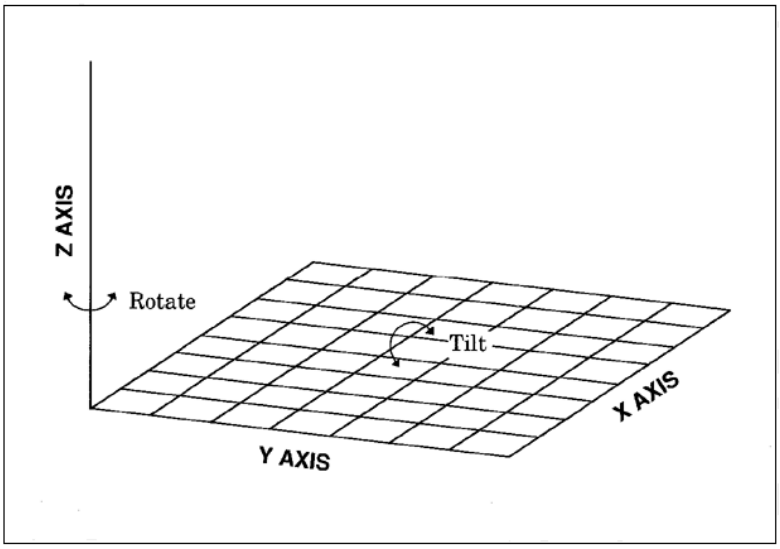
Note: See the "SCATTER Statement" on page 1554 for information on controlling axis labels and tick mark values with SCATTER statement options. △

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 53.4 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:

- 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.

Procedure Syntax

Requirements: At least one PLOT or SCATTER statement is required.

Global statements: AXIS, BY, FOOTNOTE, GOPTIONS, NOTE, TITLE

Reminder: The procedure can include the FORMAT, LABEL, and WHERE statements.

Restriction: The AXIS statement is partially supported by the Java and ActiveX devices only.

PROC G3D <DATA=*input-data-set*>
<ANNOTATE=*annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>;
PLOT *plot-request*</*option(s)*>;
SCATTER *plot-request*</*option(s)*>;

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*>;

Options

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 also: Chapter 29, "Using Annotate Data Sets," on page 641

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 also: "SAS Data Sets" on page 54 and "The Input Data Set" on page 1544

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 doesn't already exist.

See also: Chapter 7, "SAS/GRAPH Output," on page 87

PLOT Statement

Creates three-dimensional surface plots using values of three numeric variables from the input data set.

Requirements: One plot request is required.

Global statements: AXIS, BY, FOOTNOTE, GOPTIONS, NOTE, TITLE

Reminder: The procedure can include the FORMAT, LABEL, and WHERE statements.

Restriction: The AXIS statement is partially supported by the Java and ActiveX devices only.

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.

Syntax

PLOT *y*x=z </option(s)>*;

Option(s) can be one or more options from any or all of the following categories:

- appearance options:

  ANNOTATE=*annotate-data-set*

  CBOTTOM=*bottom-surface-color*

  CTOP=*top-surface-color*

ROTATE=*angle-list*
SIDE
TILT=*angle-list*
XYTYPE=0 |1 | 2 | 3

- axes options:
  CAXIS=*axis-color*
  CTEXT=*text-color*
  GRID
  NOAXIS | NOAXES
  NOLABEL
  XAXIS=*axis <1...99>*
  XTICKNUM=*number-of-major-tick-marks*
  YAXIS=*axis <1...99>*
  YTICKNUM=*number-of-major-tick-marks*
  ZAXIS=*axis <1...99>*
  ZMAX=*maximum-value*
  ZMIN=*minimum-value*
  ZTICKNUM=*number-of-major-tick-marks*
- catalog entry description options:
  DESCRIPTION=*"entry-description"*
  NAME=*"name"*

Required Arguments

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

Options

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 also: Chapter 29, "Using Annotate Data Sets," on page 641

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 it finds from the following list:

1. colors specified for labels and values on assigned AXIS statements, which override the CTEXT= option on the PLOT statement. (Colors specified on AXIS statements are supported by the Java and ActiveX devices only.)
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, 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

Note: 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. △

Note: For Java and ActiveX, specific text options specified in the AXIS statement override the CTEXT= option. △

Style reference: Color attribute of the GraphValueText and the GraphLabelText elements

CTOP=*top-surface-color*
specifies a color for the top of the plot surface.

Style reference: Color attribute of the GraphData1 element

Featured in: Example 2 on page 1561

DESCRIPTION=*"description"*
specifies the description of the plot. The maximum length for *description* is 256 characters.

The descriptive text is displayed as follows:

- the description in the Results window
- the description in the Explorer view of the catalog entry
- 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 also: "Substituting BY Line Values in a Text String" on page 294

Restriction: Partially supported for ActiveX and Java

GRID
draws reference lines at the major tick marks on all axes.

Featured in: Example 2 on page 1561.

Restriction: Not supported by Java

NAME="*name*"
specifies the name of the GRSEG catalog entry, and the name of any graphics output file created. The name can be up to 256 characters long. If the name duplicates an existing name, SAS/GRAPH adds a number, or increments the last number used to create a unique graph name–for example G3D1.

For GRSEG entries:

- the name is truncated to eight characters
- the first character is always represented in upper case
- all other characters are represented in lower case
- periods and blanks are converted to underscores

For Graphics Output files:

- SAS/GRAPH adds a number to the NAME= value, or increments the last number used.

Default: Procedure name

See also: "About Filename Indexing" on page 99

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 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

Featured in: Example 2 on page 1561

Restriction: Not supported by ActiveX

SIDE
produces a surface graph with a side wall.

Featured in: Example 3 on page 1563

Restriction: Partially support by Java

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 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

Featured in: Example 3 on page 1563

XAXIS= *AXIS<1...<99>*
assigns an axis definition.

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:

1. XYTYPE=0 (Java and ActiveX only) No lines are displayed. The plot is displayed as a solid surface.
2. XYTYPE=1 draws lines that are parallel to the X axis. The surface is displayed by using lines that represent Y axis values.
3. XYTYPE=2 draws lines that are parallel to the Y axis. The surface is displayed by using lines that represent X axis values.
4. 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.

Featured in: "Changing the Surface Appearance" on page 1552

Restriction: Not supported by Java

YAXIS=*AXIS <1...<99>*
assigns an axis definition.

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<1...<99>*
assigns an axis definition.

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

Featured in: Example 2 on page 1561

Restriction: Not supported by Java

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

Featured in: Example 2 on page 1561

Restriction: Not supported by Java

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

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 53.5 Surface Appearance for XYTYPE=1

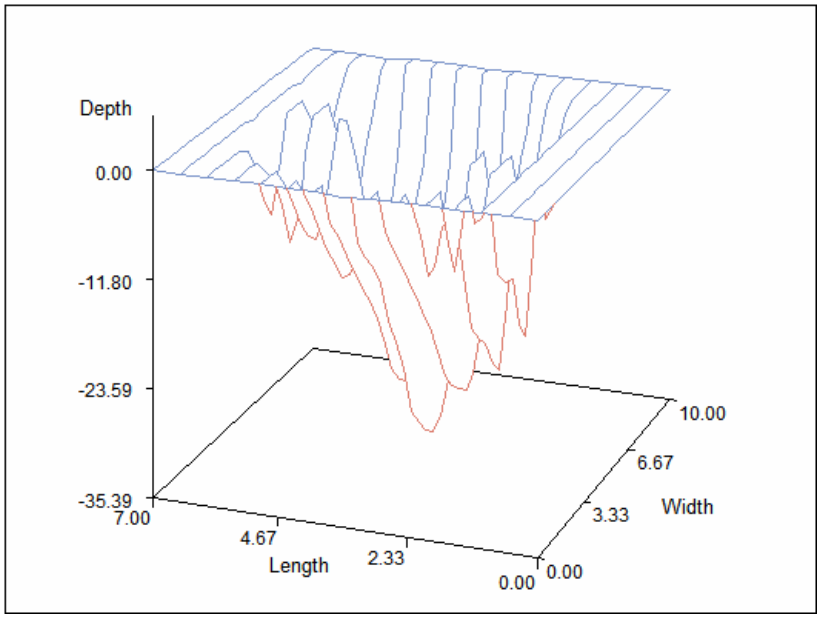


Figure 53.6 Surface Appearance for XYTYPE=2

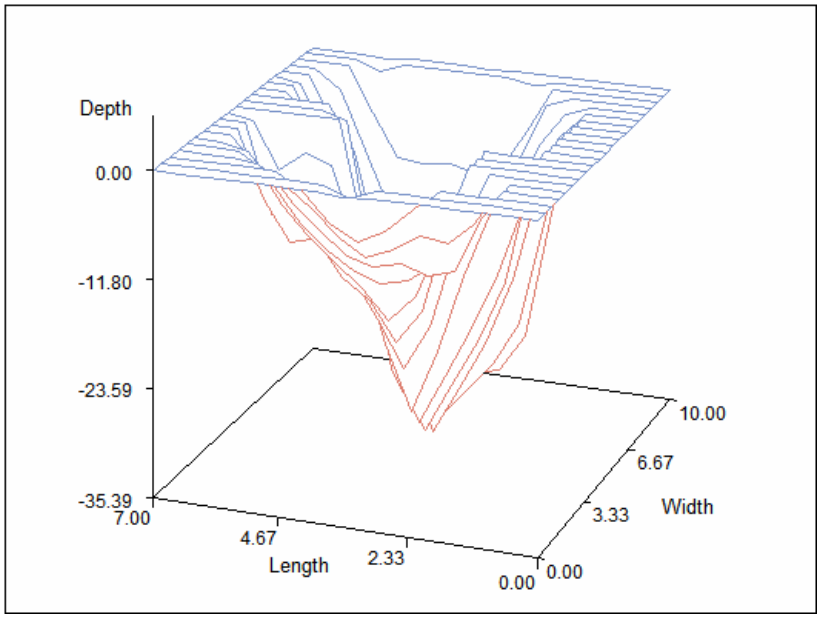
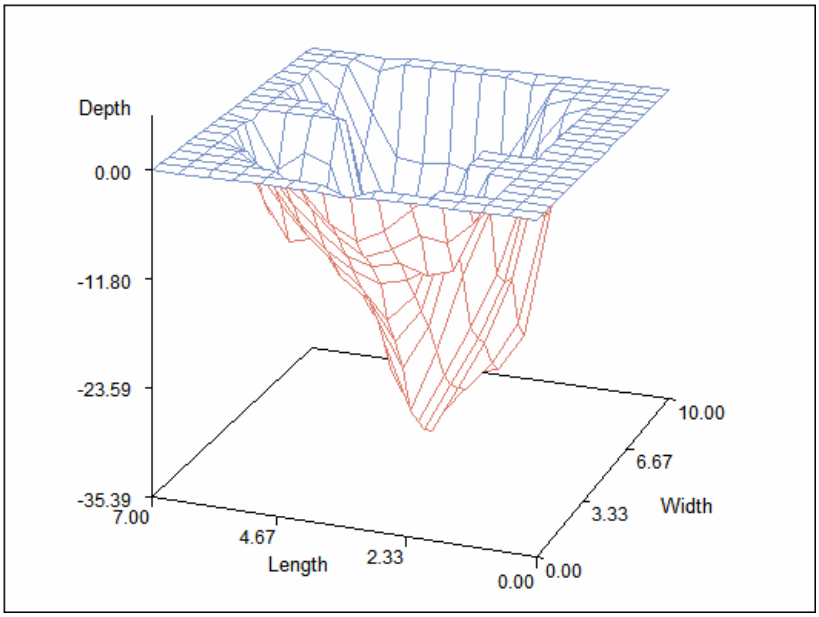


Figure 53.7 Surface Appearance for XYTYPE=3



SCATTER Statement

Creates three-dimensional scatter plots using values of three numeric variables from the input data set.

Requirements: One plot request is required.

Global statements: AXIS, BY, FOOTNOTE, GOPTIONS, NOTE, TITLE

Reminder: The procedure can include the FORMAT, LABEL, and WHERE statements.

Restriction: The AXIS statement is partially supported by Java and ActiveX devices only.

Alias: SCAT

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.

Syntax

SCATTER *y*x=z </option(s)>*;

Option(s) can be one or more options from any or all of the following categories:

- appearance options:
  - ANNOTATE=*annotate-data-set*
  - COLOR=*"data-point-color"* | *data-point-color-variable*
  - NONEEDLE
  - ROTATE=*angle-list*
  - SHAPE=*"symbol-name"* | *shape-variable*
  - SIZE=*symbol-size* | *size-variable*
  - TILT=*angle-list*
- axes options:
  - CAXIS=*axis-color*
  - CTEXT=*text-color*
  - GRID
  - NOAXIS | NOAXES
  - NOLABEL
  - XAXIS=*axis <1...99>*
  - XTICKNUM=*number-of-major-tick-marks*
  - YAXIS=*axis <1...99>*
  - YTICKNUM=*number-of-major-tick-marks*
  - ZAXIS=*axis <1...99>*
  - ZMAX=*maximum-value*
  - ZMIN=*minimum-value*
  - ZTICKNUM=*number-of-major-tick-marks*
- catalog entry description options:
  - DESCRIPTION=*"description"*
  - NAME=*"name"*

Required Arguments

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

Note: The SCATTER statement does not require a full grid of observations to generate a plot. △

Options

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 also: Chapter 29, "Using Annotate Data Sets," on page 641

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 it finds from the following list:

1 colors specified for labels and values on assigned axis statement

2 the color specified by the CTEXT= option in a SCATTER statement

3 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

Note: 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. △

Note: For Java and ActiveX only, specific text options specified in the AXIS statement override the CTEXT= option. △

Style reference: Color attribute of the GraphValueText and GraphLabelText elements

DESCRIPTION=*"description"*
specifies the description of the plot. The maximum length for *description* is 256 characters.

The descriptive text is displayed as follows:

- the description in the Results window
- the properties that you view form the Explorer window
- the description in the Explorer view of the catalog entry
- the Table of Contents that is generated when you use CONTENTS= on an ODS HTML statement, assuming 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 also: "Substituting BY Line Values in a Text String" on page 294

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 any graphics output file created. The name can be up to 256 characters long. If the name duplicates an existing name, SAS/GRAPH adds a number, or increments the last number used to create a unique graph name–for example G3D1.

For GRSEG entries:

- the name is truncated to eight characters
- the first character is always represented in uppercase
- all other characters are represented in lower case
- periods and blanks are converted to underscores

For Graphics Output files:

- SAS/GRAPH adds a number to the NAME= value, or increments the last number used.

Default: Procedure name

See also: "About Filename Indexing" on page 99

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 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

SHAPE=*"symbol-name"* | *shape-variable*
specifies a symbol name or a character variable whose values are symbol names. Symbols represent data points for scatter plots.

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:

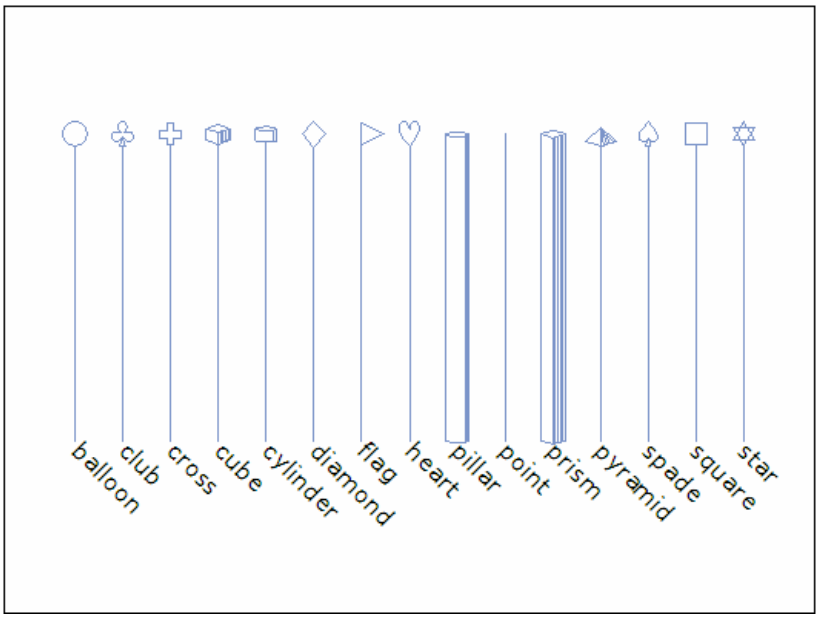
```
shape=class
```

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 53.8 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 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<1...<99>*
assigns an axis definition.

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<1...<99>*
assigns an axis definition.

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<1...<99>*
assigns an axis definition.

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 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)

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 53.8 on page 1558 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:

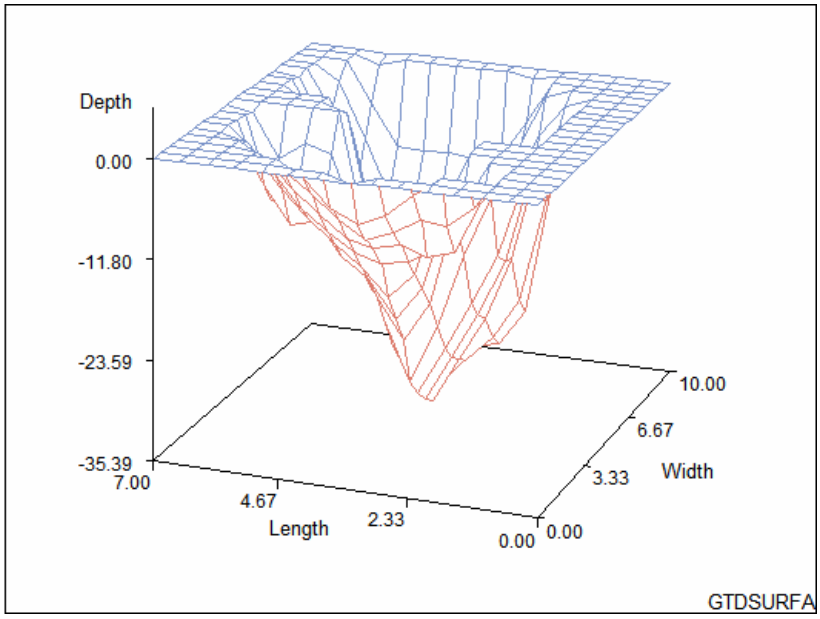
```
scatter y*x=z / size=type;
```

Examples

Example 1: Generating A Surface Plot

Procedure features:
PLOT statement

Sample library member: GTDSURFA



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.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote.

```
title "Surface Plot";
footnote j=r "GTDSURFA";
```

Generate the surface plot.

```
proc g3d data=sashelp.lake;
   plot length*width=depth;
run;
quit;
```

Example 2: Generating a Rotated Surface Plot

Procedure Features

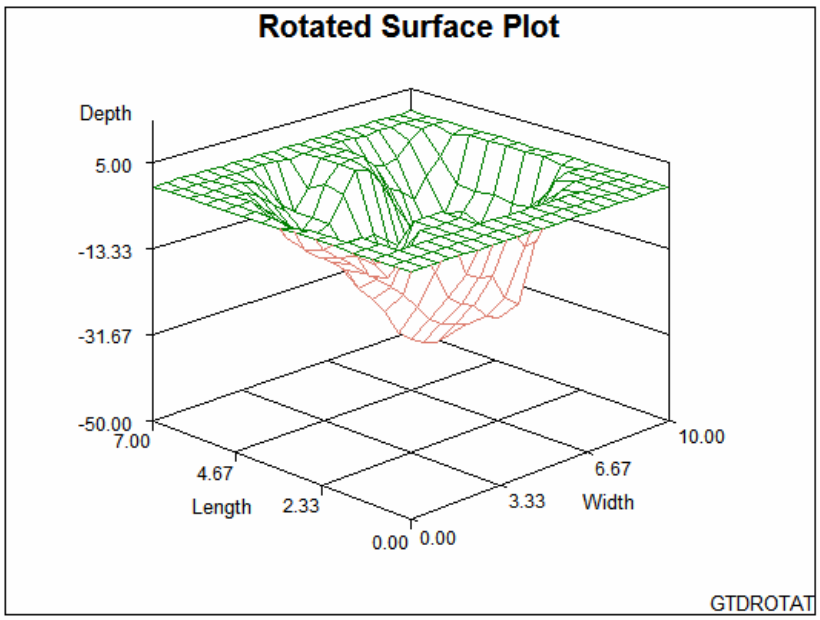
PLOT statement options:

CTOP=
GRID

ROTATE=
ZMAX=
ZMIN=

Data set: SASHELP.LAKE

Sample library member: GTDROTAT



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.

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote.

```
title "Rotated Surface Plot";
footnote j=r "GTDROTAT";
```

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

Procedure features:

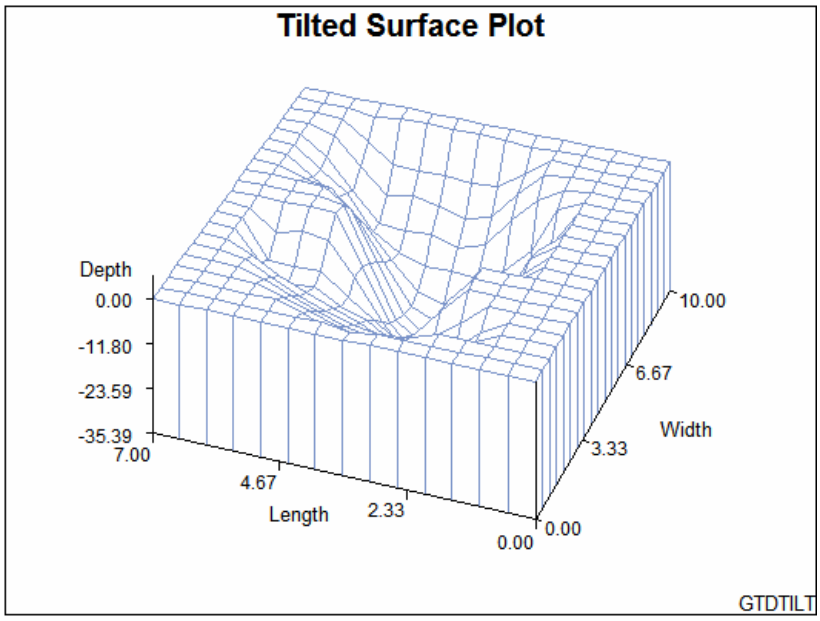
PLOT statement options:

SIDE

TILT=

Data set: SASHELP.LAKE

Sample library member: GTDTILT



Simple modifications displayed in Example 1 on page 1560 are generated by tilting the surface plot 30 degrees toward you, and adding a side wall.

Set the graphics environment.

```
goptions reset=all border;
```

Define title and footnote.

```
title "Tilted Surface Plot";
footnote j=r "GTDTILT";
```

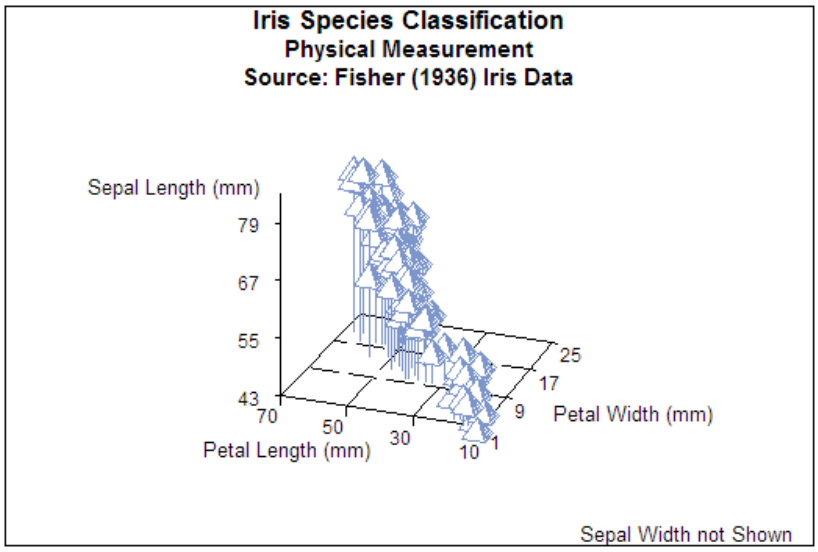
Generate the surface plot. The SIDE option draws a side wall for the graph. The TILT= option specifies a tilt angle of 15° for the plot. The initial rotation of 70° is not affected by the TILT= option.

```
proc g3d data=sashelp.lake;
   plot length*width=depth/
      side
      tilt=30;
run;
quit;
```

Example 4: Generating a Scatter Plot

Procedure features:
Scatter statement

Sample library member: GTDSCAT



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
- display the plot with the default ODS style

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.

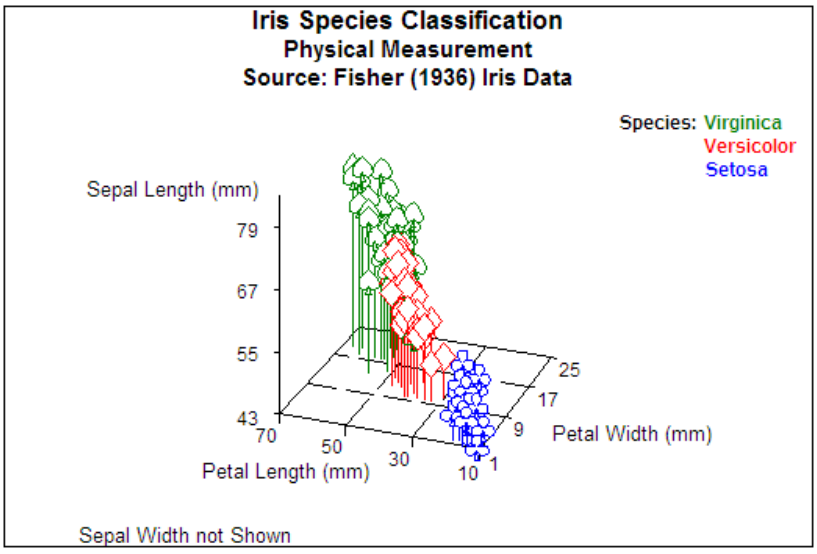
```
proc g3d data=sashelp.iris;
  scatter PetalLength*PetalWidth=SepalLength;
run;
quit;
```

Example 5: Generating a Scatter Plot with Modified Shapes

Procedure features:

Scatter statement
COLOR=
SHAPE=
DATA Step
NOTE statement

Sample library member: GTDSHAPA



This scatter plot modifies the results of measuring the petal length, petal width, and sepal length for the flowers of three species of irises by:

- using a DATA step to add a color variable and a shape variable to the data set
- using shapes to distinguish iris species
- using colors to distinguish iris species
- using a Note statement to simulate a legend

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;
run;
quit;
```

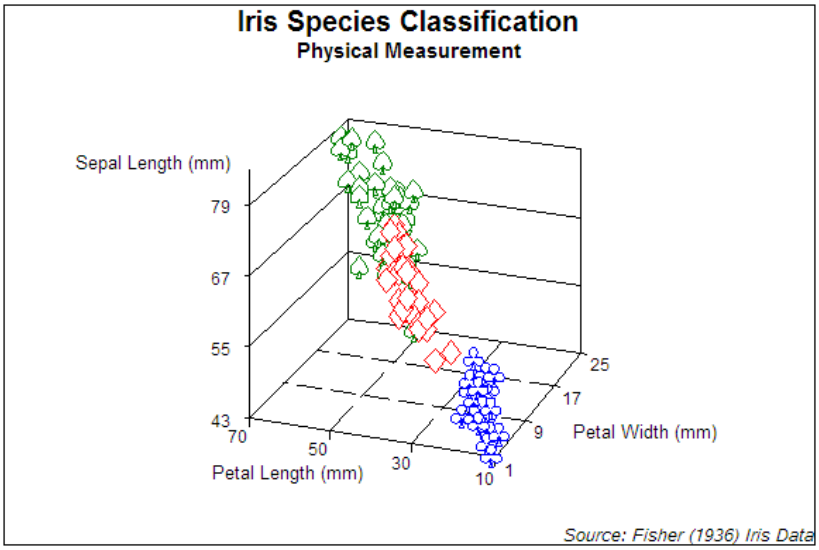
Example 6: Generating a Scatter Plot with Modified Shapes and a Grid

Procedure features:

Scatter statement
COLOR=

GRID
NONEEDLE
SHAPE=
DATA Step

Sample library member: GTDSHAPB



This scatter plot modifies the results of measuring the petal length, petal width, and sepal length for the flowers of three species of irises by:

- using a DATA step to add a color variable and a shape variable to the data set
- using shapes to distinguish iris species
- using colors to distinguish iris species
- removing needles from data points
- adding a grid

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.

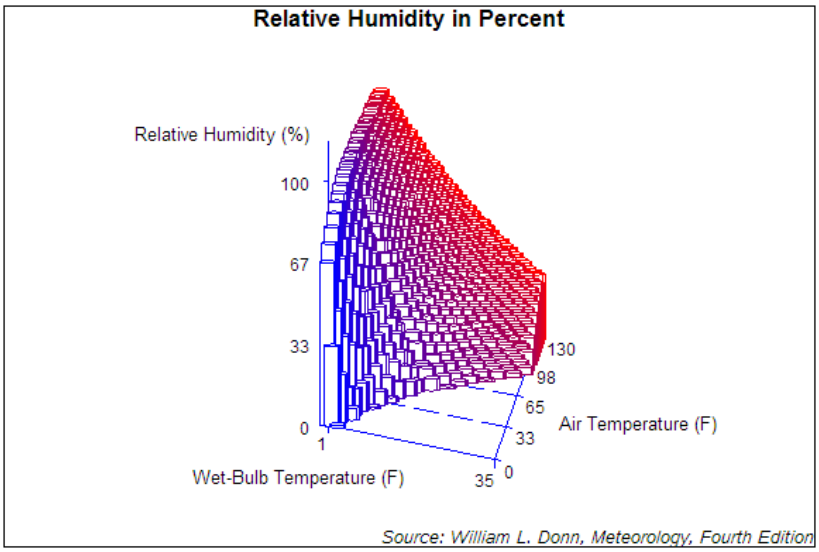
```
proc g3d data=iris;
    scatter PetalLength*PetalWidth=SepalLength/
      color=color
      shape=shape
      noneedle
      grid;
run;
quit;
```

Example 7: Generating a Rotated Scatter Plot with Modified Axes

Procedure features:

Scatter statement
CAXIS=
COLOR=
ROTATE=
SHAPE=
SIZE=
XTICKNUM=
YTICKNUM=
ZTICKNUM=
ZMAX=
ZMIN=

Sample library member: GTDROTSC



This scatter plot modifies the procedure defaults to:

- specify a shape for the data points
- classify the data by color
- specify blue as the axis color
- rotates the X-Y plane –15 degrees around the perpendicular Z axis.
- specifies five major tick marks for the Y-axis
- specifies two major tick marks for the X-axis
- specifies five major tick marks for the Z-axis
- specifies the zero as the minimum axis value for the Z-axis
- specifies the one hundred as the maximum axis value for the Z-axis

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles and footnote.

```
title1 "Relative Humidity in Percent";
footnote1 j=r f="Albany ANT/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)," in *Collected Algorithms from ACM,* New York: Association for Computing Machinery.

CHAPTER 54

The G3GRID Procedure

Overview

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:

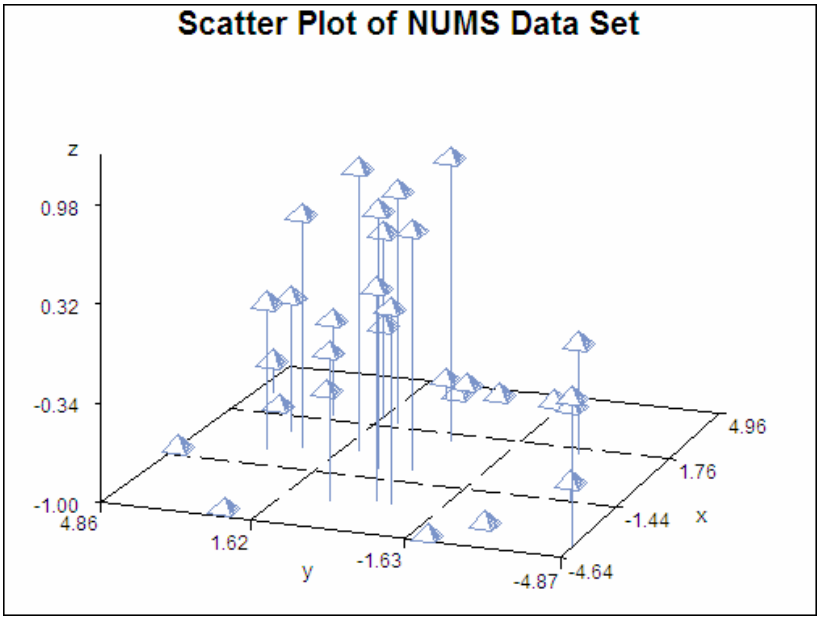
- create a rectangular grid of interpolated or smoothed values from irregularly spaced observations for use in a three-dimensional surface or contour plot
- complete a rectangular grid of interpolated or smoothed values for an input data set that has an insufficient number of observations to produce 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 54.1 on page 1572, and Figure 54.2 on page 1572 illustrate the effect of the G3GRID procedure on data.

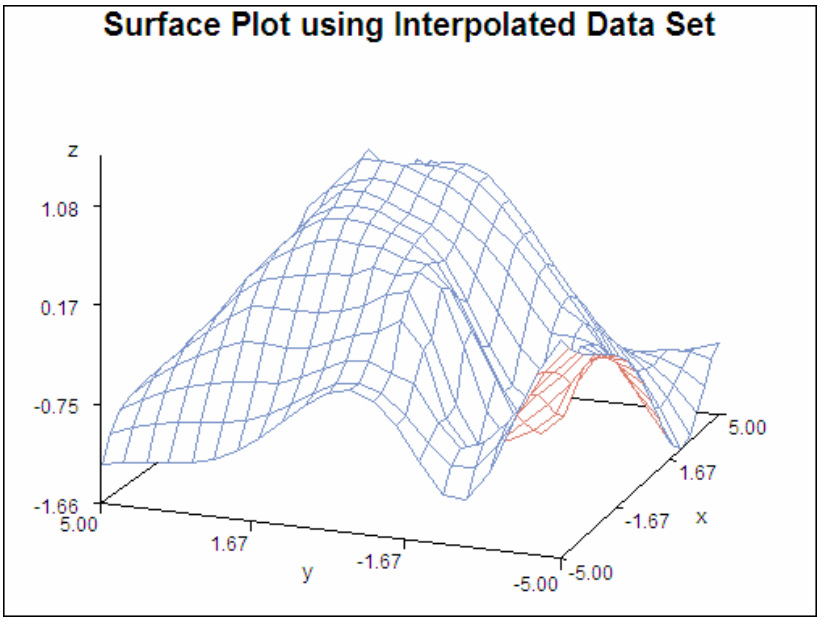
This figure shows a collection of data points, where $z=f(x,y)$. These points are randomly distributed, and cannot be displayed with a G3D surface plot, although they can be displayed with a scatter plot.

Figure 54.1 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 54.2 Surface Plot of Data Set After G3GRID Processing (gtdefin)



Concepts

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 $z\text{-}n$, 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 ($z - z\text{-}n$), 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

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 (Ripley 1981, p. 38), is used for the default method. 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, 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

$$|t - t_j| = \left((x - x_j)^2 + (y - y_j)^2 \right)^{1/2}$$

$$\mathbf{u}(x, y) = \Sigma_{j=1}^{n} c_j \mathrm{E}(t, t_j) + d_0 + d_1 x + d_2 y$$

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:

$$(\mathbf{E} + \mathbf{n}\lambda\mathbf{I})\,\mathbf{c} + \mathbf{T}\,\mathbf{d} = \mathbf{z}$$

, and

$$\mathbf{T}'\mathbf{c} = \mathbf{0}$$

where

E
is the $n \times n$ matrix 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 ,..., c_n)

z
is (z_1 ,..., 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 on 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 1574. 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.

Procedure Syntax

Requirements: Exactly one GRID statement is required.

Reminder: The procedure can include the BY statement.

PROC G3GRID <DATA=*input-data-set*>
<OUT=*output-data-set*>
<OUTTRI=*output-data-set*>;
GRID *grid-request* <*/option(s)*>;

PROC G3GRID Statement

Identifies the input data set. Can also specify one, or two output data sets.

Requirements: An input data set is required.

Syntax

PROC G3GRID <DATA=*input-data-set*>
<OUT=*output-data-set*>
<OUTTRI=*output-data-set*>;

Options

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 also: "SAS Data Sets" on page 54 and "The Input Data Set" on page 1573.

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 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.

Featured in: Example 1 on page 1581.

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.

Requirements: Exactly one grid request is required.

Syntax

GRID *grid-request* </*option(s)*>;

grid-request must be:

*y***x*=*z(s)*

option(s) can be one or more options from any or all of the following categories:

- grid options:

  AXIS1=*ascending-value-list*

  AXIS2=*ascending-value-list*

  NAXIS1=*n*

  NAXIS2=*n*

- interpolation options:

  JOIN

  NEAR=*n*

  PARTIAL

  SCALE | NOSCALE

  SMOOTH=*ascending-value-list*

  SPLINE

Required Arguments

y*x=z(s)
specifies three or more numeric variables from the input data set:

y
is one of the variables that forms the horizontal (*x*-*y*) plane

x
is another of the variables that forms 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;
```

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. 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*> >

Featured in: Example 1 on page 1581 and Example 4 on page 1588.

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*> >

Featured in: Example 1 on page 1581 and Example 4 on page 1588.

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 the 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

Featured in: Example 3 on page 1586.

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.

Featured in: Example 3 on page 1586.

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

$$\frac{1}{n}\sum_{j=1}^{n}\left(\mathrm{u}\left(\mathrm{x}_j,\mathrm{y}_j\right)-\mathrm{z}_j\right)^2+\lambda\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(\left(\frac{\partial^2\mathrm{u}}{\partial^2\mathrm{x}}\right)^2+2\left(\frac{\partial^2\mathrm{u}}{\partial\mathrm{x}\partial\mathrm{y}}\right)^2+\left(\frac{\partial^2\mathrm{u}}{\partial^2\mathrm{y}}\right)^2\right)\mathrm{dxdy}$$

where n is the number of data points, 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

Featured in: Example 2 on page 1584.

SPLINE

specifies the use of a bivariate spline (Harder and Desmarais 1972, Meinguet 1979, Green and Silverman 1994) 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.

Featured in: Example 2 on page 1584 and Example 4 on page 1588.

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 54.1 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

Options Specified	Number of Observations in Output Data Set
NAXIS1=	(value of NAXIS1=) * 11
NAXIS2=	(value of NAXIS2=) * 11
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

Example 1: Using the Default Interpolation Method

Procedure features:

G3GRID statement options:

OUT=

GRID statement options:

AXIS1=
AXIS2=

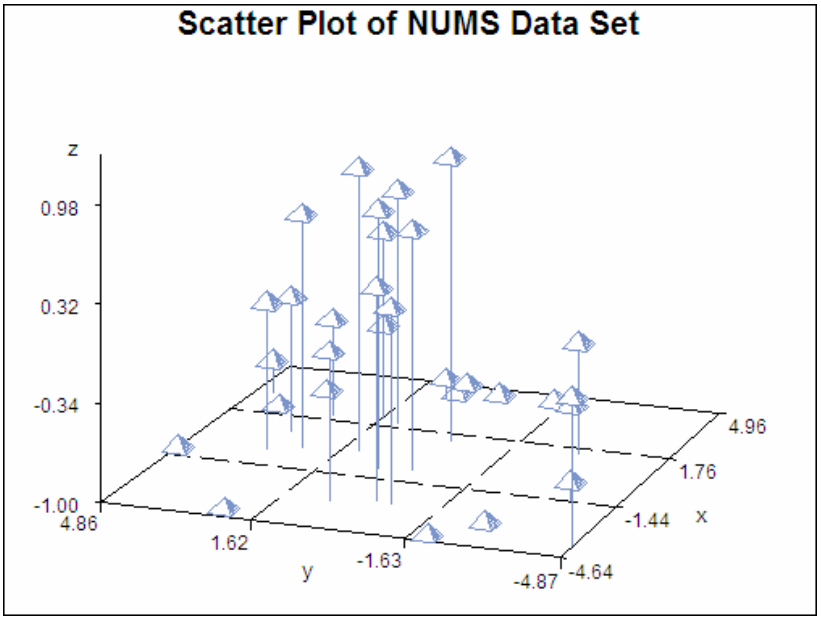
Other features:

DATA step

G3D procedure

Sample library member: GTGDEFIN

Figure 54.3 Scatter Plot of NUMS Data Set (gtgdefin)

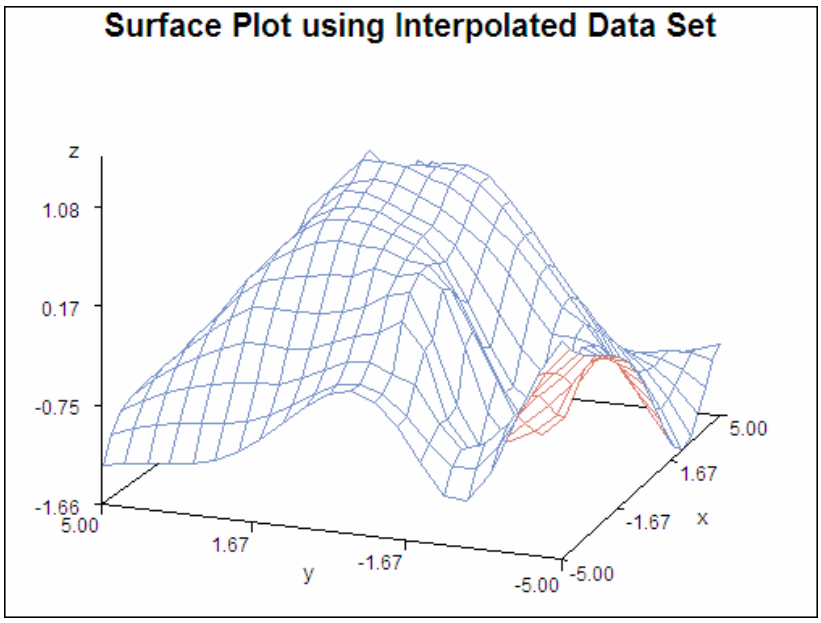


This example demonstrates the default interpolation method that is used by the GRID statement. The example first generates a scatter plot of random data to show the concentration of data values before processing the data set with the G3GRID procedure. The original data does not contain enough combinations of x, y and z values to:

- generate a surface plot with the G3D procedure
- generate a contour plot with the GCONTOUR procedure

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.

Figure 54.4 Surface Plot using Interpolated Data Set (gtgdefin)



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(35)-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

Procedure features:

GRID statement options:

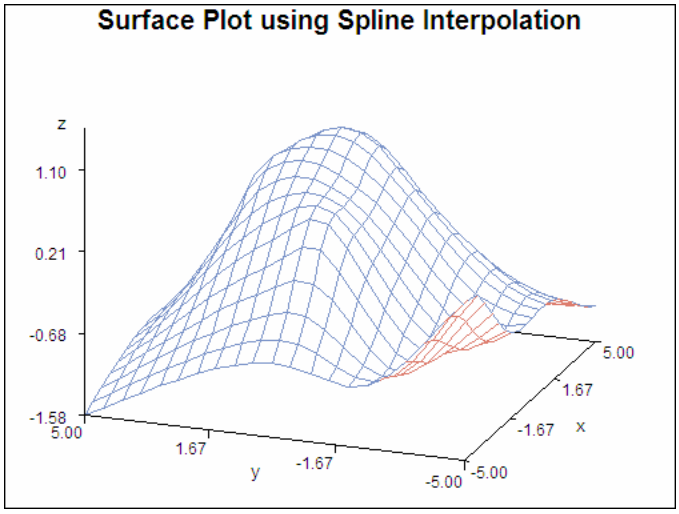
SMOOTH=
SPLINE

Data set: NUMS (see Example 1 on page 1581)

Sample library member: GTGSISS

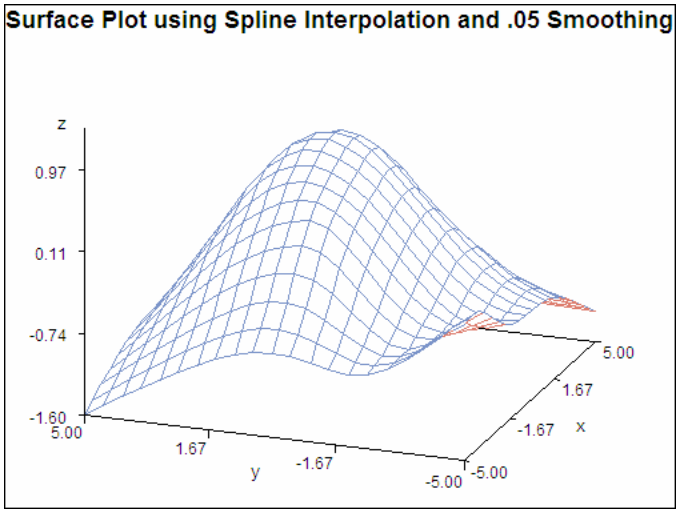
This example extends Example 1 on page 1581 to specify the SPLINE option on the GRID statement. The output data set, when used in PROC G3D, generates a smoother surface plot.

Figure 54.5 Surface Plot using Spline Interpolation (gtgsiss)



The following plot extends Example 1 on page 1581 to specify the SPLINE option, and the SMOOTH= option on 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.

Figure 54.6 Surface Plot using Spline Interpolation and .05 Smoothing (gtgsiss)



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

Procedure features:

GRID statement options:

NEAR
PARTIAL

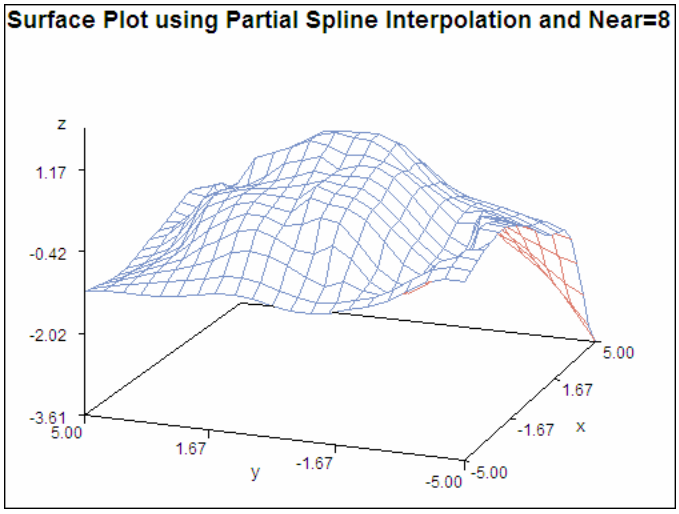
Data set: NUMS (see Example 1 on page 1581)

Sample library member: GTGPART

This example specifies a partial spline interpolation on 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 on page 1581
- does not generate the smoothness of the surface plot that results from the spline interpolation shown in Example 2 on page 1584

Figure 54.7 Surface Plot using Partial Spline Interpolation (gtgpart)



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

Procedure features:

GRID statement options:

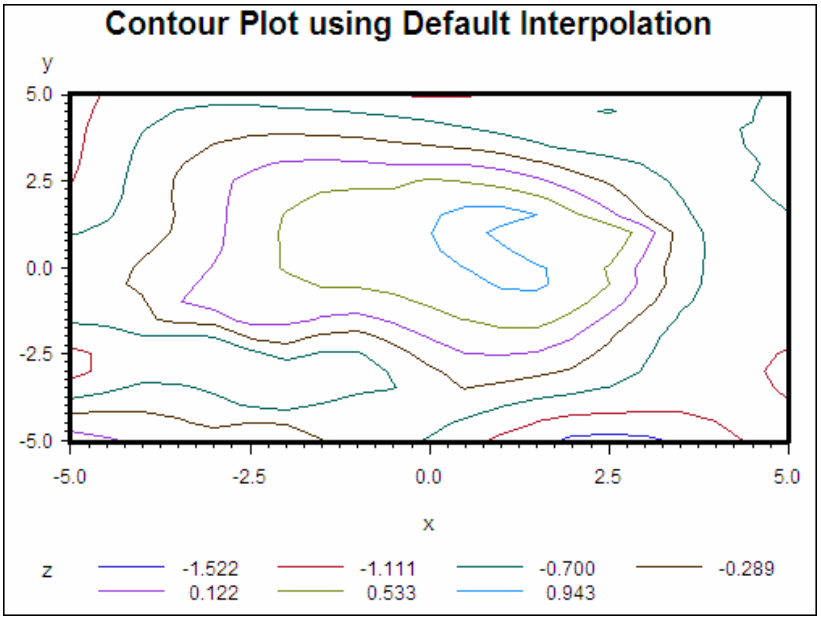
AXIS1=
AXIS2=
SPLINE

Data set: NUMS (see Example 1 on page 1581)

Sample library member: GTGSPLIN

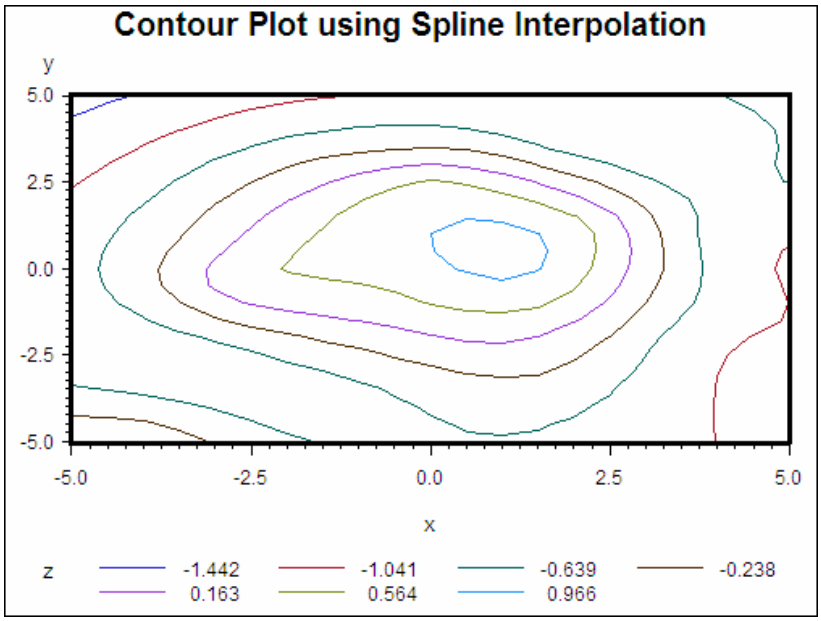
This example demonstrates the default interpolation method when used by the GCONTOUR procedure to generate a contour plot from the resulting output data set.

Figure 54.8 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.

Figure 54.9 Contour Plot Using Spline Interpolation (gtgsplin)



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: 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: John Wiley & Sons, Inc.

Wahba, G. (1990), *Spline Models for Observational Data*, Philadelphia: SIAM.

CHAPTER

55

The MAPIMPORT Procedure

Overview

The MAPIMPORT procedure enables you to import ESRI shapefiles (spatial data formats) and process the SHP files into SAS/GRAPH traditional map data sets. See "About Traditional Data Sets" on page 1244 for more information.

The MAPIMPORT procedure does not produce any graphics output. Instead, it produces an output map data set, which can be used with the GMAP procedure.

The shapefiles file types are described in the following table:

Table 55.1 Shapefiles File Types

File Extension	Description
.dbf	identification information (field-identifier names and values) assigned to specific polygon(s)
.shx	shape information for the polygon(s) that compose the map. *Note:* These files are used with SHP files and cannot be imported by themselves. △
.shp	combines the shape information for the polygon(s) that compose the map and the identification information (field-identifier names and values) assigned to the specific polygon(s)

Note: If you import a very highly-detailed map, then the GMAP procedure might produce extraneous lines when drawing it. To avoid this issue, use the GREDUCE procedure to reduce the number of map points. △

Procedure Syntax

Requirements: The name and location of an output data set and the complete path for the input data file.

Reminder: The single quotes surrounding field identifiers are optional when the field identifiers follow the SAS naming convention. Single quotes are required for field identifiers that are non-standard SAS names. When field identifiers placed in single quotes are non-standard SAS names, the field identifiers are converted to a standard SAS name in the traditional map data set. For more information about the standard SAS naming convention, see names in the SAS Language in the *SAS Language Reference: Concepts*. For more information on how invalid field identifiers placed in single quotes are renamed, see the SAS System option VALIDVARNAME in the *SAS/ACCESS for Relational Databases: Reference*.

PROC MAPIMPORT OUT= *map-data-set* DATAFILE= *'path-to-shapefile'*
<CONTENTS> <CREATE_ID_>;

EXCLUDE *'field-identifier(s)'*;

ID *'field-identifier(s)'*;

RENAME *'field-identifier-1'* = *variable-name-1* < ... *'field-identifier-n'* = *variable-name-n*>;

SELECT *'field-identifier(s)'*;

PROC MAPIMPORT Statement

Identifies the input ESRI shapefile and converts this map into a SAS/GRAPH map data set.

Requirements: The name and location of an output data set and the complete path for the input data file.

PROC MAPIMPORT OUT= *map-data-set* DATAFILE= *'path-to-shapefile'*
<CONTENTS> <CREATE_ID_>;

Required Arguments

OUT= *map-data-set*
specifies the name of the output map data set that is created.

DATAFILE= *'path-to-shapefile'*
specifies the path and filename of the shapefile that is read and processed.

Alias: INFILE=

Note: By default, all of the fields in a shapefile are included in the output map data set. To include only specific fields in the output map data set, use the SELECT statement. To exclude specific fields from the output map data set, use the EXCLUDE statement. △

Options

CONTENTS
displays information about the shapefile, including field identifier names and types.

CREATE_ID_
creates a map ID variable named _ID_ with a unique value for each polygon in the map. This variable is created automatically if the DBF file is missing.

Interaction: This statement has no effect if you also specify the ID statement.

EXCLUDE Statement

Specifies one or more fields from the shapefile thatare excluded from the output map data set.

Requirements: At least one *field-identifier* is required.

Restriction: If you specify conflicting values for the EXCLUDE and SELECT statements, then the MAPIMPORT procedure produces an error.

Restriction: If you specify the same field identifier on the EXCLUDE statement and on the ID statement, then the MAPIMPORT procedure produces an error.

EXCLUDE '*field-identifier(s)*';

Required Arguments

'field-identifier(s)'
specifies one or more fields from the shapefile that are excluded from the output map data set. All of the fields that you do not specify are included in the output map data set.

If you do not specify the EXCLUDE statement or the SELECT statement, then all of the fields from the shapefile are included in the output map data set.

ID Statement

Reorders the map polygons by one or more identifier fields.

Requirements At least one *field-identifier* is required.

Interaction: The CREATE_ID option on the PROC MAPIMPORT statement has no effect when you also specify the ID option.

ID '*field-identifier(s)*';

Required Arguments

'field-identifier(s)'
specifies one or more fields in the shape file that identify the polygons in the map. The values of the fields that you specify are used to reorder the map polygons and assign segment numbers in the output map data set.

When you do not specify the ID statement, the MAPIMPORT procedure uses the existing polygon order for the output map data set.

You might want to use the ID statement when the default output map data set does not draw properly in the GMAP procedure. If the ID variable that you specify in the GMAP procedure is not unique for each polygon, then extraneous lines might appear in your GMAP output. To ensure that the ID variable is unique for each polygon, specify the same ID statement in both the MAPIMPORT and GMAP procedures.

RENAME Statement

Renames variables in the output map data set that correspond to specific fields in the shapefile.

Requirements: At least one *field-identifier* and *variable-name* pair are required.

RENAME *'field-identifier-1' = variable-name–1 <... 'field-identifier-n' = SAS-variable-n>*

Required Arguments

'field-identifier' = variable-name
assigns a variable name in the output map data set for a field in the shapefile. You can specify multiple field identifier and variable name pairs, separated by a space.

For example, the following code renames the STNAME field to STATE, and the FIPSTATE field to STATE_FIPS:

```
rename "stname" = state "fipstate" = state_fips;
```

By default, when you do not specify the RENAME statement, the MAPIMPORT procedure uses the field name in the shapefile as the variable name in the output map data set. However, if the field name is not a valid SAS variable name, then the variable name is modified in the output map data set. For more information about valid SAS variable names, see the "Rules for Words and Names in the SAS Language" chapter of *SAS Language Reference: Concepts*.

SELECT Statement

Selects the fields from the shapefile that are included in the output map data set.

Requirements: At least one *field-identifier* is required.

Restriction: If you specify conflicting values for the EXCLUDE and SELECT statements, then the MAPIMPORT procedure produces an error.

SELECT '*field-identifier(s)*';

Required Arguments

'field-identifier(s)'
specifies one or more fields from the shapefile that are included in the output map data set. If you do not use the SELECT statement or the EXCLUDE statement, then all of the fields from the shapefile are included in the output map data set.

For field identifiers that are not valid SAS variable names, the MAPIMPORT procedure changes the name of the variable in the output map data set automatically. For more information about valid SAS variable names, see the "Rules for Words and Names in the SAS Language" chapter of *SAS Language Reference: Concepts*.

Examples

The following examples use shapefiles with the .shp and .dbf extensions. Replace the shapefiles locations, filenames, and field-identifiers with information from your shapefiles to run these examples.

Example 1: Including All Variables from the SHP Shapefile

In the following example,World30.shp contains polygons that compose a political boundary world map. All the field identifiers in the World30.shp file are included in the traditional map data set, MYWORLD.

```
PROC MAPIMPORT OUT=myworld DATAFILE="C:\world30.shp";
run;
```

Example 2: Including Selected Variables from the SHP Shapefile

In the following example, the STATES.SHP file contains polygons that compose the political boundaries of a U.S. states map. Only the STATE_FIPS (the state FIPS codes), STATE_NAME (the state name), and STATE_ABBR (the two letter state abbreviation) variables are included in the traditional map data set, MYSTATES. STATE_FIPS is renamed FIPS, STATE_NAME is renamed STATE, and STATE_ABBR is renamed ABBREV in the MYSTATES map data set.

```
PROC MAPIMPORT OUT=mystates DATAFILE="C:\states.shp";
     SELECT STATE_FIPS STATE_NAME STATE_ABBR;
     RENAME STATE_FIPS=FIPS STATE_NAME=STATE STATE_ABBR=ABBREV;
run;
```

Example 3: Excluding a Variable from the SHP Shapefile

In the following example, the STATES.SHP file contains polygons that compose the political boundaries of a U.S. state map. The variable OTHER is excluded from the traditional map data set, MYSTATES2.

```
PROC MAPIMPORT OUT=mystates2 DATAFILE="C:\states.shp";
     EXCLUDE OTHER;
run;
```

Example 4: Using the ID Statement

In the following example, the shapefile is a ZCTA file from the US Census Bureau that contain polygons that are based on ZIP codes. The ZCTA field is the identifier that you want to use, but the polygons in the shapefile do not have unique values for ZCTA. If you do not specify the ID statement, then the GMAP procedure draws extra lines between the map areas for ZCTA.

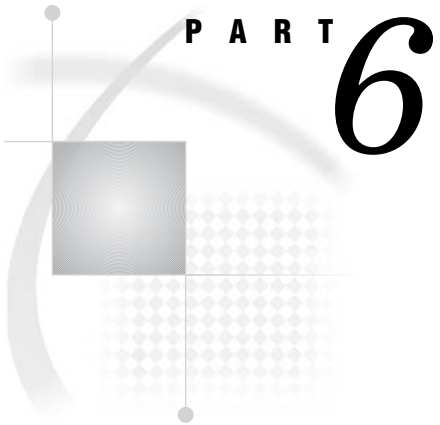
By identifying the ZCTA field in the ID statement, you ensure that the polygons for each value of ZCTA are grouped together and assigned different SEGMENT values in the output map data set. The GMAP procedure can now draw the map areas for ZCTA correctly.

```
proc mapimport out=myzcta datafile="c:\zt06_d00.shp";
  id zcta;
run;
```

Example 5: Including Selected Variables from the DBF Shapefile

In the following example, the STATES.DBF file contains the identification information (field-identifier names and values) applied to the U.S. states polygon map. Only the STATE_FIPS (the state FIPS codes), STATE_NAME (the state names), and STATE_ABBR (the two letter state abbreviations) variables are included in the traditional map data set, MYDATA. STATE_FIPS is renamed FIPS, STATE_NAME is renamed STATE, and STATE_ABBR is renamed ABBREV in the MYDATA map data set.

```
PROC MAPIMPORT OUT=mydata DATAFILE="C:\states.dbf";
     SELECT STATE_FIPS STATE_NAME STATE_ABBR;
     RENAME STATE_FIPS=FIPS STATE_NAME=STATE STATE_ABBR=ABBREV;
run;
```



PART 6

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 Java and ActiveX 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 may also indicate that an option works differently for the other devices than it does for the Java and ActiveX device drivers, or that an option works for one or more applets but not for all. For a complete description of each option or variable, refer to the documentation for the option or variable.

Global Statements

AXIS Statement

Table A1.1 ActiveX and Java Support for the AXIS Statement

Option	Supported by ActiveX?	Supported by Java?
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
STYLE=	Yes	Yes

Option	Supported by ActiveX?	Supported by Java?
VALUE=	Yes	Yes (partial)
WIDTH=	Yes (partial)	No

Text Description Suboptions

Text description suboptions are used by the LABEL=, REFLABEL=, and VALUE= options.

Table A1.2 ActiveX and Java Support for AXIS Text Description Suboptions

Option	Supported by ActiveX?	Supported by Java?
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

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 A1.3 ActiveX and Java Support for Tick Mark Description Suboptions

Option	Supported by ActiveX?	Supported by Java?
COLOR= C=	Yes	Yes
HEIGHT= H=	No	No

Option	Supported by ActiveX?	Supported by Java?
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 195 for more information.

Table A1.4 ActiveX and Java Support for the GOPTIONS Statement

Option	Supported by ActiveX?	Supported by Java?
ACCESSIBLE	Yes	Yes
ADMGDF NOADMGDF	No	No
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
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

Option	Supported by ActiveX?	Supported by Java?
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
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
GCLASS=	No	No
GCOPIES=	No	No
GDDMCOPY=	No	No
GDDMNICKNAME=	No	No
GDDMTOKEN=	No	No
GDEST=	No	No
GEND=	No	No
GEPILOG=	No	No
GFORMS=	No	No

Option	Supported by ActiveX?	Supported by Java?
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
GSTART=	No	No
GUNIT=	Yes (partial)	Yes (partial)
GWAIT=	No	No
GWRITER=	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
PAPERSOURCE=	No	No

Option	Supported by ActiveX?	Supported by Java?
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
ROTATE NOROTATE	No	No
SIMFONT=	No	No
SPEED=	No	No
SWAP NOSWAP	No	No
SWFONTRENDER	No	No
SYMBOL NOSYMBOL	No	No
TARGETDEVICE=	No	No

Option	Supported by ActiveX?	Supported by Java?
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 A1.5 ActiveX and Java Support for the LEGEND Statement

Option	Supported by ActiveX?	Supported by Java?
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)

Option	Supported by ActiveX?	Supported by Java?
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 A1.6 ActiveX and Java Support for LEGEND Text Description Suboptions

Option	Supported by ActiveX?	Supported by Java?
COLOR= C=	Yes	Yes
FONT= F=	Yes	Yes
HEIGHT= H=	Yes	Yes
JUSTIFY= J=	Yes	Yes
POSITION=	Yes (partial)	No
TICK= T=	Yes	Yes

PATTERN Statement

Table A1.7 ActiveX and Java Support for the PATTERN Statement

Option	Supported by ActiveX?	Supported by Java?
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)

Option	Supported by ActiveX?	Supported by Java?
VALUE=*map*/*plot-pattern* V=*map*/*plot-pattern*	Yes (partial)	Yes (partial)
VALUE=*pie*/*star-pattern* V=*pie*/*star-pattern*	Yes (partial)	Yes (partial)

SYMBOL Statement

Table A1.8 ActiveX and Java Support for the SYMBOL Statement

Option	Supported by ActiveX?	Supported by Java?
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
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

Option	Supported by ActiveX?	Supported by Java?
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	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)

POINTLABEL= Label Description Options

Table A1.9 ActiveX and Java Support for POINTLABEL Description Suboptions

Option	Supported by ActiveX?	Supported by Java?
COLOR= C=	Yes	No
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

Table A1.10 ActiveX and Java Support for TITLE and FOOTNOTE Statements

Option	Supported by ActiveX?	Supported by Java?
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
LSPACE=	No	No
MOVE=	No	No
ROTATE=	No	No
UNDERLIN=	Yes (partial)	Yes (partial)

PROC GAREABAR

Table A1.11 ActiveX and Java Support for GAREABAR

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC GAREABAR	DATA=	Yes	No
HBAR and VBAR	CFR FRAME=	Yes	No
	CTEXT=	Yes	No
	DISCRETE	Yes	No
	FRAME NOFRAME	Yes	No
	NAME=	Yes	No

Statement	Option	Supported by ActiveX?	Supported by Java?
	RSTAT= RESPSTAT= RESPONSESTAT=	Yes	No
	SUBGROUP=	Yes	No
	SUMVAR=	Yes	No
	WSTAT= WIDTHSTAT=	Yes	No

PROC GBARLINE

Table A1.12 ActiveX and Java Support for PROC GBARLINE

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC GBARLINE	ANNOTATE= ANNO=	Yes	No
	DATA=	Yes	No
	IMAGEMAP=	No	No
BAR	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
	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

Statement	Option	Supported by ActiveX?	Supported by Java?
	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
	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

Statement	Option	Supported by ActiveX?	Supported by Java?
	SPACE=	Yes	No
	SUM	Yes	No
	SUMVAR=	Yes	No
	TYPE=	Yes	No
	WIDTH=	Yes	No
	WOUTLINE=	Yes	No
PLOT	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

Table A1.13 ActiveX and Java Support for PROC GCHART

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC GCHART	ANNOTATE= ANNO=	Yes	Yes
	DATA=	Yes	Yes
	GOUT=	Yes	Yes
	IMAGEMAP=	No	No
BLOCK	ANNOTATE= ANNO=	Yes	Yes
	BLOCKMAX=	No	No
	CAXIS=	Yes	Yes
	COUTLINE=	Yes (partial)	Yes (partial)
	CTEXT=	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
	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
	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
	WOUTLINE=	Yes	No
HBAR, HBAR3D, VBAR, and VBAR3D	ANNOTATE= ANNO=	Yes	Yes
	ASCENDING	Yes	Yes
	AUTOREF	Yes	Yes
	AXIS=	Yes	Yes
	CAUTOREF=	Yes	Yes
	CAXIS=	Yes	Yes
	CFRAME= CFR=	Yes	Yes
	CERROR=	Yes	Yes
	CFREQ	Yes	Yes
	CFREQLABEL=	No	No
	CLIPREF	Yes	Yes
	CLM=	Yes	Yes
	COUTLINE=	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
	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
	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

Statement	Option	Supported by ActiveX?	Supported by Java?
	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
	PERCENT PCT	Yes	Yes
	PERCENTLABEL=	No	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
	WIDTH=	Yes	Yes
	WOUTLINE=	Yes	No
PIE, PIE3D, and DONUT	ACROSS=	Yes	Yes
	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

Statement	Option	Supported by ActiveX?	Supported by Java?
	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
	INVISIBLE=	Yes	Yes
	JSTYLE	Yes	Yes
	LABEL=	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

Statement	Option	Supported by ActiveX?	Supported by Java?
	SUMVAR=	Yes	Yes
	TYPE=	Yes	Yes
	VALUE=	Yes	Yes
	WOUTLINE=	Yes	No
STAR		No	No

Text Description Suboptions

Text description suboptions are used by the LABEL= option in the DONUT statement.

Table A1.14 ActiveX and Java Support for LABEL Text Description Suboptions

Option	Supported by ActiveX?	Supported by Java?
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

PROC GCONTOUR

Table A1.15 ActiveX and Java Support for PROC GCONTOUR

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC GCONTOUR	ANNOTATE= ANNO=	Yes (partial)	Yes (partial)
	DATA=	Yes	Yes
	GOUT=	No	No
	INCOMPLETE	No	No
PLOT	ANNOTATE= ANNO=	Yes (partial)	Yes (partial)

Statement	Option	Supported by ActiveX?	Supported by Java?
	AUTOHREF	Yes	No
	AUTOLABEL=	No	No
	AUTOVREF	Yes	No
	CAUTOHREF=	Yes	No
	CAUTOVREF=	Yes	No
	CAXIS=	Yes	Yes (partial)
	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
	NOLEGEND	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
	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

Table A1.16 ActiveX and Java Support for PROC GMAP

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC GMAP	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		Yes	Yes
	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		Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
BLOCK	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
	RELZERO	Yes	Yes
	SHAPE=	Yes	Yes
	STATISTIC=	Yes	Yes
	STRETCH	No	No
	UNIFORM	Yes	No
	WOUTLINE=	Yes	Yes
	XSIZE= YSIZE=	No	No
	XVIEW= YVIEW= ZVIEW=	Yes	Yes (partial)
CHORO	ANNOTATE=	Yes	Yes
	CDEFAULT=	Yes	No
	CEMPTY=	Yes	No
	COUTLINE=	Yes	Yes
	CTEXT=	Yes	Yes
	DESCRIPTION=	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
	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
	STATFMT=	Yes	Yes
	STATISTIC=	Yes	Yes
	STRETCH	No	No
	UNIFORM	Yes	No
	WOUTLINE=	Yes	Yes
	XSIZE= YSIZE=	No	No
PRISM	ANNOTATE=	Yes	Yes
	AREA=	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
	STATFMT=	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
	STATISTIC=	Yes	Yes
	STRETCH	No	No
	UNIFORM	Yes	No
	WOUTLINE=	No	Yes
	XLIGHT= YLIGHT=	No	No
	XSIZE= YSIZE=	No	No
	XVIEW= YVIEW= ZVIEW=	Yes	Yes (partial)
SURFACE		No	No

PROC GPLOT

When used with the JAVA or JAVAMETA device driver, the BUBBLE statement must have at least one axis that is assigned to a numeric variable.

Table A1.17 ActiveX and Java Support for PROC GPLOT

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC GPLOT	ANNOTATE= ANNO=	Yes	Yes
	DATA=	Yes	Yes
	GOUT=	Yes	Yes
	IMAGEMAP=	Yes	Yes
	UNIFORM	Yes (partial)	No
BUBBLE	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

Statement	Option	Supported by ActiveX?	Supported by Java?
	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)
	HMINOR= HM=	Yes	Yes
	HREF=	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
	VAXIS=	Yes (partial)	Yes (partial)
	VMINOR= VM=	Yes	Yes
	VREF=	Yes	Yes
	VREVERSE	Yes	Yes
	VZERO	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
BUBBLE2	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)
	IFRAME	Yes	Noc
	LAUTOVREF=	Yes	Yes
	LVREF= LV=	Yes	Yes
	NOAXIS NOAXES	Yes	Yes
	VAXIS=	Yes (partial)	Yes (partial)
	VMINOR= VM=	Yes	Yes
	VREF=	Yes	Yes
	VREVERSE	Yes	Yes
	VZERO	Yes	Yes
PLOT	ANNOTATE=	Yes	Yes
	AREAS=	Yes	Yes (partial)
	AUTOHREF	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
	AUTOVREF	Yes	Yes
	CAUTOHREF=	Yes	Yes
	CAUTOVREF=	Yes	Yes
	CAXIS= CA=	Yes	Yes
	CFRAME= CFR=	Yes	Yes
	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

Statement	Option	Supported by ActiveX?	Supported by Java?
	NOAXIS NOAXES	Yes	Yes
	NOLEGEND	Yes	Yes
	OVERLAY	Yes	Yes (partial)
	REGEQN	No	Yes
	SKIPMISS	Yes	Yes
	VAXIS=	Yes (partial)	Yes (partial)
	VMINOR= VM=	Yes	Yes
	VREF=	Yes	Yes
	VREVERSE	Yes	Yes
	VZERO	Yes	Yes
PLOT2	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
	CFRAME= CFR=	Yes	Yes
	COUTLINE=	Yes	No
	CTEXT= C=	Yes	Yes
	CVREF= CV=	Yes	Yes
	FRAME NOFRAME FR NOFR	Yes	Yes
	GRID	Yes	Yes
	HTML=	Yes (partial)	Yes (partial)
	HTML_LEGEND=	No	No
	LAUTOVREF=	Yes	Yes
	LEGEND=	Yes	Yes
	LVREF= LV=	Yes	Yes
	NAME=	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
	NOAXIS NOAXES	Yes	Yes
	NOLEGEND	Yes	Yes
	OVERLAY	Yes	Yes (partial)
	REGEQN	No	Yes
	SKIPMISS	Yes	Yes
	VAXIS=	Yes (partial)	Yes (partial)
	VMINOR= VM=	Yes	Yes
	VREF=	Yes	Yes
	VREVERSE	Yes	Yes
	VZERO	Yes	Yes

PROC GRADAR

Table A1.18 ActiveX and Java Support for PROC GRADAR

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC GRADAR	ANNOTATE=	Yes	No
	DATA=	Yes	No
	GOUT=	Yes	No
CHART	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
	CSTARCIRCLES= CSTARCIRCLE=	Yes	No

Statement	Option	Supported by ActiveX?	Supported by Java?
	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
	MODE=	Yes	No
	NAME=	Yes	No
	NCOLS= NCOL=	No	No
	NLEVELS=	Yes	No
	NOLEGEND	Yes	No
	NOZEROREF	Yes	No
	NROWS= NROW=	No	No

Statement	Option	Supported by ActiveX?	Supported by Java?
	ORDERACROSS=	No	No
	OTHER=	Yes	No
	OVERLAY= OVERLAYVAR=	Yes	No
	SPEED	Yes	No
	SPIDERWEB SPIDER	Yes	No
	SPKLABEL=	Yes	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
	SUMVAR=	Yes	No
	TILELEGEND=	No	No
	TILELEGLABEL=	No	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 A1.19 ActiveX and Java Support for PROC GTILE

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC TILE	DATA=	Yes	Yes
TILEBY		Yes	Yes
FLOW		Yes	Yes
	BASELINE	Yes	Yes
	CMISSING=	Yes (partial)	Yes
	COLORRAMP=	Yes	Yes
	COLORVAR=	Yes	Yes
	DESCRIPTION=	Yes	Yes
	DETAILLEVEL=	Yes	Yes
	LABELLEVEL=	Yes	Yes
	NAME=	Yes	Yes
TILE			
TOGGLE			

PROC G3D

Table A1.20 ActiveX and Java Support for PROC G3D

Statement	Option	Supported by ActiveX?	Supported by Java?
PROC G3D	ANNOTATE= ANNO=	Yes	Yes
	DATA=	Yes	Yes
	GOUT=	Yes	Yes
PLOT	ANNOTATE= ANNO=	Yes	Yes
	CAXIS=	Yes	Yes
	CBOTTOM=	Yes	No
	CTEXT=	Yes	Yes
	CTOP=	Yes	Yes

Statement	Option	Supported by ActiveX?	Supported by Java?
	DESCRIPTION= DES=	Yes (partial)	Yes (partial)
	GRID	Yes	No
	NAME=	Yes	Yes
	NOAXIS NOAXES	Yes	Yes
	NOLABEL	Yes	Yes
	ROTATE=	Yes (partial)	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	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=	Yes (partial)	No
	SHAPE=	Yes	Yes
	SIZE=	Yes	Yes
	TILT=	Yes	No

Statement	Option	Supported by ActiveX?	Supported by Java?
	XTICKNUM= YTICKNUM= ZTICKNUM=	Yes	Yes
	ZMAX= ZMIN=	Yes	Yes (partial)

Annotate Functions

ARROW

Table A1.21 ActiveX and Java Support for the ARROW Function

Variable	Supported by ActiveX?	Supported by Java?
ANGLE	Yes	Yes
COLOR	Yes	Yes
GROUP	Yes	Yes
HSYS	Yes	Yes
LINE	Yes	Yes
MIDPOINT	Yes	Yes
SIZE	Yes	Yes
STYLE	Yes	Yes
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

BAR

Table A1.22 ActiveX and Java Support for the BAR Function

Variable	Supported by ActiveX?	Supported by Java?
COLOR	Yes	Yes
GROUP	Yes	Yes
HTML	Yes	No

Variable	Supported by ActiveX?	Supported by Java?
LINE	Yes	Yes (Partial)
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

Table A1.23 ActiveX and Java Support for the DRAW Function

Variable	Supported by ActiveX?	Supported by Java?
COLOR	Yes	Yes
GROUP	Yes	Yes
HSYS	Yes	Yes
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

Table A1.24 ActiveX and Java Support for the DRAW2TXT Function

Variable	Supported by ActiveX?	Supported by Java?
COLOR	Yes	Yes
HSYS	Yes	Yes
LINE	Yes	Yes

Variable	Supported by ActiveX?	Supported by Java?
SIZE	Yes	Yes
WHEN	Yes	Yes

FRAME

Table A1.25 ActiveX and Java Support for the FRAME Function

Variable	Supported by ActiveX?	Supported by Java?
COLOR	Yes	No
HSYS	Yes	No
HTML	Yes	No
LINE	Yes	No
SIZE	Yes	No
STYLE	Yes	No
WHEN	Yes	No
XSYS, YSYS	Yes	No

IMAGE

Table A1.26 ActiveX and Java Support for the IMAGE Function

Variable	Supported by ActiveX?	Supported by Java?
HTML	Yes	Yes
IMGPATH	Yes	Yes
STYLE	Yes	Yes
WHEN	Yes	Yes
X, Y	Yes	Yes
XSYS, YSYS	Yes	Yes

LABEL

Table A1.27 ActiveX and Java Support for the LABEL Function

Variable	Supported by ActiveX?	Supported by Java?
ANGLE	Yes	No
CBORDER	Yes	Yes

Variable	Supported by ActiveX?	Supported by Java?
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

Table A1.28 ActiveX and Java Support for the MOVE Function

Variable	Supported by ActiveX?	Supported by Java?
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

Table A1.29 ActiveX and Java Support for the PIE Function

Variable	Supported by ActiveX?	Supported by Java?
ANGLE	Yes	Yes
COLOR	Yes	Yes
GROUP	Yes	Yes
HSYS	Yes	Yes
HTML	Yes	No
LINE	Yes	Yes (Partial)
MIDPOINT	Yes	Yes
ROTATE	Yes	Yes
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

Table A1.30 ActiveX and Java Support for the PIECNTR Function

Variable	Supported by ActiveX?	Supported by Java?
GROUP	Yes	Yes
HSYS	Yes	Yes
MIDPOINT	Yes	Yes
SIZE	Yes	Yes
SUBGROUP	Yes	Yes
WHEN	Yes	Yes
X, Y, Z	Yes	Yes

Variable	Supported by ActiveX?	Supported by Java?
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

PIEXY

Table A1.31 ActiveX and Java Support for the PIEXY Function

Variable	Supported by ActiveX?	Supported by Java?
ANGLE	Yes	Yes
SIZE	Yes	Yes
WHEN	Yes	Yes

POINT

Table A1.32 ActiveX and Java Support for the POINT Function

Variable	Supported by ActiveX?	Supported by Java?
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

Table A1.33 ActiveX and Java Support for the POLY Function

Variable	Supported by ActiveX?	Supported by Java?
COLOR	Yes	Yes
GROUP	Yes	Yes
HTML	Yes	No
LINE	Yes	No

Variable	Supported by ActiveX?	Supported by Java?
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

Table A1.34 ActiveX and Java Support for the POLYCONT Function

Variable	Supported by ActiveX?	Supported by Java?
COLOR	Yes	Yes
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

SYMBOL

Table A1.35 ActiveX and Java Support for the SYMBOL Function

Variable	Supported by ActiveX?	Supported by Java?
CBOX	Yes	No
CBORDER	Yes	No
COLOR	Yes	Yes
GROUP	Yes	Yes
SUBGROUP	Yes	Yes
HSYS	Yes	Yes
HTML	Yes	No
MIDPOINT	Yes	Yes

Variable	Supported by ActiveX?	Supported by Java?
SIZE	Yes	Yes
STYLE	Yes(Partial)	Yes (Partial)
TEXT	Yes (Partial)	Yes (Partial)
WHEN	Yes	Yes
X, Y, Z	Yes	Yes
XC, YC	Yes	Yes
XSYS, YSYS, ZSYS	Yes	Yes

APPENDIX

2

Using SAS/GRAPH Fonts

Introduction

SAS/GRAPH fonts are the entries in the SASHELP.FONTS catalog. Information on 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 "Devices" on page xvii), 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 155 and "TrueType Fonts That Are Supplied by SAS" on page 156 for more information.

Note: The Java and ActiveX devices do not support SAS/GRAPH fonts. SAS/GRAPH fonts cannot be used with template-based graphics (see "Device-Based Graphics and Template-Based Graphics" on page 6). △

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 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 Chapter 15, "Graphics Options and Device Parameters Dictionary," on page 327 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 on page 1199.

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 A2.1 Roman Alphabet Text Fonts

Type Style	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	
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
Bold Italic Empty	SWISSBIE	*A B C a b c 1 2 3*	

Type Style	Font Name	Type Sample	Uniform Font
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 A2.2 Non-Roman Alphabet Fonts

Type Style	Font Name	Uniform Font Name
Arabic	ARABIC	
Arabic Empty	ARABICE	
Cyrillic	CYRILLIC	CYRILLIU
David	DAVID	
Davidb	DAVIDB	
Fsong	FSONG	FSONGU
Greek	GREEK	GREEKU
Greek (serif)	CGREEK	CGREEKU
Hebrew	HEBREW	
Hebrew	NHEBREW*	
Hebrewb	HEBREWB	
Hebrew Empty	HEBREWE	

Type Style	Font Name	Uniform Font Name
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	
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 A2.1 Greek (GREEK)

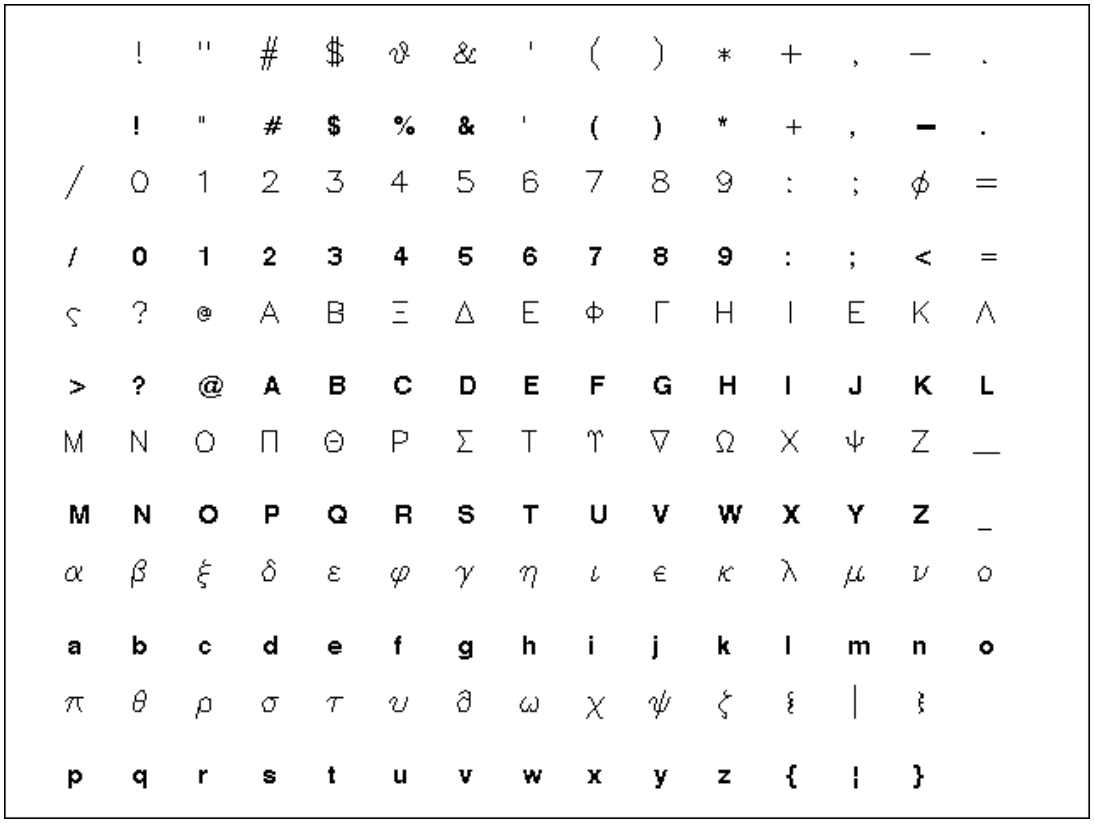


Figure A2.2 Greek Serif (CGREEK) Font

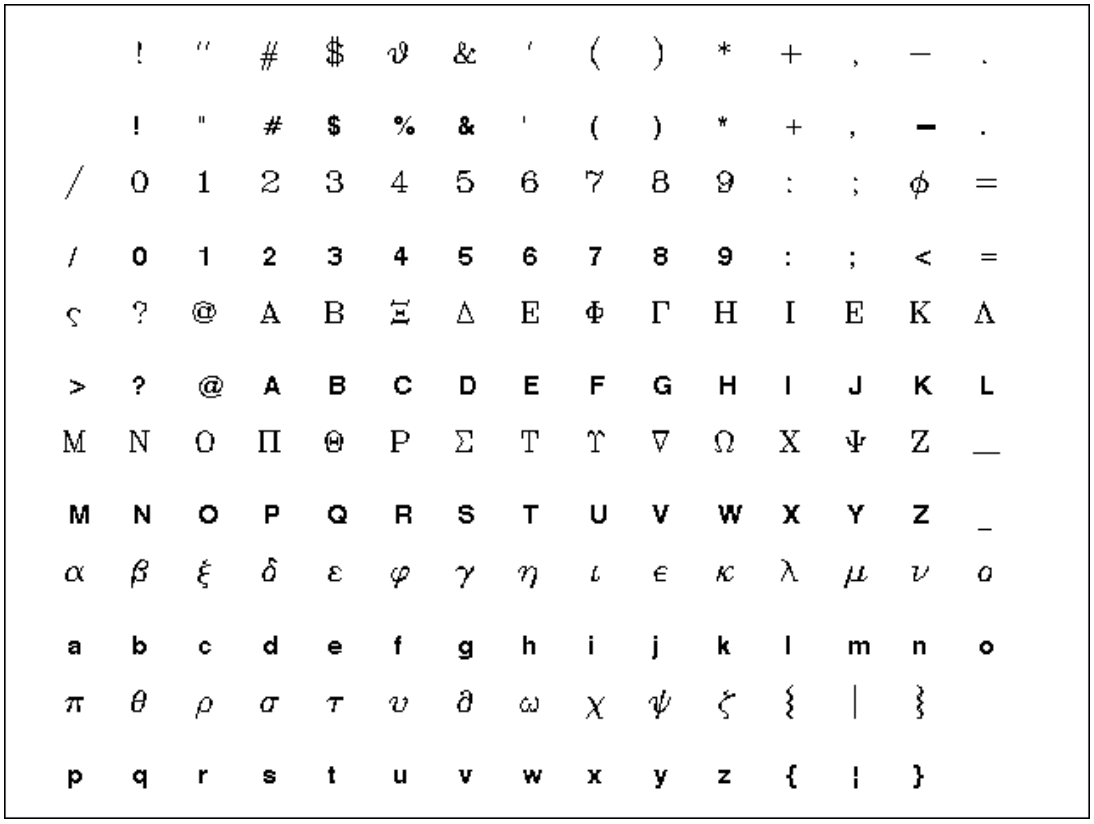


Table A2.3 Symbol Fonts

Type Style	Font Name	Uniform Font Name
Cartographic	CARTOG	CARTOGU
Electronic	ELECTRON	ELECTROU
Marker	MARKER	
Marker Empty	MARKERE *	
Math	MATH	MATHU
Music	MUSIC	MUSICU
Special	SPECIAL	SPECIALU
Weather	WEATHER	WEATHERU

*MARKERE is not displayed in the figures.

Figure A2.3 Cartographic Font

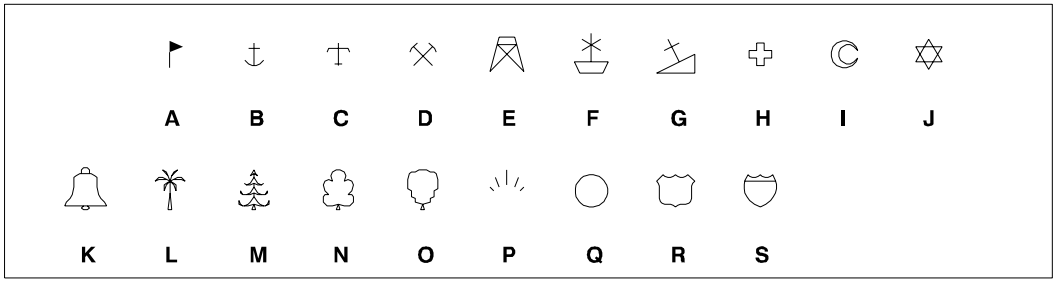
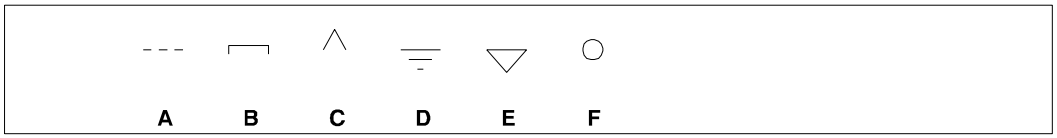


Figure A2.4 Electronic Font



Note: Figure A2.5 on page 1650 shows the MARKER font. The MARKERE font produces the same symbols but in empty (outline) form. △

Figure A2.5 Marker Font

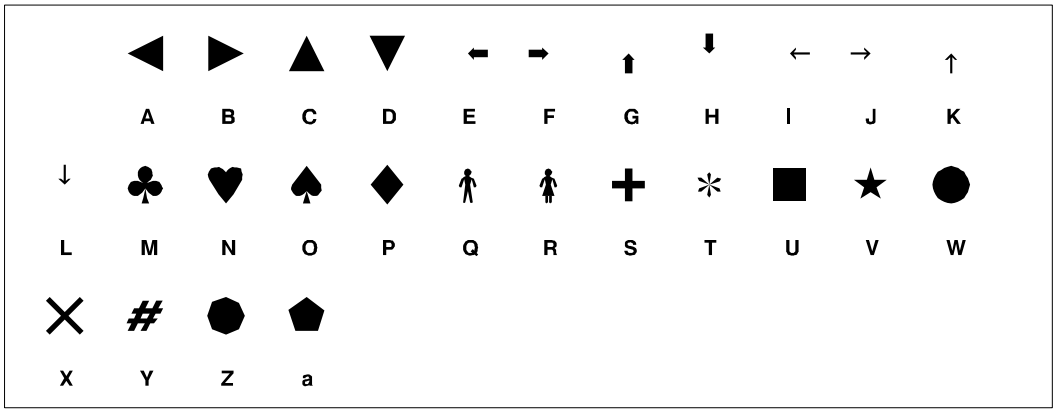


Figure A2.6 Math Font

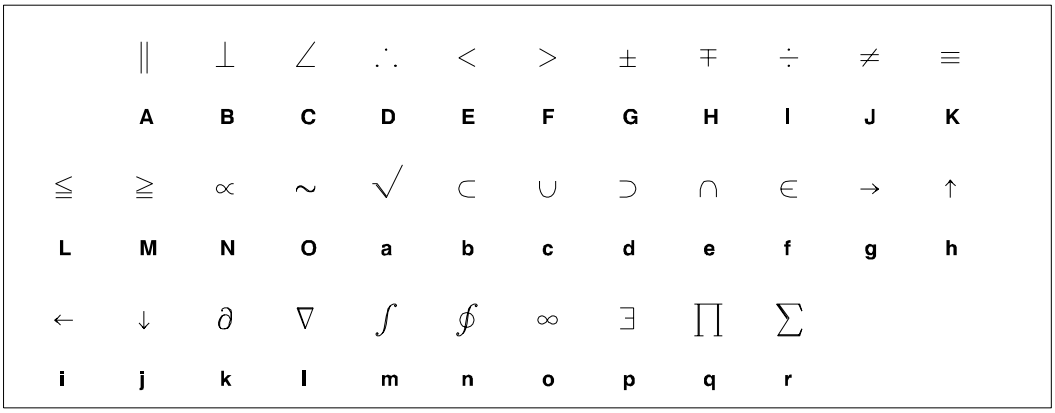


Figure A2.7 Music Font

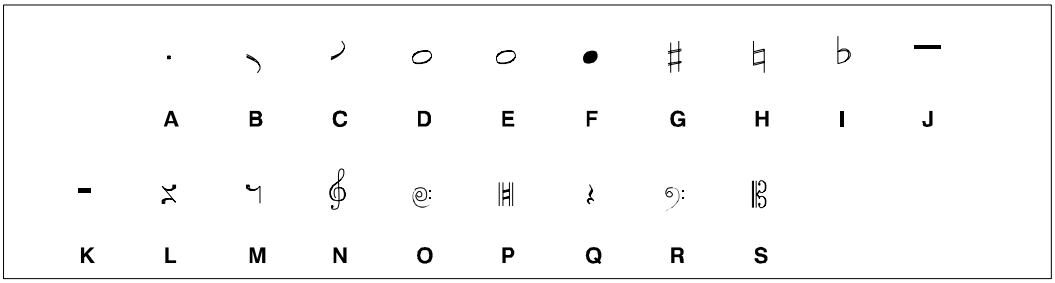


Figure A2.8 Special Font

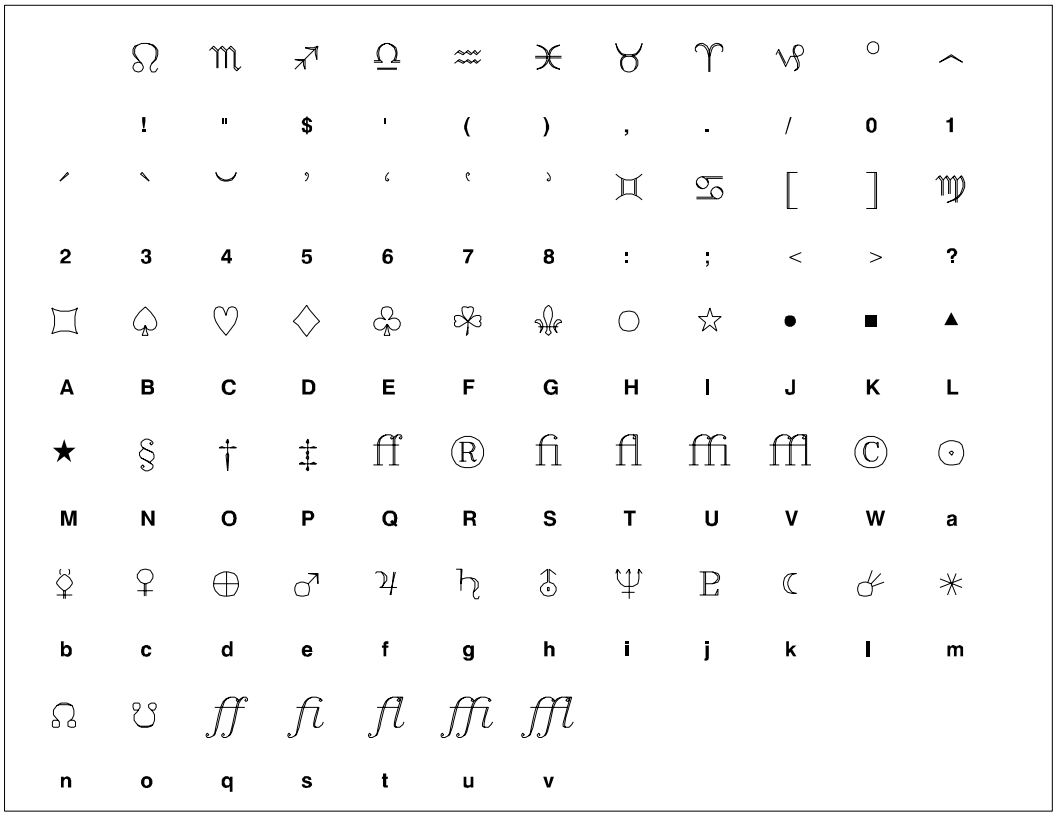
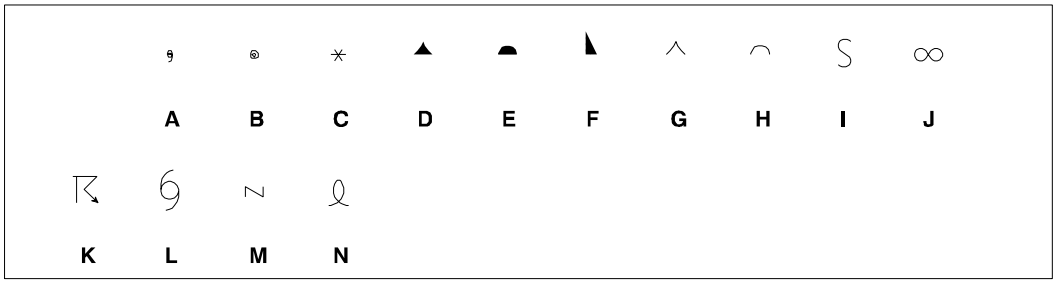


Figure A2.9 Weather Font



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” on page 420 for more information on 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 1652 for more information.

See Chapter 40, "The GFONT Procedure," on page 1175 for additional information on specifying the libref GFONT0.

APPENDIX

3

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 1656 for details.
- by using the GDEVICE procedure to specify the number of the font you want to use as the default device-resident font. See "Using the GDEVICE Procedure to Change the Default Device-Resident Font" on page 1656 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 1657 for details.
- by explicitly specifying a device-resident font name of the type HW*xxxnnn*. See "Specifying Alternative Device-Resident Fonts" on page 1657 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

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 157 for details on when device-resident fonts cannot be used.

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 1656.
- 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 1656 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" on page 338 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, "The GDEVICE Procedure," on page 1125 for information on 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 on 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 on 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 only displays 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 quotes and single quotes (see Chapter 30, “Annotate Dictionary,” on page 667 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 Chapter 15, “Graphics Options and Device Parameters Dictionary,” on page 327 for additional information.

APPENDIX

4

Transporting and Converting Graphics Output

About Transporting and Converting Graphics Output

You can use the following methods to transport and convert graphics output within the SAS System:

- 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

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 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 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 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 the SAS System. The CIMPORT procedure in a previous release cannot read a transport file created by the CPORT procedure in the current version. △

For details on 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, for example from Version 6 to Version 8, 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 on 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;
```

APPENDIX

5

GREPLAY Procedure Template Code

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 on using, creating, and modifying templates, refer to Chapter 50, "The GREPLAY Procedure," on page 1473.

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
        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 linemode. 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
   tdef L2R2S des="2 BOXES LEFT, 2 BOXES RIGHT (WITH SPACE)"
        1/llx=0   lly=52
          ulx=0   uly=100
          urx=48  ury=48
          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, 1 BOX  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 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 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= "Three 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=70  lry=0
          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

6

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *Annotate: Simply the Basics*
- *Forecasting Examples for Business and Economics Using SAS*
- *Maps Made Easy Using SAS*
- *Multiple-Plot Displays: Simplified with Macros*
- *Output Delivery System: The Basics and Beyond*
- *Quick Results with SAS/GRAPH Software*
- *SAS/GRAPH: Graph Template Language Reference*
- *SAS/GRAPH: Graph Template Language User's Guide*
- *SAS/GRAPH: Network Visualization Workshop User's Guide*
- *SAS/GRAPH: ODS Graphics Designer User's Guide*
- *SAS/GRAPH: ODS Graphics Editor User's Guide*
- *SAS/GRAPH: Statistical Graphics Procedures Guide*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *SAS Output Delivery System: User's Guide*
- *The How-To Book for SAS/GRAPH Software*

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

$GEOREF format
a geometric coordinate data arrangement that stores all the spatial information as a geometry object contained in a single variable. This format, which is used by feature tables, references the geometry objects that encapsulate the points, lines, and polygons necessary to render a map.

absolute coordinate
a coordinate that is measured from the origin of a coordinate system.

ActiveX
a Microsoft proprietary (COM) component used to display an interactive graph. The output is stored in a single file.

ActiveX control
a type of Web application that is developed specifically for the Windows operating environment. ActiveX controls can provide Web users with interactive capabilities.

area bar chart
a bar chart that applies an additional magnitude of width to the bars that results in categorized bars each with a height measure and a width measure that can be independent of each other.

aspect ratio
the ratio of a shape's width to its height in an output area such as a display, plotter, or film recorder.

attribute
a characteristic of a graphics element such as color, line type, text font, text justification, and fill pattern.

axis
a line with data values indicated by tick marks as a reference to a data value or range of values. Graphs can support more than one axis, in which case, the axes are usually perpendicular. Axis refers collectively to the axis line, the major and minor tick marks, the major tick mark values, and the axis label. See also Cartesian coordinate system.

axis area
an area bounded by axes. In SAS/GRAPH software, this area might be enclosed by an axis frame.

background
a plane on which graphics are displayed such that they appear behind or beneath objects in the foreground.

baseline
in a font, the imaginary line upon which the characters rest.

block map
a three-dimensional map that uses blocks of varying heights to represent the value of a variable for each map area.

border
a line that is drawn around an entire graphics output area. This area usually includes the title and footnote areas as well as the procedure output area. See also frame.

boundary
in the GMAP procedure, a separating line or point that distinguishes between two or more unit areas or segments.

capline
the highest point of a normal uppercase letter. In some fonts, the capline might be above the top of the letter to allow room for an accent.

Cartesian coordinate system
the two- or three-dimensional coordinate system in which perpendicular axes meet at the origin (0,0) or (0,0,0). Typically, Cartesian coordinate axes are called X, Y, and Z.

cell
in traditional SAS/GRAPH procedures, a unit of measure that is defined by the number of rows and the number of columns in the graphics output area. In ODS Graphics, a cell refers to a distinct rectangular sub-region of a graph that contains plots, text, or legends. See also aspect ratio.

center point
the location in the GRAPH window that, in conjuction with a radius point, defines the placement and shape of an ellipse or a pie.

CGM
See computer graphics metafile.

character up vector
in SAS/GRAPH software, the angle at which a character is positioned. The character up vector has two components, x and y, which determine the angle.

chart
a graph in which graphical elements, such as bars or pie slices, represent a view of the data.

chart statistic
the statistical value calculated for the chart variable: frequency, cumulative frequency, percentage, cumulative percentage, sum, or mean.

chart variable
a variable in the input data set whose values are categories of data represented by bars, blocks, slices, or spines.

chart vertices
points on a radar chart where the statistical values intersect the spokes.

choropleth map
a two-dimensional map that uses color and fill pattern combinations to represent different categories or levels of magnitude.

classification variable
a variable that is used to group (or classify) data. A classification variable can be either character or numeric values. Classification variables include group, subgroup, category, and BY variables.

CMYK
A color coding scheme that specifies a color in terms of the levels of cyan, magenta, yellow, and black components. The level of each component ranges from 0 to 255.

color list
in SAS/GRAPH software, the list of foreground colors available for the graphics output. The color list is either the default list established from the style, the list created from the device entry, or the list established from the colors specified with the COLORS= graphics option.

color map
in SAS/GRAPH software, a table that is used to translate the original colors in graphics output to different colors when replaying graphics output using the GREPLAY procedure. The table is contained in a catalog entry.

color, predefined
one of the set of colors for which SAS/GRAPH software defines and recognizes names (for example, BLACK, BLUE, and CYAN).

color, user-defined
in SAS/GRAPH software, a color expressed in RGB, HLS, or gray-scale format.

computer graphics metafile
a graphics output file written in the internationally recognized format for describing computer graphics images. This standardization allows any image in a CGM to be imported and exported among different systems without error or distortion. Short form: CGM.

confidence limits
the upper and lower values of a (usually 95%) confidence interval. In repeated sampling, approximately (1-alpha) 100% of the resulting intervals would contain the true value of the parameter that the interval estimates (where alpha is the confidence level associated with the interval).

contour plot
a three-variable plot that uses line styles or patterns to represent levels of magnitude of z corresponding to x and y coordinates.

coordinate
a value that represents the location of a data point or a graphics element with respect to a coordinate system.

coordinate system
the context in which to interpret coordinates. Coordinate systems vary according to their origin, limits, and units.

data area
the portion of the graphics output area in which data values are displayed. The data area is bounded by axes or map areas. In the Annotate facility, the data area defines a coordinate system. See also graphics output area, procedure output area, and coordinate system.

data tip
data or other detailed information that is displayed when a user positions a mouse pointer over an element in a graph. For example, a data tip typically displays the data value that is represented by a bar, a plot point, or some other element.

density value
a value assigned to each observation in a map data set reflecting the amount of detail (resolution) contributed by the observation.

dependent variable
a variable (response variable) whose value is determined by the value of another variable or by the values of a set of variables in a statistical model.

device driver
in SAS/GRAPH software, a routine that generates the specific machine-language commands needed to display graphics output on a particular device. SAS/GRAPH device drivers take device-independent graphics information produced by SAS/GRAPH procedures and create the commands required to produce the graph on the particular device.

device entry
a SAS catalog entry that stores the values of device parameters (or the characteristics) that are used with a particular output device. A device entry is a SAS catalog entry of type DEV.

device map
a catalog entry used to convert the SAS/GRAPH internal encoding for one or more characters to the device-specific encoding needed to display the characters in hardware text on a particular graphics output device. See also hardware character set.

device parameter
a value in a device entry that defines a default behavior or characteristic of a device driver. Some device parameters can be overridden by graphics options. See also graphics option.

device-independent catalog entry
a SAS catalog entry that contains graphics output in a generic format (not device-specific). A device-independent catalog entry can be replayed on any device supported by SAS/GRAPH software. See also device-dependent catalog entry.

device-resident font
a font stored in an output device.

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.

drill down
to select an element in an image in order to display additional information about that element, generally by displaying another Web page or another section in the same Web page.

end angle
for an ellipse, the measure in degrees from the major axis to the trailing edge.

export
in SAS/GRAPH software, to put a SAS catalog entry containing graphics output into a format that can be moved to another software product.

feature table
A SAS data set that uses the $GEOREF format to store geometric coordinates for each unique map area in a single variable value.

fill color
the color of a pattern in a filled, closed graphics object, such as a bar segment, a pie slice, or a map area.

font
a complete set of all the characters of the same design and style. The characters in a font can be figures or symbols as well as alphanumeric characters. See also type style.

font maximum
in the GFONT procedure, the highest vertical coordinate in a font.

font minimum
in the GFONT procedure, the lowest vertical coordinate in a font.

font units
in the GFONT procedure, units defined by the range of coordinates specified in the font data set. For example, a font in which the vertical coordinates range from 10 to 100 has 90 font units.

font, device-resident
a font stored in an output device.

font, SAS/GRAPH
a font stored in the SASHELP.FONTS catalog, and a font created by the user and stored in a GFONTn 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.

font, system
a font that can be used by any SAS procedure and by other software, such as Microsoft Word.

frame
in SAS/GRAPH software, a box enclosing a group of graphics elements. In GSLIDE procedure output, the frame encloses the procedure output area. In GPLOT, GCHART, GBARLINE, and GCONTOUR procedure output, the frame encloses the axis area. In a legend, the frame encloses the legend label and entries. See also border.

FreeType font-rendering
a method of rendering fonts that uses the FreeType engine to access the content of font files in order to render high-quality fonts for ODS and SAS/GRAPH. The FreeType engine can be used in all SAS operating environments.

geocoding
the process of adding geographic coordinates (latitude and longitude values) to an address. Each pair of coordinates can represent either the center of a region or a specific point.

geo-variable
in a feature table, the $GEOREF formatted variable that stores the spatial information as a geometry object. When a feature table is used, this variable is specified in the ID statement of the GMAP procedure.

global statement
a SAS statement that you can specify anywhere in a SAS program.

graph
a visual representation of data showing the variation of a variable in comparison to one or more other variables.

graphics element
a discrete visual part of a picture. For example, a bar in a chart and a plot's axis label are both graphics elements.

graphics object
a discrete visual element of a graph or picture (for example, a bar in a chart, a polygon, a plot's axis, and so on).

graphics option
in a SAS GOPTIONS statement, an option that controls some attribute of the graphics output. The specified value remains in effect only for the duration of the SAS session. Some graphics options override parameters that have been specified for a graphics output device.

graphics output
output from a graphics program that can be stored as a catalog GRSEG entry or as a graphics stream file. Graphics output can be displayed or printed on a graphics output device. See also device-dependent catalog entry, device-independent catalog entry, and graphics output device.

graphics output area
the area of a graphics output device where the graphics output is displayed or drawn. Typically, the graphics output area occupies the full drawing area of the device, but the dimensions of the graphics output area can be changed with graphics options or device parameters. See also procedure output area and graphics output device.

graphics output device
any terminal, printer, or other output device that is capable of displaying or producing graphical output.

graphics output file
a file that contains bitmapped or vector graphic information.

graphics primitive
a function that draws a graphics element.

graphics stream file
a file that contains device-dependent graphics commands from a SAS/GRAPH device driver. This file can be sent to a graphics device or to other software applications. Short form: GSF.

gray scale
a color-coding scheme that specifies a color in terms of gray components. Gray-scale color codes are commonly used with some laser printers and PostScript devices.

grid point
a grid location in the GRAPH window that is marked by a dot. Grid points are used for precision placement of objects.

grid request
in the G3GRID procedure, the request specified in a GRID statement that identifies the horizontal variables that identify the x, y plane and one or more z variables for the interpolation.

group variable
a variable in the input data set used to categorize chart variable values into groups.

GRSEG
a SAS catalog entry that contains graphic output in a generic format (not device-specific).

GSF
See graphics stream file.

handshaking
the exchange of signals between two devices over an interface for control or synchronization purposes. Data flow control is needed to ensure that data are not sent faster than the receiving device can process them. Handshaking usually involves sending signals between the device and the host computer in order to start and stop transmission of data.

hardware (or hardwire) handshaking
a method of data flow control in which the flow of data between the computer and device is regulated by signals sent over separate wires in the connecting cable. See also handshaking.

hardware character set
a set of character definitions held internally in a graphics output device. When a hardware character set is used, SAS/GRAPH software does not have to send the device all the commands to draw characters, only the corresponding character codes. Some devices have more than one hardware character set. See also font and device-resident.

hatch
a fill pattern consisting of parallel lines at any specified angle.

HLS
a color-coding scheme that specifies a color in terms of its hue, lightness, and saturation components. Hue is the color, lightness is the percentage of white, and saturation is the attribute of a color that determines its relative strength and its departure from gray. Lightness and saturation added to the hue produce a specific shade. See also RGB.

host computer
a workstation or minicomputer accessed by a terminal or another workstation.

host font-rendering
a method of rendering fonts that relies on the capabilities of the operating environment.

HSV (HSB)
a color-coding scheme that specifies a color in terms of its hue, saturation, and value (brightness) components. Hue is the color. Saturation is the aspect of a color that determines its relative strength and departure from gray. And value (brightness) is the color's departure from black.

identification variable
a variable common to both the map data set and the response data set that the GMAP procedure uses to associate each pair of map coordinates and each response value with a unique map area.

image file
a file that contains bitmapped graphic information. Examples include GIF, PNG, TIFF, and JPEG files. Image files are a subset of graphics output files.

image map
a diagram that associates graphic elements with HTML links to implement drill-down functionality. The graphic elements are represented by sets of coordinates. SAS/GRAPH software generates image maps on demand with the IMAGEMAP= option or with the IMAGEMAP macro.

import
to restore a SAS transport file to its original form (a SAS library, a SAS catalog, or a SAS data set) in the format that is appropriate for the host operating system. You use the CIMPORT procedure to import a SAS transport file that was created by the CPORT procedure.

include
in the graphics editor, to read in or link to a graph other than the one currently being edited.

independent variable
in SAS/GRAPH software, a variable whose value, in part, determines the value of a dependent (or response) variable. In a plot, an independent variable typically appears on the X (or horizontal) axis.

interactive graph
SAS/GRAPH output that features user controls such as menus, buttons, and pictures that a user can manipulate. The controls are driven by a Java applet or an ActiveX control.

interpolate
to estimate values that are between two or more known values.

Joint Photographic Experts Group
a file format that is used for storing noninteractive images. If you generate a chart or graph in JPEG format, you cannot subsequently change its appearance. This format is best suited to complex graphics that have many colors, because it supports 16 million colors. Short form: JPEG.

JPEG
See Joint Photographic Experts Group.

justify
to position text in relation to the left or right margin or the center of the line.

key map
a SAS catalog entry used to translate the codes generated by the keys on a keyboard into their corresponding SAS/GRAPH internal character encoding. See also device map.

label
(1) descriptive text associated with a variable. By default, this text is the name of a variable or of a label previously assigned with the LABEL= option. (2) in special cases of pie charts and star charts in the GCHART procedure, the label is the midpoint value and the value of the chart statistics for a slice or spine.

latitude
used with maps, the angular measure between the equator and the circle of parallel on which a point lies.

legend
a visual key to graphic elements in a graph. The legend consists of the legend value, the legend value description, the legend label, and the legend frame.

link to
in the graphics editor, to include one graph into another by placing a template of that graph in the current graph. The template acts as a placeholder and can be resized; it creates a connection between the graph being edited and the linked-to graph such that any changes made to the linked-to graph are reflected in the graph where a template is placed.

longitude
used with maps, the angular measure between the reference meridian and the plane intersecting both poles and a point. The reference meridian, called the prime meridian, is assigned a longitude of 0, and other longitude values are measured from there in appropriate angular units (degrees or radians, for example).

major axis
in the graphics editor, the longest axis of a graphics object.

major tick marks
the points on an axis that mark the major divisions of the axis scale. See also minor tick marks.

map
a graphic representation of an area. The area is often a geographic area, but it can also be any other area of any size. See also device map and key map.

map area
a polygon or group of polygons on a map. For example, states, provinces, and countries are typical map areas. In a map data set, a map area consists of all the observations that have the same values for the identification variable or variables. A map area is sometimes referred to as a unit area. See also identification variable.

map data set
a SAS data set that contains information that the GMAP procedure uses to draw a map. Each observation in the data set contains variables whose values are the x, y coordinates of a point on the boundary of a map area. In addition, each observation contains an identification variable whose value identifies the map area that the point belongs to.

mapping
in the GMAP procedure, the process of displaying data values on a map.

marker
a symbol such as a dot, a cross, or a diamond, that is used to indicate the location of a data point on a plot or graph.

meridian
an imaginary circle of constant longitude around the surface of the earth perpendicular to the equator. See also parallel.

metafile
a file, produced by the Metagraphics facility internal driver, that contains device-independent graphics commands in a special format. A user-written external driver routine is required to read and process the metafile.

Metagraphics driver
a type of SAS/GRAPH device driver that can be written by users. A Metagraphics driver consists of an internal driver (supplied with SAS/GRAPH software), which writes a metafile in a special format, and an external driver (written by the user), which decodes the metafile and writes device-specific commands.

midpoint
a value that represents the middle of a range of data values.

minor axis
in the graphics editor, the shortest axis of a graphics object.

minor tick marks
the divisions of an axis scale that fall between major tick marks. See also major tick marks.

needle plot
a plot in which data points are connected by a vertical line which connects to a horizontal baseline. The baseline intersects the 0 value, or the minimum value on the vertical axis.

node
a connection point between two or more links. In a node-line graph, nodes are typically represented as a box and enable you to access information and possibly to traverse the graph by drilling up or down in the structure.

offset
the distance between a graphics object's original position and its new position when it is moved. Offsets can be specified for legends, axes, an entire graph, or other graphics object.

origin
in a three-dimensional graph, the point at which the X, Y, and Z axes intersect. In a two-dimensional graph, the point at which the X and Y axes intersect.

panel
in the GREPLAY procedure, a part of the template in which one or more pictures can be displayed. A template can contain one or more panels.

parallel
an imaginary circle of constant latitude around the surface of the earth parallel to the equator. See also meridian.

pattern type
in SAS/GRAPH software, the set of fill patterns that are valid for a particular type of graph. The PATTERN statement supports three pattern types: bar and block patterns, map and plot patterns, and pie and star patterns. See also fill pattern.

pie chart
a circular chart that is divided into slices by radial lines. Each slice represents the relative contribution of each part to the whole.

pixel
an element of an electronic image. A pixel is the smallest element on a display that can be assigned a separate color.

plot
a graph in which graphics elements such as markers or lines represent a view of the data. See also coordinates.

plot line
the line joining the data points in a plot.

plotter
a class of graphics devices that typically use pens to draw hard-copy output.

PNG
See Portable Network Graphic.

polygon font
a SAS/GRAPH font in which the characters are drawn with enclosed areas that can be either filled or empty. See also stroked font.

polyline
in SAS/GRAPH software, a graphics object composed of connected line segments that might have attributes. A polyline is not a closed object; therefore, it cannot be filled with a pattern.

Portable Network Graphic
a file format that returns the graphical output in separate files and that produces a static image. This format is similar to the GIF format, but has additional features, such as support for true-color images and better compression. Short form: PNG.

PostScript
a device-independent page description language for printing high-resolution integrated text and graphics.

prism map
a three-dimensional map that uses prisms (polyhedrons with two parallel surfaces) of varying height to indicate the ordinal magnitude of a response variable.

procedure output area
the portion of the graphics output area where the output from a graphics procedure is displayed. See also graphics output area and data area.

projection
in SAS/GRAPH software, a two-dimensional map representation of unit areas on the surface of a sphere (for example, geographic regions on the surface of the Earth).

prompt character
a character sent by the host computer to a device to signal that the host has finished transmitting data and is ready for a response from the device.

protocol
a set of rules that govern data communications between computers and peripheral devices.

radar chart
a chart that shows the relative frequency of data measures with statistics displayed along spokes that radiate from the center of the chart. The charts are often stacked on top of one another with circular reference lines, thus giving them the look of a radar screen. See also star chart.

rasterizer
a device that accepts commands (such as moves and draws) as input and that converts those commands into a bit-map. Rasterizers are connected between host computers and graphics output devices that require bitmapped input.

region
in the graphics editor, an area in the GRAPH window containing more than one graphics objects.

regression analysis
an analysis that models a dependent (or response) variable as a function of one or more independent (or predictor) variables. The regression line, which is the set of predictions from the model, appears as a line or curve in a plot of the dependent variable against an independent variable.

relative coordinate
a coordinate that is measured from a point other than the origin. In the Annotate facility, this point is usually the endpoint of the last graphics element that was drawn. See also absolute coordinate.

replay
in SAS/GRAPH software, to display graphics output that is stored in a catalog entry using the GREPLAY procedure.

response data set
a SAS data set used by the GMAP procedure that contains data values associated with map areas and one or more identification variables. See also identification variable, response values, and response variable.

response levels
the individual values or ranges of values into which the GMAP or GCHART procedure divides the response variable. See also midpoint.

response values
values of a response variable that the GMAP procedure represents on a map as different pattern/color combinations, or as raised map areas (prisms), spikes, or blocks of different heights. The GCHART procedure represents response values as bars, slices, spines, or blocks. See also midpoint.

response variable
(1)in the GMAP procedure, a variable in the response data set that contains data values that are associated with a map area. (2) In the GCHART, GBARLINE, and GCONTOUR procedures, a variable whose value is determined by the value of another variable or by the values of a set of variables. See also chart variable, response data set, response levels, and response values.

RGB
a color-coding scheme that specifies a color in terms of amounts of red, green, and blue components. See also HLS.

SAS/GRAPH font
a font stored in the SASHELP.FONTS catalog, and a font created by the user and stored in a GFONTn catalog. These fonts can be used only by SAS/GRAPH procedures or other procedures that generate GRSEG catalog entries. Examples of SAS/GRAPH fonts include Swiss, Simulate, and Marker. These fonts are provided for specialized purposes only.

scatter plot
a two- or three-dimensional plot that shows the joint variation of two (or three) variables from a group of observations. The coordinates of each point in the plot correspond to the data values for a single observation.

segment
in the GMAP procedure, a polygon that is a part of a unit area consisting of more than one polygon. For example, the representation of the state of Hawaii is a single unit area which consists of a group of individual segments representing the islands, each of which is a separate polygon. In the GFONT procedure, a segment is a single continuous line that forms part of all of a character or symbol. In the DATA Step Graphics Interface (DSGI), a segment is one or more graphics primitives that can be manipulated as a unit.

select
in the graphics editor, to choose an action or a graphics object. Once a graphics object has been selected, it can be copied, deleted, or otherwise manipulated.

snap
in the graphics editor, to automatically place graphics objects in the grid display area with precision.

software handshaking
a method of data flow control in which a device and a computer exchange predefined sequences of characters to indicate when data should be transmitted between the two. See also handshaking and hardware (or hardwire) handshaking.

spine
a line on a star chart used to represent the relative value of the chart statistic for a midpoint. Spines are drawn outward from the center of the chart.

spline
in SAS/GRAPH software, a method of interpolation in which a smooth line or surface connects data points.

spokes
lines that radiate from the center of a radar or star chart. These lines represent statistical information.

standard deviation
a statistical measure of the variability of a group of data values. This measure, which is the most widely used measure of the dispersion of a frequency distribution, is equal to the square root of the variance.

star chart
a chart that shows the values of chart statistics as either spines of varying lengths or slices of varying sizes. Star charts display statistics in a circle surrounding the spines or slices. See radar chart.

static graph
SAS/GRAPH output in the form of an image.

stroked font
in SAS/GRAPH software, a font in which the characters are drawn with discrete line segments or circular arcs. See also polygon font.

style attribute
A visual property such as color, font properties, and line characteristics that have reserved names and values defined in ODS. Style attributes are collectively referenced by a style element.

subgroup variable
the variable in the input data set for a chart that is used to proportionally fill areas of the bars or blocks on a bar chart, or to identify separate rings of a pie chart.

summary variable
a variable in an input data set whose values some SAS/GRAPH procedures total or average to produce the sum or mean statistics, respectively.

surface map
a three-dimensional map that uses spikes of varying heights to indicate levels of relative magnitude.

surface plot
a three-dimensional graph that displays a grid-like surface formed by the values of the vertical (Z) variable plotted on a plane specified by the X and Y variables.

system font
a font 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 generally provide the highest quality output.

template
(1) in the GRSEG graphics editor, a representation of a graph being linked to; a template is considered a single graphics object. (2) in the GREPLAY procedure, a framework that enables you to display one or more pictures on a page. (3) in ODS

graphics (template-based graphics), a description of how output should appear when it is formatted.

templated graph
(1) in the GRSEG graphics editor, the graph to which a template links. (2) in the GREPLAY procedure, graphics output that is created by replaying one or more catalog entries of type GRSEG (graphics output) into panels in a template.

thumbnail
a small image that can be selected in order to display a larger image.

TIFF
Tagged Image File Format. An industry-standard file format for storing compressed images. The Tagged Image File Format specifies compression routines and file formats for a variety of image types, including bilevel, grayscale, and color.

tile chart
a graph that represents the relative values of data by using rectangular areas. The color of each area represents the value of one measure in the query. The size of each area represents the value of the another measure in the query. The academic term for a tile chart is 'treemap.'

tilt angle
the measure in degrees from the horizontal axis to the major axis of an object.

tool palette
in the graphics editor, the collection of icons that represent functions in the interface.

traditional map data set
a map data set that defines the boundaries of map areas by using X and Y coordinates. Each observation contains an identification variable whose value identifies the map area for that point. See also identification variable, map area, map data set, and spatial map data set.

transformation
in the DATA Step Graphics Interface (DSGI), a mapping of the window coordinates to the viewport coordinates.

translate
to change the location of a graphics object.

type style
a typeface design and its variations (for example, Swiss, Swiss Bold, and Swiss Italic). See also font.

unit
a single quantity of measuremen. In SAS/GRAPH software, units can have one of the following scales assigned: centimeters, percentages, points, inches, or cells.

unit area
See map area.

user-definable colors
the colors that can be defined using SAS color names, RGB (red, green, blue), HLS (hue, lightness, saturation), or gray-scale color equivalents.

viewport
in the DATA Step Graphics Interface (DSGI), a section of the display into which you place graphics elements or graphics output.

Web server
a server machine and software that enable organizations to share information through intranets and through the Internet.

window
in the DATA Step Graphics Interface (DSGI), a coordinate system that is used with a viewport and that can be defined by the user.

XON/XOFF handshaking
a method of data flow control in which the flow of data between a computer and a device is regulated by the transmission of XON (DC1) and XOFF (DC3) control characters between the device and the computer.

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.

SAS® Publishing Delivers!

Whether you are new to the work force or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart. Visit us online at support.sas.com/bookstore.

SAS® Press

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from SAS Press. Written by experienced SAS professionals from around the world, SAS Press books deliver real-world insights on a broad range of topics for all skill levels.

support.sas.com/saspress

SAS® Documentation

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information: SAS documentation. We currently produce the following types of reference documentation to improve your work experience:

- Online help that is built into the software.
- Tutorials that are integrated into the product.
- Reference documentation delivered in HTML and PDF – **free** on the Web.
- Hard-copy books.

support.sas.com/publishing

SAS® Publishing News

Subscribe to SAS Publishing News to receive up-to-date information about all new SAS titles, author podcasts, and new Web site features via e-mail. Complete instructions on how to subscribe, as well as access to past issues, are available at our Web site.

support.sas.com/spn

sas | THE POWER TO KNOW®